File under: T Date: 24 Mar 1984 Name: Tovar

Name: Tovar

Project: 3 Programmer: TVR

File Name: [illegible]1NNF.LST[F4A,SYS]

File Last Written: 18:20 24 Mar 1984

Time: 18:44 Date: 24 Mar 1984

Center for Computer Research
in Music and Acoustics
Department of Music
Stanford, California

```
COMMENT ⊗   VALID 00215 PAGES
C REC  PAGE   DESCRIPTION
C00001 00001
C00029 00002    01 0001         COMMENT !   VALID 00002 PAGES
C00030 00003    SLOE  March 23, 1984 20:58:09  file DSK:F41NNF.SLO  --  of -- F41NNF
C00031 00004    SLOE  March 23, 1984 20:58:09  file DSK:F41NN.SLO  --  of -- F41NNF
C00032 00005    SLOE  March 23, 1984 20:58:10  file DSK:F41NNF.SLO  --  of -- F41NNF
C00033 00006    SLOE  March 23, 1984 20:58:10  file DSK:F41AX.SLO  --  of -- F41NNF
C00038 00007    SLOE  March 23, 1984 20:59:11  file DSK:F4APR.SLO  --  of -- F41NNF
C00053 00008    SLOE  March 23, 1984 20:59:18  file DSK:F4APR.SLO  --  of -- F41NNF
C00070 00009    SLOE  March 23, 1984 20:59:29  file DSK:F4APR.SLO  --  of -- F41NNF
C00085 00010    SLOE  March 23, 1984 20:59:37  file DSK:F4APR.SLO  --  of -- F41NNF
C00101 00011    SLOE  March 23, 1984 20:59:45  file DSK:F4APR.SLO  --  of -- F41NNF
C00115 0001Z    SLOE  March 23, 1984 20:59:52  file DSK:F4APR.SLO  --  of -- F41NNF
C00126 00013    SLOE  March 23, 1984 20:59:59  file DSK:F41AX.SLO  --  of -- F41NNF
C00127 00014    SLOE  March 23, 1984 20:59:59  file DSK:F4MAP.SLO  --  of -- F41NNF
C00131 00015    SLOE  March 23, 1984 21:00:00  file DSK:F4MAP.SLO  --  of -- F41NNF
C00135 00016    SLOE  March 23, 1984 21:00:00  file DSK:F4MAP.SLO  --  of -- F41NNF
C00143 00017    SLOE  March 23, 1984 21:00:01  file DSK:F4MAP.SLO  --  of -- F41NNF
C00155 00018    SLOE  March 23, 1984 21:00:03  file DSK:F4MAP.SLO  --  of -- F41NNF
C00165 00019    SLOE  March 23, 1984 21:00:05  file DSK:F4MAP.SLO  --  of -- F41NNF
C00177 00020    SLOE  March 23, 1984 21:00:07  file DSK:F4MAP.SLO  --  of -- F41NNF
C00187 00021    SLOE  March 23, 1984 21:00:08  file DSK:F4MAP.SLO  --  of -- F41NNF
C00198 0002Z    SLOE  March 23, 1984 21:00:12  file DSK:F4MAP.SLO  --  of -- F41NNF
C00204 00023    SLOE  March 23, 1984 21:00:15  file DSK:F4MAP.SLO  --  of -- F41NNF
C00213 00024    SLOE  March 23, 1984 21:00:18  file DSK:F4MAP.SLO  --  of -- F41NNF
C00220 00025    SLOE  March 23, 1984 21:00:21  file DSK:F4MAP.SLO  --  of -- F41NNF
C00236 00026    SLOE  March 23, 1984 21:00:28  file STRING:  --  of -- F41NNF
C00242 00027    SLOE  March 23, 1984 21:00:31  file DSK:F4MAP.SLO  --  of -- F41NNF
C00251 00028    SLOE  March 23, 1984 21:00:33  file DSK:F4MAP.SLO  --  of -- F41NNF
C00260 00029    SLOE  March 23, 1984 21:00:37  file DSK:F4MAP.SLO  --  of -- F41NNF
C00266 00030    SLOE  March 23, 1984 21:00:39  file DSK:F4MAP.SLO  --  of -- F41NNF
C00273 00031    SLOE  March 23, 1984 21:00:42  file DSK:F4MAP.SLO  --  of -- F41NNF
C00278 0003Z    SLOE  March 23, 1984 21:00:44  file DSK:F4MAP.SLO  --  of -- F41NNF
C00294 00033    SLOE  March 23, 1984 21:00:51  file DSK:F4MAP.SLO  --  of -- F41NNF
C00302 00034    SLOE  March 23, 1984 21:00:53  file DSK:F4MAP.SLO  --  of -- F41NNF
C00314 00035    SLOE  March 23, 1984 21:00:58  file DSK:F4MAP.SLO  --  of -- F41NNF
C00327 00036    SLOE  March 23, 1984 21:01:06  file DSK:F4MAP.SLO  --  of -- F41NNF
C00344 00037    SLOE  March 23, 1984 21:01:16  file DSK:F4MAP.SLO  --  of -- F41NNF
C00346 00038    SLOE  March 23, 1984 21:01:17  file DSK:F4MAP.SLO  --  of -- F41NNF
C00354 00039    SLOE  March 23, 1984 21:01:20  file DSK:F4MAP.SLO  --  of -- F41NNF
C00366 00040    SLOE  March 23, 1984 21:01:26  file DSK:F4MAP.SLO  --  of -- F41NNF
C00383 00041    SLOE  March 23, 1984 21:01:34  file DSK:F4MAP.SLO  --  of -- F41NNF
C00387 0004Z    SLOE  March 23, 1984 21:01:36  file DSK:F4MAP.SLO  --  of -- F41NNF
C00398 00043    SLOE  March 23, 1984 21:01:41  file DSK:F4MAP.SLO  --  of -- F41NNF
C00404 00044    SLOE  March 23, 1984 21:01:43  file DSK:F4MAP.SLO  --  of -- F41NNF
C0042Z 00045    SLOE  March 23, 1984 21:01:51  file DSK:F4MAP.SLO  --  of -- F41NNF
C00439 00046    SLOE  March 23, 1984 21:01:58  file STRING:  --  of -- F41NNF
C00442 00047    SLOE  March 23, 1984 21:01:59  file DSK:F4MAP.SLO  --  of -- F41NNF
C00447 00048    SLOE  March 23, 1984 21:02:00  file DSK:F4MAP.SLO  --  of -- F41NNF
C00453 00049    SLOE  March 23, 1984 21:02:03  file DSK:F41AX.SLO  --  of -- F41NNF
C00454 00050    SLOE  March 23, 1984 21:02:03  file DSK:F4BASE.SLO  --  of -- F41NNF
C00466 00051    SLOE  March 23, 1984 21:02:09  file DSK:F4BASE.SLO  --  of -- F41NNF
C00482 0005Z    SLOE  March 23, 1984 21:02:15  file DSK:F4BASE.SLO  --  of -- F41NNF
C00493 00053    SLOE  March 23, 1984 21:02:19  file DSK:F4BASE.SLO  --  of -- F41NNF
C00509 00054    SLOE  March 23, 1984 21:02:27  file STRING:  --  of -- F41NNF
C00525 00055    SLOE  March 23, 1984 21:02:34  file DSK:F4BASE.SLO  --  of -- F41NNF
C00526 00056    SLOE  March 23, 1984 21:02:34  file DSK:F41AX.SLO  --  of -- F41NNF
C00527 00057    SLOE  March 23, 1984 21:02:34  file DSK:F4INST.SLO  --  of -- F41NNF
C00539 00058    SLOE  March 23, 1984 21:02:41  file STRING:  --  of -- F41NNF
C00551 00059    SLOE  March 23, 1984 21:02:48  file STRING:  --  of -- F41NNF
C00565 00060    SLOE  March 23, 1984 21:02:59  file STRING:  --  of -- F41NNF
C00578 00061    SLOE  March 23, 1984 21:03:08  file DSK:F4INST.SLO  --  of -- F41NNF
C00588 00062    SLOE  March 23, 1984 21:03:12  file DSK:F4INST.SLO  --  of -- F41NNF
C00595 00063    SLOE  March 23, 1984 21:03:16  file DSK:F4INST.SLO  --  of -- F41NNF
C00612 00064    SLOE  March 23, 1984 21:03:24  file STRING:  --  of -- F41NNF
C00621 00065    SLOE  March 23, 1984 21:03:27  file DSK:F4INST.SLO  --  of -- F41NNF
C00637 00066    SLOE  March 23, 1984 21:03:34  file STRING:  --  of -- F41NNF
C00643 00067    SLOE  March 23, 1984 21:03:37  file DSK:F4INST.SLO  --  of -- F41NNF
C00651 00068    SLOE  March 23, 1984 21:03:40  file DSK:F4INST.SLO  --  of -- F41NNF
C00667 00069    SLOE  March 23, 1984 21:03:49  file DSK:F4INST.SLO  --  of -- F41NNF
C00672 00070    SLOE  March 23, 1984 21:03:52  file DSK:F4INST.SLO  --  of -- F41NNF
C00685 00071    SLOE  March 23, 1984 21:03:57  file DSK:F4INST.SLO  --  of -- F41NNF
C00700 00072    SLOE  March 23, 1984 21:04:06  file DSK:F4INST.SLO  --  of -- F41NNF
C00710 00073    SLOE  March 23, 1984 21:04:13  file DSK:F4INST.SLO  --  of -- F41NNF
C00728 00074    SLOE  March 23, 1984 21:04:23  file STRING:  --  of -- F41NNF
C00746 00075    SLOE  March 23, 1984 21:04:32  file STRING:  --  of -- F41NNF
C00759 00076    SLOE  March 23, 1984 21:04:39  file DSK:F4INST.SLO  --  of -- F41NNF
C00774 00077    SLOE  March 23, 1984 21:04:50  file STRING:  --  of -- F41NNF
C00790 00078    SLOE  March 23, 1984 21:04:58  file DSK:F4INST.SLO  --  of -- F41NNF
C00806 00079    SLOE  March 23, 1984 21:05:08  file STRING:  --  of -- F41NNF
C00821 00080    SLOE  March 23, 1984 21:05:16  file DSK:F4INST.SLO  --  of -- F41NNF
C00835 00081    SLOE  March 23, 1984 21:05:26  file DSK:F4INST.SLO  --  of -- F41NNF
C00851 00082    SLOE  March 23, 1984 21:05:39  file STRING:  --  of -- F41NNF
C00852 00083    SLOE  March 23, 1984 21:05:40  file DSK:F4INST.SLO  --  of -- F41NNF
C00854 00084    SLOE  March 23, 1984 21:05:41  file DSK:F4INST.SLO  --  of -- F41NNF
C00865 00085    SLOE  March 23, 1984 21:05:49  file DSK:F4INST.SLO  --  of -- F41NNF
C00882 00086    SLOE  March 23, 1984 21:05:58  file DSK:F4INST.SLO  --  of -- F41NNF
C00899 00087    SLOE  March 23, 1984 21:06:07  file STRING:  --  of -- F41NNF
C00916 00088    SLOE  March 23, 1984 21:06:13  file STRING:  --  of -- F41NNF
C00919 00089    SLOE  March 23, 1984 21:06:15  file DSK:F4INST.SLO  --  of -- F41NNF
C00936 00090    SLOE  March 23, 1984 21:06:25  file DSK:F4INST.SLO  --  of -- F41NNF
C00953 00091    SLOE  March 23, 1984 21:06:34  file DSK:F4INST.SLO  --  of -- F41NNF
C00966 00092    SLOE  March 23, 1984 21:06:41  file DSK:F4INST.SLO  --  of -- F41NNF
C00983 00093    SLOE  March 23, 1984 21:06:49  file DSK:F4INST.SLO  --  of -- F41NNF
C00986 00094    SLOE  March 23, 1984 21:06:49  file DSK:F4INST.SLO  --  of -- F41NNF
C00997 00095    SLOE  March 23, 1984 21:06:55  file DSK:F4INST.SLO  --  of -- F41NNF
C01006 00096    SLOE  March 23, 1984 21:06:59  file DSK:F4INST.SLO  --  of -- F41NNF
C01018 00097    SLOE  March 23, 1984 21:07:05  file DSK:F4INST.SLO  --  of -- F41NNF
C01035 00098    SLOE  March 23, 1984 21:07:15  file DSK:F4INST.SLO  --  of -- F41NNF
C01036 00099    SLOE  March 23, 1984 21:07:16  file DSK:F4INST.SLO  --  of -- F41NNF
C01052 00100    SLOE  March 23, 1984 21:07:25  file DSK:F4INST.SLO  --  of -- F41NNF
C01058 00101    SLOE  March 23, 1984 21:07:27  file DSK:F4INST.SLO  --  of -- F41NNF
C01066 0010Z    SLOE  March 23, 1984 21:07:33  file DSK:F4INST.SLO  --  of -- F41NNF
C01072 00103    SLOE  March 23, 1984 21:07:37  file DSK:F4INST.SLO  --  of -- F41NNF
C01082 00104    SLOE  March 23, 1984 21:07:43  file DSK:F4INST.SLO  --  of -- F41NNF
C01098 00105    SLOE  March 23, 1984 21:07:50  file DSK:F4INST.SLO  --  of -- F41NNF
C01110 00106    SLOE  March 23, 1984 21:07:55  file DSK:F4INST.SLO  --  of -- F41NNF
C01120 00107    SLOE  March 23, 1984 21:08:00  file DSK:F4INST.SLO  --  of -- F41NNF
C01135 00108    SLOE  March 23, 1984 21:08:10  file DSK:F4INST.SLO  --  of -- F41NNF
C01150 00109    SLOE  March 23, 1984 21:08:20  file DSK:F4INST.SLO  --  of -- F41NNF
C01165 00110    SLOE  March 23, 1984 21:08:29  file DSK:F4INST.SLO  --  of -- F41NNF
```

```
C01485 00137    SLOE  March 23, 1984 21:11:46  file DSK:DCA.SLO  --  of -- F41NNF
C01500 00138    SLOE  March 23, 1984 21:11:52  file DSK:DCA.SLO  --  of -- F41NNF
C01518 00139    SLOE  March 23, 1984 21:12:03  file DSK:DCA.SLO  --  of -- F41NNF
C01536 00140    SLOE  March 23, 1984 21:12:14  file DSK:DCA.SLO  --  of -- F41NNF
C01553 00141    SLOE  March 23, 1984 21:12:22  file STRING:  --  of -- F41NNF
C01570 00142    SLOE  March 23, 1984 21:12:32  file DSK:DCA.SLO  --  of -- F41NNF
C01576 00143    SLOE  March 23, 1984 21:12:34  file STRING:  --  of -- F41NNF
C01578 00144    SLOE  March 23, 1984 21:12:35  file DSK:VC.SLO  --  of -- F41NNF
C01593 00145    SLOE  March 23, 1984 21:12:39  file DSK:VC.SLO  --  of -- F41NNF
C01608 00146    SLOE  March 23, 1984 21:12:46  file DSK:VC.SLO  --  of -- F41NNF
C01624 00147    SLOE  March 23, 1984 21:12:58  file STRING:  --  of -- F41NNF
C01639 00148    SLOE  March 23, 1984 21:13:06  file DSK:VC.SLO  --  of -- F41NNF
C01655 00149    SLOE  March 23, 1984 21:13:17  file STRING:  --  of -- F41NNF
C01670 00150    SLOE  March 23, 1984 21:13:29  file STRING:  --  of -- F41NNF
C01672 00151    SLOE  March 23, 1984 21:13:30  file DSK:DACDEB.SLO  --  of -- F41NNF
C01673 00152    SLOE  March 23, 1984 21:13:30  file DSK:DACDEB.SLO  --  of -- F41NNF
C01685 00153    SLOE  March 23, 1984 21:13:39  file STRING:  --  of -- F41NNF
C01687 00154    SLOE  March 23, 1984 21:13:39  file DSK:PAN.SLO  --  of -- F41NNF
C01702 00155    SLOE  March 23, 1984 21:13:45  file DSK:PAN.SLO  --  of -- F41NNF
C01718 00156    SLOE  March 23, 1984 21:13:54  file DSK:PAN.SLO  --  of -- F41NNF
C01727 00157    SLOE  March 23, 1984 21:13:58  file STRING:  --  of -- F41NNF
C01728 00158    SLOE  March 23, 1984 21:13:58  file DSK:TMPGRN.SLO  --  of -- F41NNF
C01744 00159    SLOE  March 23, 1984 21:14:07  file DSK:TMPGRN.SLO  --  of -- F41NNF
C01751 00160    SLOE  March 23, 1984 21:14:11  file STRING:  --  of -- F41NNF
C01752 00161    SLOE  March 23, 1984 21:14:11  file DSK:AUDSWX.SLO  --  of -- F41NNF
C01767 00162    SLOE  March 23, 1984 21:14:19  file DSK:AUDSWX.SLO  --  of -- F41NNF
C01780 00163    SLOE  March 23, 1984 21:14:27  file STRING:  --  of -- F41NNF
C01781 00164    SLOE  March 23, 1984 21:14:27  file DSK:F4IO.SLO  --  of -- F41NNF
C01796 00165    SLOE  March 23, 1984 21:14:37  file DSK:F4IO.SLO  --  of -- F41NNF
C01799 00166    SLOE  March 23, 1984 21:14:38  file DSK:F4IO.SLO  --  of -- F41NNF
C01812 00167    SLOE  March 23, 1984 21:14:41  file DSK:F4IO.SLO  --  of -- F41NNF
C01828 00168    SLOE  March 23, 1984 21:14:51  file DSK:F4IO.SLO  --  of -- F41NNF
C01845 00169    SLOE  March 23, 1984 21:15:02  file STRING:  --  of -- F41NNF
C01862 00170    SLOE  March 23, 1984 21:15:11  file DSK:F4IO.SLO  --  of -- F41NNF
C01879 00171    SLOE  March 23, 1984 21:15:22  file DSK:F4IO.SLO  --  of -- F41NNF
C01896 00172    SLOE  March 23, 1984 21:15:30  file STRING:  --  of -- F41NNF
C01912 00173    SLOE  March 23, 1984 21:15:39  file STRING:  --  of -- F41NNF
C01928 00174    SLOE  March 23, 1984 21:15:47  file DSK:F4IO.SLO  --  of -- F41NNF
C01944 00175    SLOE  March 23, 1984 21:15:53  file DSK:F4IO.SLO  --  of -- F41NNF
C01949 00176    SLOE  March 23, 1984 21:15:55  file DSK:F4IO.SLO  --  of -- F41NNF
C01962 00177    SLOE  March 23, 1984 21:16:07  file DSK:F4IO.SLO  --  of -- F41NNF
C01977 00178    SLOE  March 23, 1984 21:16:14  file DSK:F4IO.SLO  --  of -- F41NNF
C01993 00179    SLOE  March 23, 1984 21:16:23  file STRING:  --  of -- F41NNF
C02009 00180    SLOE  March 23, 1984 21:16:35  file STRING:  --  of -- F41NNF
C02025 00181    SLOE  March 23, 1984 21:16:43  file STRING:  --  of -- F41NNF
C02041 00182    SLOE  March 23, 1984 21:16:52  file STRING:  --  of -- F41NNF
C02056 00183    SLOE  March 23, 1984 21:16:58  file STRING:  --  of -- F41NNF
C02060 00184    SLOE  March 23, 1984 21:17:00  file DSK:F4IO.SLO  --  of -- F41NNF
C02077 00185    SLOE  March 23, 1984 21:17:06  file DSK:F4IO.SLO  --  of -- F41NNF
C02094 00186    SLOE  March 23, 1984 21:17:13  file STRING:  --  of -- F41NNF
C02099 00187    SLOE  March 23, 1984 21:17:15  file DSK:F4IO.SLO  --  of -- F41NNF
C02114 00188    SLOE  March 23, 1984 21:17:20  file DSK:F4IO.SLO  --  of -- F41NNF
C02132 00189    SLOE  March 23, 1984 21:17:30  file DSK:F4IO.SLO  --  of -- F41NNF
C02146 00190    SLOE  March 23, 1984 21:17:37  file STRING:  --  of -- F41NNF
C02163 00191    SLOE  March 23, 1984 21:17:47  file STRING:  --  of -- F41NNF
C02171 00192    SLOE  March 23, 1984 21:17:52  file DSK:LPTX.SLO  --  of -- F41NNF
C02186 00193    SLOE  March 23, 1984 21:17:57  file STRING:  --  of -- F41NNF
C02203 00194    SLOE  March 23, 1984 21:18:04  file DSK:LPTX.SLO  --  of -- F41NNF
C02221 00195    SLOE  March 23, 1984 21:18:12  file STRING:  --  of -- F41NNF
C02238 00196    SLOE  March 23, 1984 21:18:19  file DSK:LPTX.SLO  --  of -- F41NNF
C02250 00197    SLOE  March 23, 1984 21:18:22  file STRING:  --  of -- F41NNF
C02251 00198    SLOE  March 23, 1984 21:18:22  file DSK:F4IO.SLO  --  of -- F41NNF
C02267 00199    SLOE  March 23, 1984 21:18:28  file DSK:F4IO.SLO  --  of -- F41NNF
C02271 00200    SLOE  March 23, 1984 21:18:29  file DSK:F4IO.SLO  --  of -- F41NNF
C02283 00201    SLOE  March 23, 1984 21:18:35  file DSK:F4IO.SLO  --  of -- F41NNF
C02299 00202    SLOE  March 23, 1984 21:18:50  file STRING:  --  of -- F41NNF
C02315 00203    SLOE  March 23, 1984 21:19:00  file STRING:  --  of -- F41NNF
C02329 00204    SLOE  March 23, 1984 21:19:08  file DSK:F4IO.SLO  --  of -- F41NNF
C02343 00205    SLOE  March 23, 1984 21:19:13  file DSK:F4IO.SLO  --  of -- F41NNF
C02359 00206    SLOE  March 23, 1984 21:19:18  file DSK:F4IO.SLO  --  of -- F41NNF
C02376 00207    SLOE  March 23, 1984 21:19:23  file DSK:F4IO.SLO  --  of -- F41NNF
C02383 00208    SLOE  March 23, 1984 21:19:25  file DSK:F4IO.SLO  --  of -- F41NNF
C02386 00209    SLOE  March 23, 1984 21:19:26  file DSK:F4IO.SLO  --  of -- F41NNF
C02401 00210    SLOE  March 23, 1984 21:19:31  file DSK:F4IO.SLO  --  of -- F41NNF
C02417 00211    SLOE  March 23, 1984 21:19:38  file DSK:F4IO.SLO  --  of -- F41NNF
C02433 00212    SLOE  March 23, 1984 21:19:45  file STRING:  --  of -- F41NNF
C02435 00213    SLOE  March 23, 1984 21:19:46  file DSK:F4IO.SLO  --  of -- F41NNF
C02436 00214    SLOE  March 23, 1984 21:19:46  file DSK:F41BS.SLO  --  of -- F41NNF
C02495 00215    SLOE  March 23, 1984 21:19:55  file DSK:F41NNF.SLO  --  of -- F41NNF
C02496 ENDMK
C•;
```

```
01 0001         COMMENT !   VALID 00002 PAGES
01 0002         C REC  PAGE   DESCRIPTION
01 0003         C00001 00001
01 0004         C00002 00002    .INSERT F41NN
01 0005         C00003 ENDMK
01 0006         C!;
01 0007
```

```
02 0008         .INSERT F41NN
```

```
01 0001 0       OSP = 0
01 0002 0       OSQ = 0
01 0003 0       FAST = 0
```

```
02 0008 1       FAST = 1
02 0009         .INSERT F41AX
```

```

      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

      XLIST
                                                    LIST

    1 WAITS = 1         ;Special WAITS map stuff

    1 STANSW = 1        ;Special CCRMA I/O devices

    0 OLDBLT = 0

    0 OLDECC = 0

    1 FZAROV = 1        ;KA style overflow interrupts

    0 TYMNET = 0

    1 TIMER = 1

    1 TAPE = 1          ;INCLUDE TAPE CODE

    1 DSK = 1 ;INCLUDE DISK CODE

    0 VID = 0  ;Exclude Foovision code

    0 IMP = 0  ;Dream on...

    1 DCA = 1
    20          DCA-UDEV = DLSDEV
    0  olddls = 0
    1  DLS4 = 1

    1 LPT = 1           ;Include LPT microcode
    15          LPT-UDEV = 15

    1 VC = 1            ;Versatec
    14          VCDEV = 14

    30          PAN-UDEV        = 30    ;uDevice address
    30          GRN-UDEV        = 30    ;uDevice address
    30          AS-UDEV         = 30    ;uDevice address
    3 PAN-INT-ENB     = 3     ;Interrupt enable

    0 JSYSDF = 0        ;SUPPRESS JSYS DEVICE

    0 AUGGYF = 0        ;SUPPRESS AUGMENT INSTRUCTIONS

    1 DBOOTF = 1        ;GET DISK BOOTER

    31          DIAGDEV = 31    ;ENABLE I/O DIAGNOSTIC MICROCODE FOR THIS DEVICE IF
                                                     NON-ZERO

    1 WAITSFIX = WAITS          ;FIXES FROM CCRMA NOT YET ACCEPTED INTO MAIN
                                                     UCODE

    10000       HILOC = 10000
                                                    :HILOC
                                                     .use[HILOC]
                                                     [ xlist
                                                       list ]
                                                    .insert F4APR
```

```
01 0001
01 0002
01 0003                                                 ;-----------------------------------------------------------------
01 0003                                                 ------
01 0004                                                 ;
01 0005                                                 ;       APR and PI instructions
01 0006                                                 ;
01 0007                                                 ;-----------------------------------------------------------------
01 0007                                                 ------
01 0008
01 0009
01 0010                                                 ;NOTE:  APR AMEM2  = STATUS
01 0011                                                 ;           AMEM17 = LIGHTS
01 0012                                                 ;           AMEM16 = DATA SWITCHES
01 0013                                                 ;           AMEM15 = CONTROL SWITCHES
01 0014
01 0015         42      APRSTS = 40 + APR-STATUS        ;Firmware status bits for APR are in
01 0015                                                  AMEM[2]
01 0016         44      PISTS  = 40 + PI-STATUS         ;DIDTO FOR PI AMEM[4]
01 0017
01m0018                                                  .opcode[700]    [xlist
01 0018                                                    list  ];APR & PI
01 0019 07600 010731170040000001400046025571457000          D[CONST 0] DEST[DEV-ADR] CLR-DEV-FROM-INTR JUMP[APIOT] NORM $
01 0020 07601 010731170040000001400046025571457000          D[CONST 0] DEST[DEV-ADR] CLR-DEV-FROM-INTR JUMP[APIOT] NORM $
01 0021
01m0022                                                  .reloc
01m0022                                                  [.USE[HILOC]
01m0022                                                  [ xlist
01 0023                                                    list ]                                ]
01 0024
01 0025                                                 ;;;APR & PI IOT DISP TABLE
01 0026
01 0027
01 0028 10000 540731170000060554161623254204160000     APRDSP: ILGIOT $                                                ; BLKI
01 0028                                                  APR,
01 0029 10001 010731170000060554161623654314160000             CONT $
01 0030
01 0031 10002 010731310000000014161561267304560000             D[D%AMEM16] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<Z0] LBJUMP[SEQI]
01 0031                                                  ]$                                  ;DATAI APR,
01 0032                                                          ;-- RD SW
01 0033 10003 010731170000060554161623654314160000             CONT $
01 0034
01 0035                                                 .REPEAT 1 - WAITS [
01 0036                                                                                     ;BLKO IS ECC FAST CHECK
01 0037                                                 D[APRSTS] ROT[24.] MASK[2.] COND[-ZERO] JUMP[APRBKO] C550 $
01 0038                                                         ;JUMP ON SOFT ECC OR NXM
01 0039                                                 D[PISTS] ROT[20.] MASK[1.] COND[ZERO] LBJUMP[APRBKO] C550 $
01 0040                                                         ;LBJUMP ON NOT HARD ECC
01 0041
01 0042                                                                                     ;DATAO FAST CHECK OF 60HZ CLOCK
01 0042                                                  FLAG
01 0043                                                 CLR-INT PUSHJ[CLKRDFLG] NORM $
01 0044                                                  ;Get CLOCK FLAG in AR 35, rest of AR = 0
01 0045                                                 D[AR] CONDSKP[ALU=0] $
01 0046                                                 ];.REPEAT 1 - WAITS
01 0047                                                 .REPEAT WAITS [
01 0048                                                 ILGIOT $ CONT $         ;BLKO
01 0049                                                 ILGIOT $ CONT $         ;DATAO
01m0050                                                 ];[
01m0050 10004 540731170000060554161623254204160000     ILGIOT $ CONT $         ;BLKO
01m0050 10005 010731170000060554161623654314160000
01m0050 10006 540731170000060554161623254204160000     ILGIOT $ CONT $         ;DATAO
01m0050 10007 010731170000060554161623654314160000
01 0050                                                 ].REPEAT WAITS
01 0051
01 0052 10010 010730170000060554027623657714160000             D[IR] MASK[13] DEST[Q] NORM $                           ;CONO
01 0053 10011 010650170000000001267620255714560000             D[CONST 33] ROT[5] ALU[-D&Q] DEST[Q] JUMP[APRCO] NORM $
01 0054
01 0055 10012 010731170000000014161622254314560000             PUSHJ[APRCI] NORM $                                     ;CONI
01 0056                                                           ; -- GET BITS IN AR & Q
01 0057 10013 010731310000000014045561254304560000             D[AR] MASK[18.] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<Z0]
01 0057                                                  LBJUMP[SEQI] ]$
01 0058
01 0059 10014 010731170000000014161622254314560000             PUSHJ[APRCI] NORM $                                     ; CONS2
01 0059
01 0060                                                           ;-- GET BITS IN AR & Q
01 0061 10015 010730170000000014045620257714560000             D[IR] MASK[18.] DEST[Q] JUMP[CTYCZ] NORM $
01 0062
01 0063 10016 010731170000000014161622254314560000             PUSHJ[APRCI] NORM $                                     ; CONSO
01 0063                                                  APR,
01 0064                                                           ; -- GET BITS IN AR & Q
01 0065 10017 010730170000000014045620257714560000             D[IR] MASK[18.] DEST[Q] JUMP[CTYCS] NORM $
01 0066
01 0067                                                 PIDSP:  .repeat 3[ ILGIOT $
01 0068                                                         CONT $
01m0069                                                 ]
01m0069 10020 540731170000060554161623254204160000     [ ILGIOT $
01m0069 10021 010731170000060554161623654314160000             CONT $
01m0069 10022 540731170000060554161623254204160000     ][ ILGIOT $
01m0069 10023 010731170000060554161623654314160000             CONT $
01m0069 10024 540731170000060554161623254204160000     ][ ILGIOT $
01m0069 10025 010731170000060554161623654314160000             CONT $
01 0069 10026 010331170000060554161503454314216000     ]       DFRQ ALU[MEMAC] DEST[HOLD] $                            ;DATAO
01 0069                                                  PI,
01 0070                                                                 ;-- Set lights
01 0071 10027 540731170000060554161636254604160000             D[MEM] DEST[D%AMEM17] NEOI $
01 0072
01 0073 10030 010731170000000014161620254314560000             JUMP[PICONO] NORM $                                     ;CONO PI,
01 0073
01 0074 10031 010731170000060554161623654314160000             CONT $
01 0075
01 0076 10032 010731170000000014161622254314560000             PUSHJ[PICONISUB] NORM $                                 ; CONI
01 0076                                                  PI,
01 0077                                                                 ;-- Get bits in AR & Q
01 0078 10033 010731310000000014161561254304560000             D[AR] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<Z0] LBJUMP[SEQI] ]$
01 0079
01 0080 10034 010731170000000014161622254314560000             PUSHJ[PICONISUB] NORM $; CONS2 -- GET BITS IN AR & Q
01 0081 10035 010730170000000014045620257714560000             D[IR] MASK[18.] DEST[Q] JUMP[CTYCZ] NORM $
01 0082
01 0083 10036 010731170000000014161622254314560000             PUSHJ[PICONISUB] NORM $; CONSO -- GET BITS IN AR & Q
```

```
01 0089                                                 .REPEAT 1 - WAITS [
01 0090
01 0091                                                 .PAIR
01 0092                                                 APRBKO: CONDSKP[NEVER] $
01 0093                                                         ;HERE WE NO SKIP SINCE ECC CONDITION IS TRUE
01 0094                                                 CONDSKP[ALWAYS] $
01 0095                                                         ;HERE WE SKIP SINCE NO ECC CONTITION IS TRUE
01 0096
01 0097
01 0098                                                 ];.REPEAT 1 - WAITS
01 0099
01 0100
01 0101 10040 010651370000006054060362365577416000     APRCO:  D[CONST 1] ROT[3] ALU[-D&Q] DEST[AR] SHORT $
01 0102                                                         ;AR HAS ENABLES
01 0103 10041 010770170040060550761623657714160000             D[IR] ROT[36. - 1.] ALU[NOTD] DEST[Q] CLR-DEV-FROM-INTR NORM $
01 0104 10042 010631171777777373337503655531416000             D[NLIT 777777575557] ALU[DORQ] DEST[HOLD] NORM $
01 0105                                                         ;GET NOT OF CLEARS THAT MUST BE SHIFTED RIGHT ONCE IN
                                                        HOLD
01 0106 10043 010770170000060554161623657714160000             D[IR] ALU[NOTD] DEST[Q] NORM $
01 0107 10044 010630171777777615777623655531416000             D[NLIT 7777777067777] ALU[DORQ] DEST[Q] NORM $
01 0108                                                         ;GET NOT OF CLEARS THAT DON'T NEED TO SHIFT IN Q
01 0109 10045 010640170000006055416162365471516000             D[MEM] ALU[D&Q] DEST[Q] NORM $
01 0110                                                         ;COMBINE THEM INTO Q
01 0111                                                         ;REALY DOING -(XorY) BUT THAT IS SAME AS -Xand-Y
01 0112 10046 010640170000006055416162366131416000             D[APRSTS]  ALU[D&Q] DEST[Q] $
01 0113                                                         ; Clear indicated bits.
01 0114 10047 010631370000006055416104365431416000             D[AR] ALU[DORQ]  DEST[APRSTS AR] NORM $
01 0115                                                         ; Set indicated bits, PI chan
01 0116 10050 010730170000060554161623646114160000             D[PC-FLAGS] DEST[Q] C600 $
01 0117 10051 010731002000000010203620257614560000             D[IR] ROT[41] MASK[1] COND[OBUS=0] JUMP[APRCO2] C550 $
01 0118                                                         ;Jump if no CLR OVERFLOW bit.
01 0119 10052 010650170000006055060247365571417000             D[CONST 1] ROT[35.] ALU[-D&Q] DEST[Q PC-FLAGS]  $
01 0120                                                         ; Remove overflow bit from current PC flags.
01 0121 10053 010731002000000000740362025761456000     APRCO2: D[IR] ROT[36] MASK[1] COND[OBUS=0] JUMP[APRCO3] C550 $
01 0122                                                         ; Jump if no CLR FLT OV.
01 0123 10054 010651170000006055000247365571417000             D[CONST 1] ROT[35. - 3] ALU[-D&Q] DEST[PC-FLAGS] $
01 0124                                                         ; Remove bit from current PC flags
01 0125 10055 010731000040000006603622257614560000     APRCO3: D[IR] ROT[33] MASK[1] COND[-OBUS=0] CLR-INT PUSHJ[CLKCLR] C550 $
01 0125
01 0126                                                         ; Clear Clock Flag if indicated.
01 0127 10056 010731170040000014161622254314560000             PUSHJ[APRCII] CLR-DEV-FROM-INTR NORM $
01 0128                                                         ; Get APR Coni bits.
01 0129 10057 010730170000060554161623666314160000             D[D%AMEM0 + APR-MODE] DEST[Q] $
01 0130                                                         ;Get copy of current mode bits.
01 0131 10060 010731002000000014007620254214560000             D[AR] MASK[3] COND[OBUS=0] JUMP[APRC1] C550 $
01 0132                                                         ; Jump if no PI channel.  We don't want to enable
                                                        arithmetic
01 0133                                                         ; micro-interrupts in that case.
01 0134 10061 010731006000000000776162025421456000             D[AR] ROT[31.] COND[SIGNOFF] C550 JUMP[APRC1] $
01 0135                                                         ; J IF NO OV INT ENBL
01 0136                                                         ; *** Don't both enables need to be checked???
01 0137 10062 010630170000000014041300255714560000             D[CONST 20] ALU[DORQ] DEST[Q ( D%AMEM0 + APR-MODE )] JUMP[APRC2]
                                                        $
01 0138                                                         ; Turn on arithmetic interrupts.
01 0139 10063 010650170000000014041300255714560000     APRC1:  D[CONST 20] ALU[-D&Q] DEST[Q ( D%AMEM0 + APR-MODE )] JUMP[APRC2]
                                                        $
01 0140                                                         ; Turn off arithmetic interrupts
01 0141 10064 010731170000060540160223666114170000     APRC2:  D[D%AMEM0 + APR-MODE] ROT[CPU-MODE-ROT] DEST[CPU-MODE] LONG $
01 0142                                                         ;Output new mode bits.
01 0143 10065 010730170042000014161620254314560000             CLR-AROV-TRAP-FLAG D[AR] DEST[Q] JUMP[APRCK1] $
01 0144                                                         ;Go check for APR interrupts.
01 0145
01 0146                                                   ;Check for clock interrupts (and other things)
01 0147 10066 010731170000000014161622254314560000     APRCHK: PUSHJ[APRCI] NORM $; GET CONI BITS IN AR & Q
01 0148                                                 APRCK1:
01 0149                                                  .repeat 1 - WAITS [
01 0150                                                         D[AR] MASK[3] COND[OBUS=0] JUMP[MAIN] C550 $
01 0151                                                         ; Done if PI=0
01 0152                                                         D[AR] ROT[24.] MASK[2] COND[-ZERO] JUMP[APDINT] C550 $
01 0153                                                         ; J IF SOFT ECC OR NXM INT
01 0154                                                         D[AMEM-ABS PI-STATUS] ALU[NOTD] DEST[ALU1 Q] NORM $
01 0155                                                         ;USE ALU1 Q FOR PI BITS SO DONT HURT APR BITS IN Q
01 0156                                                         ;GET NOT OF APR FLAGS KEPT IN PI CONI WORD
01 0157                                                         D[CONST 3] ROT[35. - 20.] ALU[ALU1 D&Q] COND[ZERO] JUMP[APDINT]
01 0157                                                  C550 $
01 0158                                                         ;JUMP IF PAR ERR INTS ENABLED AND ON
01 0159                                                         D[AR] ROT[20.] MASK[1] COND[-OBUS=0] JUMP[APDINT] C550 $
01 0160                                                         ; J if PDLOV int
01 0161                                                         D[AR] ROT[43] ALU[D&Q] DEST[Q] SHORT $
01 0162                                                         ; AND mask with flag
01 0163                                                 .REPEAT FZAROV  [
01 0164                                                 ;I think this is the right thing.  It was commented out after translation
01 0165
01 0166                                                 ;from F2.       TVR/Oct82
01 0167                                                         D[CONST 11] ROT[3] ALU[D&Q] COND[-OBUS=0] JUMP[APDINT] C550 $
01 0168                                                         ; J if OV int
01 0169                                                 ];.REPEAT F2AROV
01 0170                                                 ;;;**** Is this the same code as French used?
01 0171                                                 ;;;**** If so, shouldn't there be an enable for soft interrupt?
01 0171                                                 ;;;     D[CONST 31] ROT[35. - 26.] ALU[D&Q] COND[OBUS=0] JUMP[MAIN] C550
                                                         $
01 0172                                                 ;;;             ; Done if no CLOCK int, soft ECC interrupt or NXM
01 0173                                                         D[CONST 10] ROT[6] ALU[D&Q] COND[OBUS=0] JUMP[MAIN] C550 $
01 0174                                                         ; Done if no CLOCK int
01 0175                                                     ;  \ /
01 0176                                                   ];.repeat 1 - WAITS
01 0177                                                  .repeat WAITS [
01 0178                                                         D[AR] ROT[27.] COND[OBUS<0] JUMP[MAIN] C550 $
01 0179                                                         ;If no APR interrupt is not requested, don't interrupt.
01 0180                                                         ;Otherwise, fall thru
01m0181                                                   ];[
01m0181 10067 010731004000000006761620254214560000             D[AR] ROT[27.] COND[OBUS<0] JUMP[MAIN] C550 $
01m0181                                                         ;If no APR interrupt is not requested, don't interrupt.
01m0181                                                         ;Otherwise, fall thru
01 0181                                                   ].repeat WAITS
01 0182
01 0183                                                 ;       \ /     (Falls thru from APRCK1)
01 0184 10070 010730370000000014007620254314560000     APDINT: D[AR] MASK[3] DEST[Q AR] JUMP[PIGEN] NORM $
01 0185                                                           ; Get chan #, cause interrupt
01 0186
```

```
01 0192                                                         D[AR] ROT[R2] ALU[DORQ] DEST[AR] SHORT $
01 0193                                                       ]
01m0194                                                         APRGBT[1 6]
01m0194
01m0194 10072 010730170000060540203623646114160000           [   D[PC-FLAGS] ROT[1] MASK[1] DEST[Q] C600 $
01m0194 10073 010631370000060541561623654374160000               D[AR] ROT[6] ALU[DORQ] DEST[AR] SHORT $
01m0195                                                        ]  APRGBT[4 41]
01m0195
01m0195 10074 010730170000060541003623646114160000           [   D[PC-FLAGS] ROT[4] MASK[1] DEST[Q] C600 $
01m0195 10075 010631370000060550361623654374160000               D[AR] ROT[41] ALU[DORQ] DEST[AR] SHORT $
01m0196                                                        ]  APRGBT[7 33]
01m0196
01m0196 10076 010730170000060541603623646114160000           [   D[PC-FLAGS] ROT[7] MASK[1] DEST[Q] C600 $
01m0196 10077 010631370000060546761623654374160000               D[AR] ROT[33] ALU[DORQ] DEST[AR] SHORT $
01 0197 10100 010730170000060543761623654374160000            ]  D[AR] ROT[17] DEST[Q] SHORT $; SET FLAGS
01 0198 10101 010241170040060554160463654314170000               ALU[0] DEST[DEV-ADR] CLR-DEV-FROM-INTR NORM $
01 0199                                                     .repeat 1 - WAITS [
01 0200                                                         D[APRSTS]  ALU[DORQ] DEST[Q AR] NORM POPJ $
01 0201                                                     ];.repeat 1 - WAITS
01 0202
01 0203                                                     .repeat WAITS [
01 0204                                                     ;SAIL's KA-10 has a processor modification which sets bit 27 (400 bit) if
                                                             the
01 0205                                                     ;APR is NOT interrupting.  This feature is used in a CONSZ chain and
                                                             would
01 0205                                                     ;consume many instructions to simulate in a critical path in the system.
01 0206
01 0206
01 0207                                                             D[APRSTS] ALU[DORQ] DEST[Q AR] NORM $
01 0208                                                                     ;Construct full status
01 0209                                                             D[AR] MASK[3] COND[OBUS=0] JUMP[APRCIN] NORM $
01 0210                                                                     ;Set 400 bit if interrupts disabled
01 0211                                                                     ;(This may not be the same as SAIL)
01 0212                                                     ;;      D[AR] ROT[19.] MASK[1] COND[-OBUS=0] POPJ C550 $
01 0213                                                     ;;              ;If pushdown overflow, we will interrupt.
01 0214                                                             D[CONST 21] ROT[12.] ALU[D&Q] COND[-OBUS=0] POPJ C550 $
01 0215                                                                     ;If pushdown overflow, we will interrupt.
01 0216                                                             D[AR] ROT[43] ALU[D&Q] DEST[Q] SHORT $
01 0217                                                                     ;Combine flag with enable to see which interrupts
01 0218                                                             D[CONST 11] ROT[3] ALU[D&Q] COND[-OBUS=0] JUMP[APRCIY] C550 $
01 0219                                                                     ;Jump if overflow is causing an interrupt.
01 0220                                                     ::;;*** Shouldn't there be an enable for soft ECC interrupts?
01 0221                                                             D[CONST 15] ROT[9.] ALU[D&Q] COND[-OBUS=0] JUMP[APRCIY] C550 $
01 0222                                                                     ;Jump if clock or soft ECC is causing an interrupt.
01 0223                                                     APRCIN: D[CONST 4] ROT[6] DEST[Q] SHORT $
01 0224                                                                     ;Not interrupting, set 400 bit in APR status
01 0225                                                     APRCIY: D[AR] ALU[DORQ] DEST[AR Q] POPJ NORM $
01 0226                                                                     ;Turn on 400 bit if needed.
01 0227                                                                     ;Note, that since we ANDed something with Q, ORing in Q
01 0227                                                      is
01 0228                                                                     ;a NO-OP if we didn't go thru APRCIN.
01m0229                                                     ];[
01m0229                                                     ;SAIL's KA-10 has a processor modification which sets bit 27 (400 bit) if
01m0229                                                      the
01m0229                                                     ;APR is NOT interrupting.  This feature is used in a CONSZ chain and
01m0229                                                      would
01m0229                                                     ;consume many instructions to simulate in a critical path in the system.
01m0229
01m0229 10102 010630370000060554161623661314160000                 D[APRSTS] ALU[DORQ] DEST[Q AR] NORM $
01m0229                                                                     ;Construct full status
01m0229 10103 010731002000000001400762025431456000                 D[AR] MASK[3] COND[OBUS=0] JUMP[APRCIN] NORM $
01m0229                                                                     ;Set 400 bit if interrupts disabled
01m0229                                                                     ;(This may not be the same as SAIL)
01m0229                                                     ;;      D[AR] ROT[19.] MASK[1] COND[-OBUS=0] POPJ C550 $
01m0229                                                     ;;              ;If pushdown overflow, we will interrupt.
01m0229 10104 010641000000060543043624255614160000                 D[CONST 21] ROT[12.] ALU[D&Q] COND[-OBUS=0] POPJ C550 $
01m0229                                                                     ;If pushdown overflow, we will interrupt.
01m0229 10105 010640170000060550761623654374160000                 D[AR] ROT[43] ALU[D&Q] DEST[Q] SHORT $
01m0229                                                                     ;Combine flag with enable to see which interrupts
01m0229 10106 010641000000000000623620255614560000                 D[CONST 11] ROT[3] ALU[D&Q] COND[-OBUS=0] JUMP[APRCIY] C550 $
01m0229                                                                     ;Jump if overflow is causing an interrupt.
01m0229                                                     ::;;*** Shouldn't there be an enable for soft ECC interrupts?
?ERROR - set location with : must be followed by a valid expression
01m0229 10107 010641000000000002233620255614560000                 D[CONST 15] ROT[9.] ALU[D&Q] COND[-OBUS=0] JUMP[APRCIY] C550 $
01m0229                                                                     ;Jump if clock or soft ECC is causing an interrupt.
01m0229 10110 010730170000060541411623655774160000         APRCIN: D[CONST 4] ROT[6] DEST[Q] SHORT $
01m0229                                                                     ;Not interrupting, set 400 bit in APR status
01m0229 10111 010630370000060554161624254314160000         APRCIY: D[AR] ALU[DORQ] DEST[AR Q] POPJ NORM $
01m0229                                                                     ;Turn on 400 bit if needed.
01m0229                                                                     ;Note, that since we ANDed something with Q, ORing in Q
01m0229                                                      is
01m0229                                                                     ;a NO-OP if we didn't go thru APRCIN.
01 0229                                                     ].repeat WAITS
01 0230
01 0231
01m0232                                                     APIOT:  UIOTRP[MUUO]
01m0232                                                      [      COND[USER] JUMP[MUUO] NORM
01 0232 10112 010731124000000001416162025431456000             ]$                  ;Trap if not IOT-USER Mode
01m0233                                                             IOTDIS
01m0233
01m0233 10113 010730170040060543211623657714160000          [      CLR-DEV-FROM-INTR D[IR] ROT[13.] MASK[4] DEST[Q] NORM $
01m0233 10114 010600170000060543211623657774160000                 D[IR] ROT[13.] MASK[4] ALU[D+Q] DEST[Q] SHORT $
01m0233 10115 010601370000060544445623660314160000                 D[D%AMEM0] ROT[18.] MASK[18.] ALU[D+Q] DEST[AR] NORM $
01m0233 10116 010731170000000000160007254114560000                 D[AR] ROT[MUA-ROT] ODISP LONG $
01 0234                                                          ]
01 0235 10117 010241170000060554161043654314160000         APRRST: ALU[0] DEST[APRSTS] $
01 0236 10120 010241170000000314161543654314160000                 ALU[0] DEST[AMEM-ABS APR-MODE] $
01 0237 10121 010730170020000000001623655314160000                 D[LIT APRDSP * 1000000] DEST[Q] NORM $
01 0238 10122 010631170000000000001004255314160000                 D[LABEL ILLINT] ALU[DORQ] DEST[D%AMEM0] POPJ $
01 0239
01 0240
```

```
02 0241                                                                 ;-----------------------------------------------------------------------
02 0242                                                                 ------
02 0243                                                                 ;
02 0244                                                                 ;       PI - Priority Interrupt Service         Macro Device 4
02 0245                                                                 ;
02 0246                                                                 ;-----------------------------------------------------------------------
02 0246                                                                 ------
02 0247
02 0248                                                                 ;The PI system is entirely emulated- there is no hardware for it.
02 0249
02 0250                                                                 ;PI SYSTEM USE OF APR AMEM---
02 0251                                                                 ; 4     MEM PAR ERR(BIT 19), MEM PAR ERR INTRPT ENB(BIT 20),
02 0252                                                                 ;        PI SYSTEM ON(BIT 28) CHN1-7 ON (BITS 29-35)
02 0253                                                                 ; 5     WAITING RQ 1-7 (11-18)  IN PROG 1-7 (29-35)
02 0254                                                                 ; 6     RQ COUNTS - 4-BIT FIELDS, CHN. 7 AT RIGHT-END OF WORD.
02 0255                                                                 ;
02 0256                                                                 ;Other uses of APR AMEM are documented at the beginning
02 0257
02 0258                                                                 PI-GET-CHN:     ;Make binary chn. no. from mask in AR, then call
02 0258                                                                  PIGETMASK.
02 0259 10123 010731370000060547161623654314160000                             D[AR] ROT[28.] DEST[AR] NORM $
02 0260                                                                                 ;Put RQ 1 into bit 1
02 0261 10124 710731170000060554015623655714160000                             D[CONST 6] DEST[MA] NORM $
02 0262                                                                                 ;MA will get 7-CHN (for use in shifting)
02 0263 10125 010731204000000000036162025421456000             PIL1:   D[AR] ROT[1] DEST[AR] C550 COND[OBUS<0] JUMP[PIGETMASK] $
02 0264                                                                                 ;Found first bit ?
02 0265 10126 710721170000202534161620256714560000                             D[MA] ALU[D-1] DEST[MA] NORM JUMP[PIL1] $
02 0266                                                                                 ;No. Decrement count and loop.
02 0267                                                                 PIGETMASK: ;Load AR29-35 with single bit corresponding to chan.
02 0268 10127 010731170000060554160443656714170000                             D[MA] DEST[ROTR] NORM $
02 0269                                                                                 ;Load rotate amount.
02 0270 10130 010731370000060552003624255714160000                             D[CONST 1] ROT[R] DEST[AR] NORM  POPJ $
02 0271                                                                                 ;Make mask of first bit only in AR.
02 0272
02 0273
02 0274                                                                 PI-CHECK-RQS:    ;SEE IF IT IS TIME TO TAKE AN INTRPT.
02 0275                                                                 .REPEAT STANSW [        ;Determine which PI channels should be enabled
02 0275                                                                  for PAN
02 0276                                                                         D[D%AMEM4] ROT[35. - 10.] DEST[Q] PUSHJ[PANIST] NORM $
02 0277                                                                                 ;Select only those channels turned on and ready to
02 0277                                                                  interrupt
02 0278                                                                         MAPF[PAN-INT-ENB] CYLEN[IOB-OUT] alu[0] dest[dev-adr] $
02 0279                                                                                 ;Finish updating enable state in PAN interface
02m0280                                                                 ];[     ;Determine which PI channels should be enabled for PAN
02m0280 10131 010730170000000000636162226231456000                             D[D%AMEM4] ROT[35. - 10.] DEST[Q] PUSHJ[PANIST] NORM $
02m0280                                                                                 ;Select only those channels turned on and ready to
02m0280                                                                  interrupt
02m0280 10132 010241170340060554160463654114170000                             MAPF[PAN-INT-ENB] CYLEN[IOB-OUT] alu[0] dest[dev-adr] $
02m0280                                                                                 ;Finish updating enable state in PAN interface
02 0280                                                                 ].REPEAT STANSW
02 0281 10133 010731202000000000441762026261456000             D[D%AMEM5]  ROT[18.] MASK[7] DEST[AR]  C550 COND[OBUS=0]
02 0281                                                                  JUMP[MAIN] $
02 0282                                                                                 ;Any RQ'S ?
02 0283 10134 010241170040202474160462254314570000             ALU[0] DEST[DEV-ADR] CLR-DEV-FROM-INTR NORM PUSHJ[PI-GET-CHN] $
02 0284                                                                                 ;Get unary chn no. in AR, shift amt. in MA, ROTR
02 0285 10135 010731006000000007161620262214560000             D[D%AMEM4] ROT[34] C550 COND[-OBUS<0] JUMP[MAIN] $
02 0286                                                                                 ;Exit if PI sys not on.
02 0287 10136 010730170000060552017623647314160000             D[MASK 7] ROT[R] DEST[Q] NORM $
02 0288                                                                                 ;Mask of chn and all higher chns.
02 0289 10137 010641000000000001416162026261456000             D[D%AMEM5] ALU[D&Q] C550 COND[-OBUS=0] JUMP[MAIN] $
02 0290                                                                                 ;Exit if this or higher chn in progress.
02 0291 10140 010730170000060554161623654314160000             D[AR] DEST[Q] NORM $ ;MOVE UNARY CHN # TO Q.
02 0292 10141 010641002000000001416162026221456000             D[D%AMEM4] ALU[D&Q] C550 COND[OBUS=0] JUMP[MAIN] $
02 0293                                                                                 ;EXIT IF CHN NOT ON.
02 0294 10142 010731170000060540560443656714170000             D[MA] ROT[2] DEST[ROTR] NORM $
02 0295                                                                                 ;GET SHIFT AMT 4 TIMES LARGER, TO ACCESS CNT FIELD
02 0296 10143 010730170000060552003623655714160000             D[CONST 1] ROT[R] DEST[Q] NORM $
02 0297                                                                                 ;A ONE ALIGNED WITH RQ CNT FIELD FOR THIS CHN.
02 0298 10144 011620170000060554161503663314160000             D[D%AMEM6] ALU[D-Q] DEST[Q HOLD] NORM $
02 0299                                                                                 ;DECREMENT OUR WAITING RQ COUNT.
02 0300 10145 010660170000060554161623663314160000             D[D%AMEM5] ALU[D#Q] DEST[Q] NORM $ ;DID WE OVERFLOW ?
02 0301                                                                 .REPEAT 1 - WAITSFIX [
02 0302                                                                         D[CONST 20] ROT[R] ALU[D&Q] COND[-OBUS=0] C550 JUMP[.] $
02 0303                                                                 ];.REPEAT 1 - WAITSFIX
02 0304                                                                 .REPEAT WAITSFIX [
02 0305                                                                         D[CONST 20] ROT[R] ALU[D&Q] COND[-OBUS=0] C550 JUMP[PIERR] $
02 0306                                                                           ;LOOP HERE FOREVER IF WE OVERFLOWED 4-BIT CNT FIELD
02m0307                                                                 ];[
02m0307 10146 010641000000000001204162025561456000             D[CONST 20] ROT[R] ALU[D&Q] COND[-OBUS=0] C550 JUMP[PIERR] $
02m0307                                                                           ;LOOP HERE FOREVER IF WE OVERFLOWED 4-BIT CNT FIELD
02 0307                                                                 ].REPEAT WAITSFIX
02 0308 10147 010730170000060554161143654715160000             D[MEM] DEST[D%AMEM6 Q] NORM $
02 0309                                                                                 ;PUT BACK WORD OF RQ COUNTS.
02 0310                                                                         D[CONST 17] ROT[R] ALU[D&Q] C550
02 0311 10150 010641000000000012037620255614560000               COND[-OBUS=0] JUMP[PIINTGO] $ ;JUMP IF OUR COUNT NEQ 0
02 0312 10151 010770170000000004561620254314560000             D[AR] ROT[18.] ALU[NOTD] DEST[Q] NORM JUMP[PIL11] $
02 0313                                                                                 ;MASK FOR CLEARING THE WAITING RQ BIT.
02 0314 10152 010770170000060554001623655714160000             PIINTGO:D[CONST 0] ALU[NOTD] DEST[Q] NORM $
02 0315                                                                                 ;DON'T CLEAR THE BIT, RQ'S STILL WAITING
02 0316 10153 010640170000060554161623662714160000             PIL11:  D[D%AMEM5] ALU[D&Q] DEST[Q] NORM $
02 0317                                                                                 ;GET STATUS B, EITHER DO OR DO NOT CLEAR RQ BIT
02 0318 10154 010631170000060554161123654314160000             D[AR] ALU[D0RQ] DEST[D%AMEM5] NORM $
02 0319                                                                                 ;Set IN PROGRESS bit, store Status B.
02 0320 10155 010730370000060554161623646114160000             D[PC-FLAGS] DEST[Q AR] C600 $
02 0321                                                                                 ;Get PC into Q, AR.
02 0322 10156 010651170000060547402423655714170000             D[CONST 1] ROT[35. - 5] ALU[-D&Q] DEST[PC-FLAGS] NORM $
02 0323                                                                                 ;Clear User Mode.
02 0324 10157 010700170046060554135623655714160000             D[CONST 56] DEST[Q] SPEC[CLR-MAP-SR] NORM $
02 0325                                                                                 ;Prepare to calc. intrpt. address.
02 0326                                                                                 ;Make sure we don't fetch user trap locations!!
02 0327 10160 711611170000000000036162025671456000             D[MA] ROT[1] ALU[Q-D] DEST[MA] NORM JUMP[PIMUUO] $
02 0328                                                                                 ;Fetch intrpt. instr and go interpret it.
02 0329
```

```
03 0330
03 0331
03 0332 10161 010241170041006055416046365431417000
03 0333
03 0334
03 0335
03 0336
03 0337
03 0338
03m0339
03m0339 1016Z 0107310020000000141616202542145600O
03m0339
03 0339
03 0340 10163 010730170000060554161623662314160O0
03 0341 10164 010601170000060544403103655714160O0
03 0342 10165 010730170000060554017623655714160O0
03 0343
03 0344 10166 711611170000202574007622254314560O0
03 0345
03 0346 10167 010731006000000007161620262214560O0
03 0347
03 0348 10170 010730170000060552017623647314160O0
03 0349
03 0350 10171 010641000000000141616202626145600O
03 0351
03 0352 10172 010730170000060554161623654314160O0
03 0353 10173 010641000000203254161620262214560O0
03 0354
03 0355
03 0356 10174 010731170000060540560443656714170O0
03 0357
03 0358 10175 010730170000060552003623655714160O0
03 0359
03 0360 10176 010600170000060554161503663314160O0
03 0361
03 0362 10177 010641000000204032021620255614560O0
03 0363
03 0364 10200 010731170000060554161143654715160O0
03 0365
03 0366 10201 010730170000060554161623662714160O0
03 0367
03 0368 10202 010631170000000004561120254314560O0
03 0369
03 0370
03 0371
03 0372 10203 010241170041006055416046365431417000
03 0373
03 0374
03 0375
03 0376
03 0377
03 0377
03 0378
03 0379
03 0380
03 0381
03 0382
03 0382
03 0383
03 0384
03 0384
03 0385
03 0386
03 0387
03m0388
03m0388 10204 010730170000060554161623666314160O0
03m0388
03m0388 10205 010731002000000014007620261314560O0
03m0388
03m0388
03m0388
03m0388 10206 010731006000000007761620261314560O0
03m0388
03m0388
03m0388 10207 010630170000000014041300255714560O0
03m0388
03m0388
03m0388 10210 010650170000000014041300255714560O0
03m0388
03m0388
03m0388 10211 010731170000060540160223666114170O0
03m0388
03 0388
03 0389 10212 010731200000202474017622262614560O0
03 0389
03 0390
03 0390
03 0391 10213 010730170000060554161623662714160O0
03 0392 10214 010651170000202634161120254314560O0
03 0393
03 0394
03 0395
```

```
PIGEN:   ;ENTER WITH CHN IN AR TO REQUEST INTRPT.
        ALU[0] DEST[DEV-ADR] CLR-DEV-FROM-INTR NORM  $
.REPEAT 1 - WAITS [
        D[AR]  C550 COND[OBUS=0] JUMP[MAIN] $
];.REPEAT 1 - WAITS
.REPEAT WAITS [ ;If some device does this, i want to know!!  TVR/Oct80
        D[AR]  C550 COND[OBUS=0] JUMP[PIERR] $
                ;Should never do this!  To proceed, jump to MAIN
];[     ;If some device does this, i want to know!!  TVR/Oct80
        D[AR]  C550 COND[OBUS=0] JUMP[PIERR] $
                ;Should never do this!  To proceed, jump to MAIN
].REPEAT WAITS
        D[D%AMEM4] DEST[Q] NORM $
        D[CONST 1] ROT[18.] ALU[D+Q] DEST[D%AMEM4] NORM $
        D[CONST 7] DEST[Q] NORM $
                ;7-CHN is amt to shift by for mask bit.
        D[AR] MASK[3] ALU[Q-D] DEST[MA] NORM PUSHJ[PIGETMASK] $
                ;Load ROTR, form mask in AR.
PIGEN1: D[D%AMEM4] ROT[34] C550 COND[-OBUS<0] JUMP[PIGENWT] $
                ;BRANCH IF PI SYS NOT ON.
        D[MASK 7] ROT[R] DEST[Q] NORM $
                ;MASK OF CHN AND ALL HIGHER CHNS.
        D[D%AMEM5] ALU[D&Q] C550 COND[-OBUS=0] JUMP[PIGENWT] $
                ;BRANCH IF THIS OR HIGHER CHN IN PROGRESS.
        D[AR] DEST[Q] NORM $ ;MOVE UNARY CHN # TO Q.
        D[D%AMEM4] ALU[D&Q] C550 COND[-OBUS=0] JUMP[PIINTGO] $
                ;IF CHN ON, GO TAKE INTRPT.
PIGENWT:        ;INTRPT CANNOT HAPPEN NOW, SO SET A WAITING RQ.
        D[MA] ROT[2] DEST[ROTR] NORM $
                ;GET SHIFT AMT 4 TIMES LARGER, TO GET CNT FIELD
        D[CONST 1] ROT[R] DEST[Q] NORM $
                ;A ONE ALIGNED WITH RQ CNT FIELD FOR THIS CHN.
        D[D%AMEM6] ALU[D+Q] DEST[Q HOLD] NORM $
                ;INCREMENT OUR WAITING RQ COUNT.
        D[CONST 10] ROT[R] ALU[D&Q] COND[-OBUS=0] C550 JUMP[. + 2] $
          ;DON'T LET COUNT GET HIGHER THAN 7.
        D[MEM] DEST[D%AMEM5] NORM $
                ;PUT BACK WORD OF RQ COUNTS.
        D[D%AMEM5] DEST[Q] NORM $
                ;GET STATUS B.
        D[AR] ROT[18.] ALU[DORQ] DEST[D%AMEM5] NORM JUMP[MAIN] $
                ;Set WAITING RQ bit.

PI-DISMISS:
        CLR-DEV-FROM-INTR ALU[0] DEST[DEV-ADR] NORM $
.REPEAT FZAROV [
        D[D%AMEM0 + APR-MODE] DEST[Q] $
                ;Get copy of current mode bits.
        D[APRSTS] MASK[3] COND[OBUS=0] JUMP[PIDSM1] $
                ; Jump if no PI channel.  We don't want to enable
 arithmetic
                ; micro-interrupts in that case.
        D[APRSTS] ROT[31.] COND[SIGNOFF] JUMP[PIDSM1] $
                ; J IF NO OV INT ENBL
                ; *** Don't both enables need to be checked???
        D[CONST 20] ALU[DORQ] DEST[Q ( D%AMEM0 + APR-MODE )] JUMP[PIDSM2]
 $
                ; Turn on arithmetic interrupts.
PIDSM1: D[CONST 20] ALU[-D&Q] DEST[Q ( D%AMEM0 + APR-MODE )] JUMP[PIDSM2]
 $
                ; Turn off arithmetic interrupts
PIDSM2: D[D%AMEM0 + APR-MODE] ROT[CPU-MODE-ROT] DEST[CPU-MODE] LONG $
                ;Output new mode bits.
];[
        D[D%AMEM0 + APR-MODE] DEST[Q] $
                ;Get copy of current mode bits.
        D[APRSTS] MASK[3] COND[OBUS=0] JUMP[PIDSM1] $
                ; Jump if no PI channel.  We don't want to enable
 arithmetic
                ; micro-interrupts in that case.
        D[APRSTS] ROT[31.] COND[SIGNOFF] JUMP[PIDSM1] $
                ; J IF NO OV INT ENBL
                ; *** Don't both enables need to be checked???
        D[CONST 20] ALU[DORQ] DEST[Q ( D%AMEM0 + APR-MODE )] JUMP[PIDSM2]
 $
                ; Turn on arithmetic interrupts.
PIDSM1: D[CONST 20] ALU[-D&Q] DEST[Q ( D%AMEM0 + APR-MODE )] JUMP[PIDSM2]
 $
                ; Turn off arithmetic interrupts
PIDSM2: D[D%AMEM0 + APR-MODE] ROT[CPU-MODE-ROT] DEST[CPU-MODE] LONG $
                ;Output new mode bits.
].REPEAT FZAROV
        D[D%AMEM5] MASK[7] DEST[AR] C550 COND[-OBUS=0] PUSHJ[PI-GET-CHN]
 $
          ;Get bit corresponding to highest chan. in progress into the
 AR.
        D[D%AMEM5] DEST[Q] NORM $
        D[AR] ALU[-D&Q] DEST[D%AMEM5] NORM JUMP[PI-CHECK-RQS] $
         ;Clear highest chan. in progress, go check pending PI RQs
```

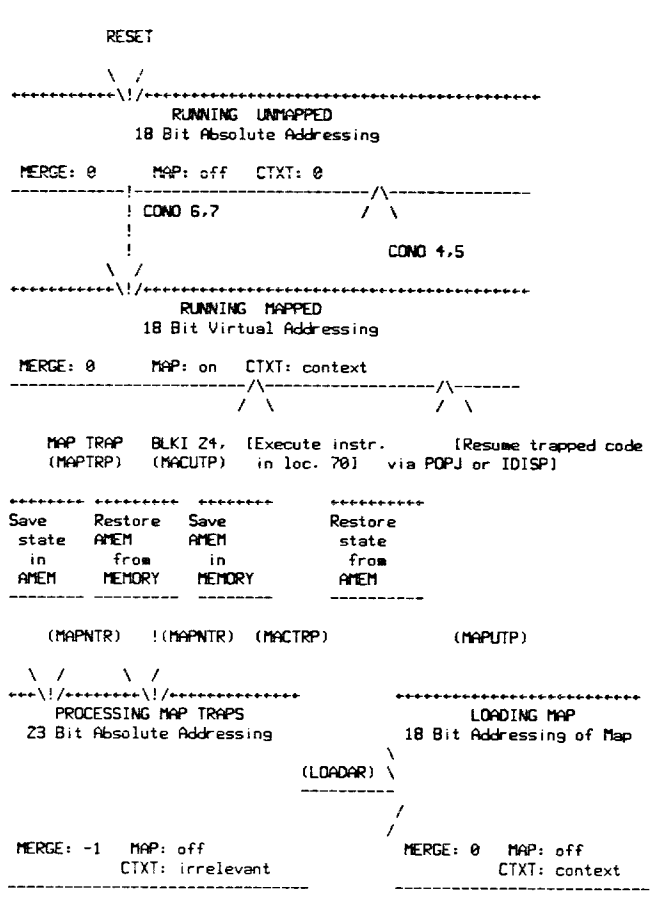
```
04 0396
04 0397                                                 PICONO:  ;Here from any CONO PI,
04 0398 10215 010731370000060554017623656714160000             D[MA] MASK[7]  DEST[AR] NORM $
04 0399 10216 010241170041060554160463654314170000             ALU[0] DEST[DEV-ADR] CLR-DEV-FROM-INTR NORM $
04 0400 10217 010730170000060554161623662314150000             D[PISTS]  DEST[Q] NORM $ ;GET STATUS A
04 0401 10220 010731006000000005361620256614560000             D[MA] ROT[21.]  C550 COND[-OBUS<0] JUMP[PIL7] $
04 0402 10221 010630170000060543603623655714160000             D[CONST 1] ROT[35. - 20.] ALU[DORQ] DEST[Q] NORM $
04 0403                                                          ; TURN ON PAR ERR INTRPT ENB.
04 0404 10222 010731006000000000516162025661456000     PIL7:   D[MA] ROT[20.]  C550 COND[-OBUS<0] JUMP[PIL8] $
04 0405 10223 010650170000060543603623655714160000             D[CONST 1] ROT[35. - 20.] ALU[-D&Q] DEST[Q] NORM $
04 0406                                                          ;TURN OFF PAR ERR INT ENB
04 0407 10224 010731006000000004761620256614560000     PIL8:   D[MA] ROT[19.] C550 COND[-OBUS<0] JUMP[PIL9] $
04 0408 10225 010650170000060544003623655714160000             D[CONST 1] ROT[35. - 19.] ALU[-D&Q] DEST[Q] NORM $
04 0409                                                          ;CLEAR MEM PAR ERR FLAG
04 0410                                                 .REPEAT WAITS [
04 0411                                                         D[APRSTS] DEST[ALU1 Q] $
04 0412                                                         D[CONST 1] ROT[35. - 24.] ALU[ALU1 -D&Q] DEST[APRSTS] $
04 0413                                                          ;And soft ECC error flag
04m0414                                                 ];[
04m0414 10226 010730570000060554161623661314160000             D[APRSTS] DEST[ALU1 Q] $
04m0414 10227 010651570000060542603043655714160000             D[CONST 1] ROT[35. - 24.] ALU[ALU1 -D&Q] DEST[APRSTS] $
04m0414                                                          ;And soft ECC error flag
04 0414                                                 ].REPEAT WAITS
04 0415 10230 010731020000000001361622256614560000     PIL9:   D[MA] ROT[5]  C550 COND[OBUS18] PUSHJ[PI-RESET] $
04 0416 10231 010731022000000002561620256614560000             D[MA] ROT[12]  C550 COND[-OBUS18] JUMP[PIL3] $
04 0417 10232 010630170000060541603623655714160000             D[CONST 1] ROT[7] ALU[DORQ] DEST[Q] NORM $ ; PI ON
04 0418 10233 010731022000000000236162025661456000     PIL3:   D[MA] ROT[11]  C550 COND[-OBUS18] JUMP[PIL4] $
04 0419 10234 010650170000060541603623655714160000             D[CONST 1] ROT[7] ALU[-D&Q] DEST[Q] NORM $ ; PI OFF
04 0420 10235 010731022000000000176162025661456000     PIL4:   D[MA] ROT[7]  C550 COND[-OBUS18] JUMP[PIL5] $
04 0421 10236 010630170000060554161623654314160000             D[AR] ALU[DORQ] DEST[Q] NORM $ ; CHNS ON
04 0422 10237 010731022000000002161620256614560000     PIL5:   D[MA] ROT[10]  C550 COND[-OBUS18] JUMP[PIL6] $
04 0423 10240 010650170000060554161623654314160000             D[AR] ALU[-D&Q] DEST[Q] NORM $ ; CHNS OFF
04 0424 10241 010231170000060554161103654314160000     PIL6:   ALU[Q] DEST[PISTS]  NORM $
04 0425 10242 010731022000020621561620256614560000             D[MA] ROT[6]  C550 COND[-OBUS18] JUMP[PI-CHECK-RQS] $
04 0426 10243 011710170000060554017623654314160000             D[AR] MASK[7] ALU[0-D] DEST[Q] NORM $
04 0427                                                                  ;GENERATED INTRPTS REQUESTED. CHECK TO
04 0428                                                                  ; MAKE SURE ONLY ONE CHN IS SPECIFIED.
04 0429 10244 010640170000060554017623654314160000             D[AR] MASK[7] ALU[D&Q] DEST[Q] NORM $
04 0430                                                         D[AR] MASK[7] ALU[D-Q] C600
04 0431                                                 .REPEAT 1 - WAITSFIX [
04 0432                                                          COND[-OBUS=0] JUMP[.] $ ;HANG HERE IF MORE THAN ONE.
04 0433                                                 ];.REPEAT 1 - WAITSFIX
04 0434                                                 .REPEAT WAITSFIX [
04 0435                                                          COND[-OBUS=0] JUMP[PIERR] $ ;LOSE IF MORE THAN ONE.
04m0436                                                 ];[
04m0436 10245 011621000000000014017620254114560000              COND[-OBUS=0] JUMP[PIERR] $ ;LOSE IF MORE THAN ONE.
04 0436                                                 ].REPEAT WAITSFIX
04 0437 10246 010731370000202474017622254314560000             D[AR] MASK[7]  DEST[AR] NORM PUSHJ[PI-GET-CHN] $
04 0438                                                                  ;GET BINARY CHN. NUMBER AND UNARY MASK.
04 0439 10247 010731170000203574161620254314560000             NORM JUMP[PIGEN1] $ ;GO GENERATE REQUEST.
04 0440
04 0441                                                 .REPEAT WAITSFIX [
04 0442                                                 PIERR:  D[CONST 45] DEST[AR] JUMP[HLTLOC] $
04 0443                                                                  ;PI system is screwed up.
04m0444                                                 ];[
04m0444 10250 010731370000000014113620255714560000     PIERR:  D[CONST 45] DEST[AR] JUMP[HLTLOC] $
04m0444                                                                  ;PI system is screwed up.
04 0444                                                 ].REPEAT WAITSFIX
04 0445
04 0446                                                 PICONISUB:
04 0447 10251 010730170000000114045623670314160000             D[AMEM-ABS 4]  MASK[18.] DEST[Q] NORM $
04 0448                                                           ;GET SYS ON AND CHN ON BITS.
04 0449 10252 010731370000000134021623670314160000             D[AMEM-ABS 5] MASK[10] DEST[AR] NORM $
04 0450                                                           ;GET PI IN PROG BITS
04 0451 10253 010630370000060542161623654314160000             D[AR] ROT[10] ALU[DORQ] DEST[AR Q] NORM $
04 0452 10254 010731370000000124421623670314160000             D[AMEM-ABS 5]  ROT[18.] MASK[10] DEST[AR] NORM $
04 0453                                                           ;GET WAITING RQ BITS, AND RETURN IN LEFT HALF.
04 0454 10255 010630370000060544561624254314160000             D[AR] ROT[18.] ALU[DORQ] DEST[AR Q] NORM POPJ $
04 0455
04 0456                                                 PI-RESET:
04 0457 10256 010241170000000114161543654314160000             ALU[0] DEST[AMEM-ABS 4] NORM $
04 0458 10257 010241170000000134161543654314160000             ALU[0] DEST[AMEM-ABS 5] NORM $
04 0459 10260 010240170000000154161543654314160000             ALU[0] DEST[AMEM-ABS 6 Q] NORM $
04 0460 10261 010731170000060554161624254314160000             NORM POPJ $
04 0461
04 0462
04 0463                                                 END-OF-PI-CODE:
04 0464
```

```
01 0060
01 0061                                                .insert F4MAP
```

```
01 0001
01 0002
01 0003
01 0004
01 0005
01 0006
01 0007
01 0008
01 0009
01 0010
01 0011
01 0012
01 0013
01 0014
01 0015
01 0016
01 0017
01 0018
01 0019
01 0020
01 0021
01 0022
01 0023
01 0024
01 0025
01 0026
01 0027
01 0028
01 0029
01 0030
01 0031
01 0032
01 0033
01 0034
01 0035
01 0036
```

```
COMMENT !   VALID 00031 PAGES
C REC  PAGE   DESCRIPTION
C00001 00001
C00002 00002    NOTES:
C00004 00003
C00007 00004
C00012 00005
C00016 00006
C00021 00007
C00025 00008
C00029 00009
C00031 00010
C00034 00011
C00037 00012
C00043 00013
C00045 00014
C00048 00015
C00050 00016
C00053 00017
C00055 00018
C00061 00019
C00064 00020
C00069 00021
C00074 00022
C00081 00023
C00085 00024
C00090 00025
C00097 00026
C00099 00027
C00103 00028
C00105 00029
C00118 00030
C00120 00031
C00123 ENDMK
C!;
```

```
02 0037                                 ;NOTES:
02 0038                                 ;PHYS PAGE #'S IN 71 AND 72 STILL RESTRICTED TO 20 BIT PHYS ADR SPACE.
02 0039                                 ;MUST JUMBLE 71 AND 72 UP TO YIELD BIGGER PAGE #'S (14 BITS LONG)
02 0040
02 0041
02 0042
02 0043                                 COMMENT %
02 0044
02 0045                                 PERMANENT DOCUMENTATION (MAINTAIN IT PLEASE):
02 0046
02 0047                                 How code works with respect to state saving and restoring:
02 0048
02 0049                                 Get trap.
02 0050                                 Save either short or long amem state immediately as appropriate.
02 0051
02 0052                                 Analyze situation:
02 0053                                 If just AR refill - do it and restore state from amem and resume.
02 0054                                 If need macro trap - store amem state in mem (long or short)
02 0055                                                      continue long or short mem state save
02 0056                                                      take macro trap.
02 0057
02 0058                                 When TENEX macro paging code wants to resume it does a BLKI.
02 0059                                 Long or short amem block is restored from mem.
02 0060                                 Any state in mem not contained in amem is restore directly to machine
02 0060                                  state.
02 0061                                 Start over again just as if trap had just taken.
02 0062
02 0063
```


RESET
RUNNING UNMAPPED
18 Bit Absolute Addressing
MERGE: 0
MAP: off
CTXT: 0
CONO 6,7
CONO 4,5
RUNNING MAPPED
18 Bit Virtual Addressing
MERGE: 0
MAP: on
CTXT: context
MAP TRAP (MAPTRP)
BLKI Z4, (MACUTP)
[Execute instr. in loc. 70]
[Resume trapped code via POPJ or IDISP]
Save state in AMEM
Restore AMEM from MEMORY
Save AMEM in MEMORY
Restore state from AMEM
(MAPNTR)
!(MAPNTR)
(MACTRP)
(MAPUTP)
PROCESSING MAP TRAPS
Z3 Bit Absolute Addressing
LOADING MAP
18 Bit Addressing of Map
(LOADAR)
MERGE: -1
MAP: off
CTXT: irrelevant
MERGE: 0
MAP: off
CTXT: context

```
04 0111
04 0112
04 0113                                 LONG STATE BLOCK:
04 0114
04 0115                                        ABS-AMEM STATE INDEX SYMBOLS STATE01 = 00, STATE02 = 01 ETC.
04 0116
04 0117                                        07 L = LC
04 0118                                        07 M = MEM BUS FLIP FLOP (MB/HOLD SELECT)
04 0119                                        07 A = MAP-USER-SR ACTIVE
04 0120
04 0121                                 MEM !0!1!2!3!4!5!6!7!8!9!0!1!2!3!4!5!6!7!8!9!0!1!2!3!4!5!6!7!8!9!0!1!2!3!
04 0121                                 4!5!
04 0122                                 NDX
04 0123                                     !-------------------------------------------------------------------
04 0123                                 ---!
04 0124                                 00  !0 0 0!                        MB-STATUS
04 0124                                     !
04 0125                                     !-------------------------------------------------------------------
04 0125                                 ---!
04 0126                                 01  !                              IR
04 0126                                     !
04 0127                                     !-------------------------------------------------------------------
04 0127                                 ---!
04 0128                                 02  !                              MA
04 0128                                     !
04 0129                                     !-------------------------------------------------------------------
04 0129                                 ---!
04 0130                                 03  !                              ALU0 Q
04 0130                                     !
04 0131                                     !-------------------------------------------------------------------
04 0131                                 ---!
04 0132                                 04  !                              AR
04 0132                                     !
04 0133                                     !-------------------------------------------------------------------
04 0133                                 ---!
04 0134                                 05  !                              HOLD
04 0134                                     !
04 0135                                     !-------------------------------------------------------------------
04 0135                                 ---!
04 0136                                 06  !                              MB
04 0136                                     ! A
04 0137                                     !-------------------------------------------------------------------
04 0137                                 ---! !
04 0138                                 07  !      AMEM-P       !    CPUMODE!          ! DEV-ADR !          ! MAPSR
04 0138                                  !A! M
04 0139                                     !-------------------------------------------------------------------
04 0139                                 ---! E
04 0140                                 10  !          LOOP-CTR              !
04 0140                                     ! M
04 0141                                     !-------------------------------------------------------------------
04 0141                                 ---!--
04 0142                                 11  !          PC-FLAGS                    !              PC
04 0142                                     ! M
04 0143                                     !-------------------------------------------------------------------
04 0143                                 ---! E
04 0144                                 12  !          JMEM-P        !
04 0144                                     ! M
04 0145                                     !-------------------------------------------------------------------
04 0145                                 ---!
04 0146                                 13  !      @<JMEM-P)+0            !
04 0146                                     !
04 0147                                     !-------------------------------------------------------------------
04 0147                                 ---!
04 0148                                 14  !      @<JMEM-P)+1            !
04 0148                                     !
04 0149                                     !-------------------------------------------------------------------
04 0149                                 ---!
04 0150                                 15  !      @<JMEM-P)+2            !
04 0150                                     !
04 0151                                     !-------------------------------------------------------------------
04 0151                                 ---!
04 0152                                 16  !      @<JMEM-P)+3            !
04 0152                                     !
04 0153                                     !-------------------------------------------------------------------
04 0153                                 ---!
04 0154                                 17  !      @<JMEM-P)+4            !
04 0154                                     !
04 0155                                     !-------------------------------------------------------------------
04 0155                                 ---!
04 0156
04 0157                                     !0!1!2!3!4!5!6!7!8!9!0!1!2!3!4!5!6!7!8!9!0!1!2!3!4!5!6!7!8!9!0!1!2!3!
04 0157                                 4!5!
04 0158
04 0159
```

```
05 0160
05 0161
05 0162                                 SHORT STATE BLOCK:
05 0163
05 0164                                         ABS-AMEM STATE INDEX SYMBOLS STATE01 = 00, STATE02 = 01 ETC.
05 0165
05 0166                                         03 A = MAP-USER-SR ACTIVE
05 0167
05 0168                                 MEM !0!1!2!3!4!5!6!7!8!9!0!1!2!3!4!5!6!7!8!9!0!1!2!3!4!5!6!7!8!9!0!1!2!3!
05 0168                                 4!5!
05 0169                                 NDX
05 0170                                     !-------------------------------------------v-----------------------
05 0170                                 ---!
05 0171                                 00  !1 0 0!                        MB-STATUS
05 0171                                    ! ↑
05 0172                                     !-------------------------------------------------------------------
05 0172                                 ---!
05 0173                                 01  !                               IR
05 0173                                    ! A
05 0174                                     !-------------------------------------------------------------------
05 0174                                 ---! !
05 0175                                 02  !                               MA
05 0175                                    ! M
05 0176                                     !-------------------------------------------------------------------
05 0176                                 ---! E
05 0177                                 03  !                   !    CPUMODE!                              ! MAPSR
05 0177                                  !A! M
05 0178                                     !-------------------------------------------------------------------
05 0178                                 ---!--
05 0179                                 04  !         PC-FLAGS                  !       !          MIPC
05 0180                                    ! M
05 0180                                     !-------------------------------------------------------------------
05 0181                                 ---! E
05 0181                                 05  !                                 PC
05 0182                                    ! M
05 0182                                     !-------------------------------------------------------------------
05 0183                                 ---! V
05 0184
05 0184                                     !0!1!2!3!4!5!6!7!8!9!0!1!2!3!4!5!6!7!8!9!0!1!2!3!4!5!6!7!8!9!0!1!2!3!
05 0185                                 4!5!
05 0186
05 0187                                 MAGIC PAGER MAIN MEMORY LOCATIONS:
05 0188
05 0189                                     !0!1!2!3!4!5!6!7!8!9!0!1!2!3!4!5!6!7!8!9!0!1!2!3!4!5!6!7!8!9!0!1!2!3!
05 0189                                 4!5!
05 0190
05 0191                                     !-------------------------------------------------------------------
05 0191                                 ---!
05 0192                                 71  !       ! LMT ! PHYS USER PT PAGE # ! ACBASE  !     !  PHYS PSB PAGE
05 0192                                  #  !
05 0193                                     !-------------------------------------------------------------------
05 0193                                 ---!
05 0194                                 72  !    AGE REG      !                                    !  PHYS STATE PG
05 0194                                  #  !
05 0195                                     !-------------------------------------------------------------------
05 0195                                 ---!
05 0196
05 0197                                     !0!1!2!3!4!5!6!7!8!9!0!1!2!3!4!5!6!7!8!9!0!1!2!3!4!5!6!7!8!9!0!1!2!3!
05 0197                                 4!5!
05 0198
05 0199
05 0200                                 30  # AR REFILLS
05 0201                                 31  # CONO 0'S
05 0202
05 0203                                 33  # CONO 3'S
05 0204                                 34  # CONO 1'S
05 0205
05 0206
05 0207                                 MAGIC PSB LOCATIONS:
05 0208
05 0209                                 567 INDEX INTO PSB TO READ STATE BLOCK PTR
05 0210
05 0211                                 571 TRAP STATUS WORD
05 0212                                 572 UNUSED (USED TO HOLD STORE DATA)
05 0213                                 573 HOLDS PC FROM INTRUPT MACRO INSTR IN 70
05 0214
05 0215
```

```
06 0216
06 0217
06 0218                                 MAP AMEM USAGE:
06 0219
06 0220                                        02 R = MAP RESIDIENT MONTIOR (PAGE 1)
06 0221                                        02 F = RESIDENT MONITOR (PAGE 1) IS CURRENTLY UNMAPPED (FORCED
06 0221                                 1:1 MAP)
06 0222
06 0223                                        13 R = XCT ACCESS REQUIRED
06 0224                                        13 W = READ ACCESS REQUIRED
06 0225                                        13 X = WRITE ACCESS REQUIRED
06 0226
06 0227                                  A-
06 0228                                 MEM !0!1!2!3!4!5!6!7!8!9!0!1!2!3!4!5!6!7!8!9!0!1!2!3!4!5!6!7!8!9!0!1!2!3!
06 0228                                 4!5!
06 0229                                 NDX
06 0230                                    !-------------------------------------------------------------------
06 0230                                 ---!
06 0231                                 00  !                    INTERRUPT DISPATCH (UNUSED)
06 0231                                    !
06 0232                                    !-------------------------------------------------------------------
06 0232                                 ---!
06 0233                                 01  !   AGE REG      !       ACBASE ADR (FROM 0TH PT ENTRY FOR WAITS)
06 0233                                    !
06 0234                                    !-------------------------------------------------------------------
06 0234                                 ---!
06 0235                                 02  !R!F!                ADR LIMIT (SMALLEST ILLEGAL VIRTUAL ADR)
06 0236                                    !-------------------------------------------------------------------
06 0236                                 ---!
06 0237                                 03  !                    USER PAGE TABLE PHYS ADR
06 0237                                    !
06 0238                                    !-------------------------------------------------------------------
06 0238                                 ---!
06 0239                                 04  !                    PSB PHYS ADR
06 0239                                    !
06 0240                                    !-------------------------------------------------------------------
06 0240                                 ---!
06 0241                                 05  !                    STATE/AC PAGE PHYS ADR
06 0241                                    !
06 0242                                    !-------------------------------------------------------------------
06 0242                                 ---!
06 0243                                 06  !
06 0243                                    !
06 0244                                    !-------------------------------------------------------------------
06 0244                                 ---!
06 0245                                 07  !
06 0245                                    !
06 0246                                    !-------------------------------------------------------------------
06 0246                                 ---!
06 0247                                 10  !
06 0247                                    !
06 0248                                    !-------------------------------------------------------------------
06 0248                                 ---!
06 0249                                 11  !
06 0249                                    !
06 0250                                    !-------------------------------------------------------------------
06 0250                                 ---!
06 0251                                 12  !
06 0251                                    !
06 0252                                    !-------------------------------------------------------------------
06 0252                                 ---!
06 0253                                 13  !                        !R!W!X! !       FAILING MA
06 0253                                    !
06 0254                                    !-------------------------------------------------------------------
06 0254                                 ---!
06 0255                                 14  !                        SCRATCH
06 0255                                    !
06 0256                                    !-------------------------------------------------------------------
06 0256                                 ---!
06 0257                                 15  !                   1ST EXEC SWAPPED VIRTUAL ADR
06 0257                                    !
06 0258                                    !-------------------------------------------------------------------
06 0258                                 ---!
06 0259                                 16  !                      EXEC-CTXT COPY
06 0259                                    !
06 0260                                    !-------------------------------------------------------------------
06 0260                                 ---!
06 0261                                 17  !                      USER-CTXT COPY
06 0261                                    !
06 0262                                    !-------------------------------------------------------------------
06 0262                                 ---!
06 0263
06 0264                                    !0!1!2!3!4!5!6!7!8!9!0!1!2!3!4!5!6!7!8!9!0!1!2!3!4!5!6!7!8!9!0!1!2!3!
06 0264                                 4!5!
06 0265
06 0266
```

```
07 0267
07 0268                                 New, Improved, Final (for a while) SPECIAL MB STATUS BITS:
07 0269
07 0270                                         INSTR FETCH => 1B7!1B24
07 0271
07 0272
07 0273                                 BIT   MEANING
07 0274                                 ---   -------
07 0275
07 0276                                 04      MAP Parity Error
07 0277                                 05      MAP Tag Failure
07 0278                                 06      Illegal Access
07 0279
07 0280                                 07 Memory cycle occurred during a uinst. which did NOT have MI MEM WAIT
07 0281                                         on. Will be off only if DF/WT or DF/IF started operation causing
07 0281                                  trap.
07 0282                                 08 Memory cycle was started by a DFRQ.
07 0283                                 09 ERROR !! A Write was started before the last Read data was used !!
07 0284
07 0285                                 10      HARD ECC
07 0286                                 11      ANY ECC
07 0287                                 12:17   ECC SYNDROME
07 0288
07 0289                                 18 (MAPF[02] of a DF cycle) Unused.  Use of this bit presents a problem
07 0289                                  when
07 0290                                         untrapping to a D[MEM] cycle from ECC, since we may then get a
07 0290                                  MAPTRP
07 0291                                         from the same uinst., in which case this bit will be wrong.
07 0292
07 0293                                 19 (MAPF[01] of a DF cycle)  Unused.  See comment for bit 18.
07 0294
07 0295                                 20 (MAPF[10] of the aborted (trapped) cycle)
07 0296                                          On a DF trap, means a R-M-W is intended - trap may be from write
07 0296                                  prot.
07 0297                                 21 (MAPF[04] of the aborted (trapped) cycle)
07 0298                                         On any MAP trap, means resume execution by re-fetching instr (as
07 0298                                  in IF)
07 0299                                 22:23  (MAPF[02:01] of the aborted (trapped) cycle)
07 0300                                         Decoded:
07 0301                                         00      normal
07 0302                                         01      Call this guy's special fix-up code via
07 0303                                                  TRUE LBPOPJ $  ;...he will return to MAP-RET with the
07 0304                                                  stack containing the proper place to resume execution
07 0304                                  later.
07 0305                                         10      failing addr. is in IR18:35, and NOT in MA.
07 0306                                         11      failing addr. is in AR18:35, and NOT in MA.
07 0307                                 24 Means trap was from the cycle following a Principle Idisp.  Therefore,
07 0307                                  the
07 0308                                         losing reference was either the fetch of the current instruction,
07 0308
07 0309                                         or the data fetch of a DF/WT or DF/IF  during the first uinst.
07 0309                                  after
07 0310                                         the dispatch that entered the current instruction.
07 0311                                          If on, only short state need be saved; we will resume with an
07 0311                                  IDISP
07 0312                                         either after re-fetching the instr. or (if the fetch is a DF and
07 0312
07 0313                                         bit 21 is off) using the contents of the IR.
07 0314                                 25      Last memory operation started was a Data Store.
07 0315
07 0316                                 SUMMARY: CYCLE-TYPE = If (25 & -24) Then WRITE
07 0317                                                       Else If (08 & (-07 Or -24)) Then DATA-FETCH
07 0318                                                       Else INSTRUCTION-FETCH
07 0319
07 0320                                 On fetches, ECC and MAP traps occur during (and abort !) the cycle
07 0320                                  containing
07 0321                                  the D[MEM] (for data fetches) or the cycle following the IDISP (for
07 0322                                  instruction fetches).
07 0323
07 0324                                 On stores, MAP traps occur during the cycle WHEN THE STORE IS HAPPENING.
07 0324
07 0325
07 0326                                    %    ;END OF HUGE DOCUMENTATION COMMENT
07 0327
07 0328
```

```
00 0329
00 0330
00 0331                                         ;MAP PARAMETERS
00 0332
00 0333     24      PGR = 24                    ;THE MACRO PAGER DEVICE #
00 0334
00 0335                                         .REPEAT WAITS [
00 0336                                         GRUNTSW = 0                 ;WAITS always aborts instruction after
00 0336                                          page
00 0337                                                                     ;faults.
00m0338                                                 ];[
00m0338     0 GRUNTSW = 0                   ;WAITS always aborts instruction after
00m0338                                          page
00m0338                                                                     ;faults.
00 0338                                                 ].REPEAT WAITS
00 0339
00 0340                                         ;MAGIC EXEC SPACE MAIN MEM LOCS
00 0341
00 0342                                         .REPEAT 1 - WAITS [
00 0343
00 0344                                         N-AR-REFILLS = 30           ;# AR REFILLS
00 0345                                         N-COND-0 = 31               ;# COND 0'S
00 0346                                         EDDT-LIKES-THIS = 32        ;DON'T TOUCH
00 0347                                         N-COND-3 = 33               ;# COND 3'S
00 0348                                         N-COND-1 = 34               ;# COND 1'S
00 0349
00 0350                                                 ];REPEAT 1 - WAITS
00 0351
00 0352     70      PGR70 = 70                  ;CONTAINS INTERRUPT INSTRUCTION
00 0353     71      PGR71 = 71                  ;CONTAINS UPT, PSB, ACBAS, ADR LIMIT
00 0354     72      PGR72 = 72                  ;CONTAINS AGE, STATE/AC PAGE
00 0355
00 0356     3000    MMAP = 3000                 ;BASE OF MMAP
00 0357     4000    CST0 = 4000                 ;BASE OF CST0
00 0358     20000   SPT = 20000                 ;BASE OF SPT
00 0359
00 0360
00 0361                                         ;MAGIC PSB LOCS
00 0362
00 0363                                         .REPEAT GRUNTSW [
00 0364                                         PSB567 = 567                ;INDEX INTO PSB TO READ STATE BLOCK PTR
00 0365                                                 ];.REPEAT GRUNTSW
00 0366
00 0367     571     PSB571 = 571                ;TRAP STATUS WORD
00 0368     572     PSB572 = 572                ;UNUSED (USED TO HOLD STORE DATA)
00 0369     573     PSB573 = 573                ;HOLDS PC FROM INTRUPT MACRO INSTR IN 70
00 0369
00 0370
00 0371
00 0372                                         ;Local (MAPDEV) AMEM assignments:
00 0373
00 0374     41      AGE/ACB = D%AMEM1
00 0375     42      FCS/LMT = D%AMEM2
00 0376     43      UPTADR  = D%AMEM3
00 0377     44      PSBADR  = D%AMEM4
00 0378     45      STAADR  = D%AMEM5
00 0379
00 0380     52      SAVE-MAP-SR     = D%AMEM12
00 0381     53      CYTYPE  = D%AMEM13      ;cycle-type code in left half, failing address in
00 0381                                      right
00 0382     54      ATEMP   = D%AMEM14      ;scratch
00 0383     55      SWPMON  = D%AMEM15      ;First non-forced exec address
00 0384     56      XCTXT   = D%AMEM16      ;Exec mode context
00 0385     57      UCTXT   = D%AMEM17      ;User mode context
00 0386
00 0387                                         ;Global (MAPDEV) AMEM assignments:
00 0388
00 0389     21      MAP-AGE/ACB = (MAPDEV * 20) + (AGE/ACB - 40)
00 0390     25      MAP-STAADR  = (MAPDEV * 20) + (STAADR  - 40)
00 0391     36      MAP-XCTXT   = (MAPDEV * 20) + (XCTXT   - 40)
00 0392     37      MAP-UCTXT   = (MAPDEV * 20) + (UCTXT   - 40)
00 0393
00 0394
00 0395                                         ;STATE AMEM BLOCK (ALWAYS ACCESSED VIA AMEM-ABS)
00 0396
00 0397     260     STATE00 = (STATEDEV * 20) + 0   ;INT DISPATCH, DON'T TOUCH (REALLY
00 0397                                              UNUSED)
00 0398
00 0399     261     STATE01 = (STATEDEV * 20) + 1   ;SAME FOR LONG AND SHORT STATES
00 0400     262     STATE02 = (STATEDEV * 20) + 2   ; "
00 0401     263     STATE03 = (STATEDEV * 20) + 3   ; "
00 0402     264     STATE04 = (STATEDEV * 20) + 4   ;DIFFERENT FOR LONG AND SHORT STATES
00 0403     265     STATE05 = (STATEDEV * 20) + 5   ;NOT USED FOR SHORT STATE
00 0404     266     STATE06 = (STATEDEV * 20) + 6   ; "
00 0405     267     STATE07 = (STATEDEV * 20) + 7   ; "
00 0406     270     STATE10 = (STATEDEV * 20) + 10  ; "
00 0407     271     STATE11 = (STATEDEV * 20) + 11  ; "
00 0408
00 0409     11        AMEMSTATESIZE = STATE11 - STATE01 + 1 ;SIZE OF STATE BLOCK WHEN IN AMEM
00 0409
00 0410
00 0411     40      MMEMSTATESIZE = 32.                 ;SIZE OF STATE BLOCK WHEN IN MAIN
00 0411                                         MEM
00 0412                                                                                 ;POWER OF 2 FOR NOW
00 0413
00 0414     5 JMEMSIZE = 5.                     ;# JMEM STACK LOCS SAVED IN STATE
00 0414
00 0415
00 0416
```

```
09 0417
09 0418
09 0419
09 0420
09 0421
09m0422                                                 .opcode[702]   [xlist
09 0422                                                  list  ]:MAP IF DEVICE PGR (UNUSED IF 20)
09 0423
09 0424                                                       CLR-DEV-FROM-INTR
09 0425 07604 010731170040000014002460Z5561457000             D[CONST MAPDEV] DEST[DEV-ADR] JUMP[MAPIOT] T400 $
09 0426                                                       CLR-DEV-FROM-INTR
09 0427 07605 010731170040000014002460Z5561457000             D[CONST MAPDEV] DEST[DEV-ADR] JUMP[MAPIOT] T400 $
09 0428
09 0429
09m0430                                                 .reloc
09m0430                                                [.USE[HILOC]
09m0430                                                [ xlist
09 0431                                                  list ]                                  ]
09m0432                                                MAPIOT: UIOTRP[MUUO]
09m0432                                                 [      COND[USER] JUMP[MUUO] NORM
09 0432 10262 010731124000000014161620254314560000      ]$
09 0433 10263 010731006000000000236162025761456000           D[IR] ROT[9.] COND[SIGNOFF] JUMP[MAIN] C550 $
09 0434                                                              ;BIT 09 MUST BE ON OR ELSE NOP FOR DEV 20
09 0435 10264 010730170000060543Z076Z365775416000             D[IR] ROT[13.] MASK[3] DEST[Q] TNRM $
09 0436                                                              ;GET BITS 10:12
09 0437 10265 010601370000000000000162365535416000            D[LABEL MPIOTD] ALU[D+Q] DEST[AR] TNRM $
09 0438 10266 010731170000000000160007254114560000            D[AR] ROT[MUA-ROT] ODISP LONG $
09 0439                                                              ;DISPATCH ON INSTRUCTION TYPE
09 0440
09 0441
09 0442 10267 010731170000000014161620254354560000      MPIOTD: JUMP[MAPBKI] TNRM $             ;BLKI   RESUME FROM TRAP
09 0443 10270 010731170000000014161620254354560000              JUMP[MAPDTI] TNRM $             ;DATAI  READ EXEC MAP WD
09 0444 10271 010731170000000014161620254354560000              JUMP[MAPBKO] TNRM $             ;BLKO   SET 1ST SWAPPING PG
09 0445 10272 010731170000000014161620254354560000              JUMP[MAPDTO] TNRM $             ;DATAO  SET EXEC MAP WD
09 0446 10273 010731170000000014161620254354560000              JUMP[MAPCNO] TNRM $             ;CONO   CONTROL PAGER
09 0447 10274 010731170000000014161620254354560000              JUMP[MAPCNI] TNRM $             ;CONI   UNUSED
09 0448 10275 010731170000000014161620254354560000              JUMP[MAPCS2] TNRM $             ;CONS2  READ USER MAP WD
09 0449 10276 010731170000000014161620254354560000              JUMP[MAPCSO] TNRM $             ;CONSO  SET USER MAP WD
09 0450
09 0451
09 0452
```

```
10 0453
10 0454
10 0455                                         ;CONSO LOADS THE MAP-TAG OF THE MAP WD MAPPING USER EFFECTIVE ADR OF
10 0455                                          INSTR.
10 0456                                         ;WITH CONTENTS OF AC 0, MAP-MA FROM AC 1
10 0457
10 0458 10277 010730170000000000074036Z0Z5575456000    MAPCSO: D[CONST 1] ROT[35. - 5] DEST[Q] TNRM JUMP[MAPD10] $
10 0459                                                         ;SET USER MODE AND JUMP TO LOAD MAP LOC.
10 0460
10 0461                                         ;DATAO LOADS THE MAP-TAG OF THE MAP WD MAPPING EXEC EFFECTIVE ADR OF
10 0461                                          INSTR.
10 0462                                         ;WITH CONTENTS OF AC 0, MAP-MA FROM AC 1
10 0463                                         ;THIS REPLACES BBN CONO 2. (SORT OF)
10 0464
10 0465
10 0466 10300 0102401700000605541616Z365435416000      MAPDTO: ALU[0] DEST[Q] TNRM $
10 0467 10301 0107313700000605541616Z364631416000      MAPD10: D[PC-FLAGS] DEST[AR] T300 $
10 0468 10302 0102311700000605541604Z365421417000              ALU[Q] DEST[PC-FLAGS] T400 $
10 0469
10 0470                                                 .REPEAT 1 - WAITS [
10 0471                                                         D[AC0] ALU[NOTA] DEST[MAP-TAG] LONG $
10 0472                                                                 ;LOAD MA'S MAP-TAG WITH AC0 BITS 03:26
10 0473                                                         D[AC1] ALU[A] DEST[Q] $
10 0474                                                         D[CONST 3] ROT[35. - 11.] ALU[D#Q] DEST[MAP-MA] LONG $
10 0475                                                                 ;LOAD MA'S MAP-MA WITH AC1 BITS 03:26
10 0476                                                                 ;Complement the Read and Write access bits.
10 0477                                                             ];.REPEAT 1 - WAITS
10 0478
10 0479                                                 .REPEAT WAITS [
10 0480                                                         D[CONST 1] ROT[35. - 3.] ALU[NOTD] DEST[MAP-TAG] LONG $
10 0481                                                                 ;Invalidate this entry
10m0482                                                             ];[
10m0482 10303 0107711700000605500025636555141700Z      10m0482         D[CONST 1] ROT[35. - 3.] ALU[NOTD] DEST[MAP-TAG] LONG $
10m0482                                                                 ;Invalidate this entry
10 0482                                                             ].REPEAT WAITS
10 0483
10 0484 10304 5407311700000605541604Z325420417000              D[AR] DEST[PC-FLAGS] NEOI $
10 0485
10 0486
10 0487                                         ;STORE MAP-TAG FOR MAP WORD MAPPING USER EFFECTIVE ADR OF INSTRUCTION
10 0488                                         ;INTO AC 0, MAP-MA INTO AC 1
10 0489
10 0490 10305 010730170000000000074036Z0Z5575456000    MAPCS2: D[CONST 1] ROT[35. - 5] DEST[Q] TNRM JUMP[MAPDI1] $
10 0491                                                         ;SET USER FOR SURE AND JUMP TO READ MAP-TAG
10 0492
10 0493
10 0494                                         ;STORE MAP-TAG FOR MAP WORD MAPPING EXEC EFFECTIVE ADR OF INSTRUCTION
10 0495                                         ;INTO AC 0, MAP-MA INTO AC 1
10 0496
10 0497 10306 0102401700000605541616Z365435416000      MAPDTI: ALU[0] DEST[Q] TNRM $
10 0498 10307 0107313700000605541616Z364631416000      MAPDI1: D[PC-FLAGS] DEST[AR] T300 $
10 0499 10310 0102311700000605541604Z365421417000              ALU[Q] DEST[PC-FLAGS] T400 $
10 0500 10311 0107211700000605540000036555141700Z              ALUI[-1] DEST[EOBUS-NULL] LONG $
10 0501                                                          ;Wait for EXEC to settle, charge up MC's Local Eobus to all 1's.
10 0501
10 0502 10312 0107731700000605541600037421141600Z              D[MAP-TAG] ALU[NOTD] DEST[B AC0] LONG $
10 0503                                                                 ;Read MA's MAP-TAG (greatly mangled) into AC0.
10 0504 10313 0107331700000605541600237415141600Z              D[MAP-MA] ALU[D] DEST[B AC1] LONG $
10 0505                                                                 ;READ MA'S MAP-MA INTO AC1
10 0506 10314 5407311700000605541604Z325420417000              D[AR] DEST[PC-FLAGS] NEOI $
10 0507
10 0508
10 0509
```

```
11 0510
11 0511
11 0512                                                 ;SET FIRST SWAPPING ADR OF MONITOR
11 0513                                                 ;FROM EFFECTIVE ADR OF INSTR.
11 0514
11 0515 10315 010731170000000001404532225675456000     MAPBKO: D[MA] MASK[18.] DEST[D%AMEM15] PUSHJ[FORCEM] TNRM $
11 0516                                                                 ;Get 1st adr of swappable monitor in D%AMEM15, and set
11 0517                                                                 ; up the map to force 1:1 mapping for pages
11 0517                                                  0,Z:<D%AMEM15-1)
11 0518 10316 010241170000000001416134225431456000             ALU[0] DEST[D%AMEM16] PUSHJ[EXECLR] $
11 0519                                                                 ;Reset context and sweep all non-forced exec map wds,
11 0520                                                                 ; guaranteeing only proper map wds have comparator
11 0520                                                  disabled.
11 0521 10317 540731170000060554161623254204160000             NEOI $
11 0522
11 0523
11 0524                                                 ;MAP CONI
11 0525
11 0526 10320 540731170000060554161623254204160000     MAPCNI: ILGIOT $
11 0527                                                                 ;CONI NOT YET DEFINED
11 0528
11 0529
11 0530                                                 ;MAP CONO
11 0531
11 0532 10321 010731004000000000656162025761456000     MAPCNO: D[IR] ROT[26.] CS50 SIGNON JUMP[MAPACB] $
11 0533                                                                 ;If 1000 bit is on, reload AC PTR only.
11 0534 10322 010731004000000000636162025761456000             D[IR] ROT[25.] CS50 SIGNON JUMP[MAPAGE] $
11 0535                                                                 ;If 2000 bit is on, reload AGE only.
11 0536 10323 010730170000060554007623657754160000             D[IR] MASK[3] DEST[Q] TNRM $
11 0537                                                                 ;GET INDEX INTO BBNCNO IN Q
11 0538 10324 010601370000000000000162365535416000             D[LABEL BBNCNO] ALU[D+Q] DEST[AR] TNRM $
11 0539 10325 010731170000000000160007254114560000             D[AR] ROT[MUA-ROT] ODISP LONG $
11 0540                                                                 ;MIPC MUST BE 00:13
11 0541                                                                 ;TREAT AS NORMAL CONO IF NOT SPECIAL CASE
11 0542
11 0543
11 0544                                                 ;BBN DEFINED CONOS FOR PAGER
11 0545
11 0546 10326 010731170000000001416162025435456000     BBNCNO: JUMP[MAPCO0] TNRM $     ;CONO 0 - RESET PAGER PARAMETERS
11 0547 10327 010731170000000001416162025435456000             JUMP[MAPCO1] TNRM $     ;CONO 1 - CLEAR ALL EXEC MAP WDS
11 0548 10330 010731170000000001416162025435456000             JUMP[ITRAP] TNRM $      ;CONO 2 - ILLEGAL, REPLACED BY DATAO
11 0549 10331 010731170000000001416162025435456000             JUMP[MAPCO3] TNRM $     ;CONO 3 - CLEAR ALL USER MAP WDS
11 0550 10332 010731170000000001416162025435456000             JUMP[MAPCO4] TNRM $     ;CONO 4 - TURN PGR OFF
11 0551 10333 010731170000000001416162025435456000             JUMP[MAPCO5] TNRM $     ;CONO 5 - SAME AS CONO 4
11 0552 10334 010731170000000001416162025435456000             JUMP[MAPCO6] TNRM $     ;CONO 6 - PGR ON, DON'T MAP EXEC PAGE 1
11 0553 10335 010731170000000001416162025435456000             JUMP[MAPCO7] TNRM $     ;CONO 7 - PGR ON, MAP EXEC PAGE 1
11 0554
11 0555
```

```
12 0556
12 0557
12 0558
12 0559                                         ;COND 1000!X
12 0560                                         ;RELOAD THE AC BASE REG. FROM LOW 5 BITS OF E
12 0561
12 0562 10336 010731370000060554013623657754160000     MAPACB: D[IR] MASK[5] DEST[AR] TNRM $
12 0563 10337 010730171776000000001623655354160000             D[NLIT 777000000000] DEST[Q] TNRM $
12 0564                                                         ;MASK TO RETAIN AGE
12 0565 10340 010640170000060554161623660754160000             D[D%AMEM1] ALU[D&Q] DEST[Q] TNRM $
12 0566                                                         ;GET POSITIONED AND ISOLATED AGE IN Q
12 0567 10341 010631170000060541161023654344160000             IFRQ D[AR] ROT[4] ALU[DORQ] DEST[D%AMEM1] TNRM $
12 0568                                                         ;ENTER AGE!ACBASE IN AMEM
12 0569 10342 540731170000060554161623254254160000             EOI $
12 0570
12 0571
12 0572                                         ;COND 2000!X
12 0573                                         ;RELOAD THE AGE REG. FROM LOW 9 BITS OF E
12 0574
12 0575 10343 010731370000060554023623657754160000     MAPAGE: D[IR] MASK[9.] DEST[AR] TNRM $
12 0576                                                         ;GET AGE FROM IR INTO AR
12 0577 10344 010730170000060554067623660754160000             D[D%AMEM1] MASK[27.] DEST[Q] TNRM $
12 0578                                                         ;GET ALL OF MAP D%AMEM1 EXCEPT FOR AGE
12 0579 10345 010631170000060546761023654344160000             IFRQ D[AR] ROT[27.] ALU[DORQ] DEST[D%AMEM1] TNRM $
12 0580                                                         ;ENTER AGE!PREVIOUS CONTENTS
12 0581                                                         ;START INSTR FETCH
12 0582 10346 540731170000060554161623254254160000             EOI $
12 0583
12 0584
12 0585                                         ;COND 0
12 0586
12 0587 10347 010731170000000014161622254314560000     MAPCO0: PUSHJ[MEMADR-23BIT-ABS] $
12 0588 10350 010731170013060554161623654314160000             SPEC[MAP-DISABLE] $
12 0589
12 0590                                         .REPEAT 1 - WAITS [
12 0591                                                 D[CONST N-COND-0] DEST[MA] TNRM $
12 0592                                                         ;LOAD MA WITH MAGIC LOC, DISABLE MAP
12 0593                                                 DF/WT D[MEM] ALU[D+1] DEST[MEMSTD] $
12 0594                                                         ;BUMP # MAP RESETS VIA COND 0
12 0595
12 0596                                                           ];.REPEAT 1 - WAITS
12 0597
12 0598 10351 710731170000060554163623655754160000             D[CONST PGR71] DEST[MA] TNRM $
12 0599                                                         ;LOAD MA WITH PGR71
12 0600                                                         ;FETCH MAGIC LOC
12 0601 10352 030731370000060554027623654753160000             DF/WT D[MEM] MASK[11.] DEST[AR] TNRM $
12 0602                                                         ;DATA AT MEM, MASK OFF PSB PHYS ADR
12 0603 10353 010731170000060542361103654354160000             D[AR] ROT[9.] DEST[D%AMEM4] TNRM $
12 0604                                                         ;STORE PSB ADR IN D%AMEM4
12 0605 10354 010731370000060544427623654755160000             D[MEM] ROT[18.] MASK[11.] DEST[AR] TNRM $
12 0606                                                         ;MASK OFF PHYS UPT ADR
12 0607 10355 010731170000060542361063654354160000             D[AR] ROT[9.] DEST[D%AMEM3] TNRM $
12 0608                                                         ;STORE UPT ADR IN D%AMEM3
12 0609
12 0610                                         .REPEAT 1 - WAITS [     ;WAITS sets D%AMEM1 with first page in user page
12 0610                                          table
12 0611
12 0612                                                 D[MEM] ROT[23.] MASK[5] DEST[AR] TNRM $
12 0613                                                         ;MASK OFF AC BASE ADR (SHIFTED DOWN 4)
12 0614                                                 D[AR] ROT[4] DEST[D%AMEM1] TNRM $
12 0615                                                         ;STORE AC BASE ADR IN D%AMEM1 SANS AGE
12 0616
12 0617                                                           ];.REPEAT 1 - WAITS
12 0618
12 0619 10356 011711202000000001607620254515560000             D[MEM] ROT[7] MASK[3] ALU[0-D] DEST[AR] COND[ZERO] JUMP[MAPCOB]
12 0619                                          C600 $
12 0620                                                         ;GET ADDRS LIMIT, J IF AR/ 0
12 0621 10357 010731370000060554007623654354160000             D[AR] MASK[3] DEST[AR] TNRM $
12 0622 10360 010731370000000003561620254354560000             D[AR] ROT[14.] DEST[AR] JUMP[MAPCOC] TNRM $
12 0623                                                         ; FORM SMALLEST ILLEGAL ADDRESS IN AR
12 0624
12 0625 10361 010731370000060543441623655754160000     MAPCOB: D[CONST 20] ROT[14.] DEST[AR] TNRM $
12 0626                                                         ;FULL 256K (NO ADR LIMIT!)
12 0627 10362 010730171400000000001623655354160000     MAPCOC: D[NLIT 600000000000] DEST[Q] TNRM $
12 0628                                                         ;PREPARE MASK OF MAPONE!ONETO1 BITS
12 0629 10363 010640170000060554161623660354160000             D[D%AMEM2] ALU[D&Q] DEST[Q] TNRM $
12 0630                                                         ;GET JUST THOSE BITS FROM D%AMEM2
12 0631 10364 010631170000060554161043654354160000             D[AR] ALU[DORQ] DEST[D%AMEM2] TNRM $
12 0632                                                         ;ENTER ENTIRE THING BACK INTO D%AMEM2 (REALLY JUST RESET
12 0632                                          ADR LMT)
12 0633
12 0634                                         .REPEAT 1 - WAITS [
12 0635
12 0636                                                 D[CONST PGR72] DEST[MA] TNRM $
12 0637                                                         ;LOAD MA FOR AGER!STATE PG
12 0638                                                 DF/WT D[MEM] DEST[Q] TNRM $
12 0639                                                         ;GET AGER!STATE PG IN Q
12 0640
12 0641                                                           ];.REPEAT 1 - WAITS
12 0642
12 0643                                         .repeat WAITS [ ;Setup user page table and use first page as address of
12 0643                                          ACs
12 0644
12 0645                                                 D[D%AMEM3] DEST[MA] TNRM $
12 0646                                                         ;LOAD MA FOR FETCH OF FIRST LOCATION OF USER PAGE TABLE
12 0647                                                 DFRQ D[CONST PGR72] DEST[MA] TNRM $
12 0648                                                         ;START FETCH OF UPT 1ST WD
12 0649                                                         ;LOAD MA FOR FETCH OF AGER!STATE PAGE
12 0650                                                 DFRQ D[MEM] MASK[27.] DEST[D%AMEM1] TNRM $
12 0651                                                         ;SAVE 1ST ENTRY OF USER PAGE TABLE IN D%AMEM1 SANS AGE
12 0652                                                         ;START FETCH OF PGR72
12 0653                                                 D[MEM] DEST[Q] TNRM $
12 0654                                                         ;GET AGER!STATE PAGE IN Q
12 0655
12m0656                                                       ];[       ;Setup user page table and use first page as
12m0656                                          address of ACs
12m0656
12m0656 10365 710731170000060554161623661754160000             D[D%AMEM3] DEST[MA] TNRM $
12m0656                                                         ;LOAD MA FOR FETCH OF FIRST LOCATION OF USER PAGE TABLE
12m0656 10366 710731170000060554165623655754160000             DFRQ D[CONST PGR72] DEST[MA] TNRM $
12m0656                                                         ;START FETCH OF UPT 1ST WD
```

```
12m0656 10370 01073017000006055416162365475516000              D[MEM] DEST[Q] TNRM $
12m0656                                                                ;GET AGER!STATE PAGE IN Q
12m0656
12 0656                                                              ].repeat WAITS
12 0657
12 0658
12 0659 10371 01064017177600000000162365535416000              D[NLIT 777000000000] ALU[D&Q] DEST[Q] TNRM $
12 0660                                                                ;GET JUST ISOLATED AND POSITIONED AGE IN Q
12 0661 10372 01063137000006055406762366075416000              D[D%AMEM1] MASK[27.] ALU[DORQ] DEST[AR] TNRM $
12 0662                                                                ;GET AGER!PREVIOUS CONTENTS IN AR
12 0663 10373 01073117000006055416102365435416000              D[AR] DEST[D%AMEM1] TNRM $
12 0664                                                                ;UPDATE AMEM WITH AGE!ACBASE
12 0665 10374 01073137000006055402762365475516000              D[MEM] MASK[11.] DEST[AR] TNRM $
12 0666                                                                ;Page for storing state (and AC's).
12 0667 10375 01073017000006054236162365435416000              D[AR] ROT[9.] DEST[Q] TNRM $
12 0668                                                                ;Make into an address.
12 0669 10376 01063117000000000141112225575456000              D[CONST 4] ROT[6.] ALU[DORQ] DEST[D%AMEMS] PUSHJ[USRCLR] TNRM $
12 0670                                                                ;Enter state/ac phys base adr in AMEM.
12 0671                                                                ;INVALIDATE USER AR'S
12 0672
12 0673                                                      .REPEAT 1 - WAITS [
12 0674                                                              PUSHJ[EXECLR] TNRM $
12 0675                                                                ;INVALIDATE EXEC AR'S
12 0676                                                                ];.REPEAT 1 - WAITS
12 0677
12 0678                                                      .REPEAT WAITS [ ;CONO PAG,10 does not invalidate EXEC AR's for WAITS
12 0679                                                              D[IR] ROT[35. - 3] MASK[1] COND[OBUS=0] PUSHJ[EXECLR] TNRM $
12m0680                                                                ];[     ;CONO PAG,10 does not invalidate EXEC AR's for
12m0680                                                       WAITS
12m0680 10377 01073100200000001000362225775456000              D[IR] ROT[35. - 3] MASK[1] COND[OBUS=0] PUSHJ[EXECLR] TNRM $
12 0680                                                                ].REPEAT WAITS
12 0681
12 0682 10400 01073117001206055416162365435416000      MAP-ENABLE TNRM $
12 0683 10401 01073117000000001416162025431456000              JUMP[MAIN] $
12 0684                                                                ;UNDO MAP-DISABLE AT START OF THINGS
12 0685                                                                ;DOES NOT NECESSARILY TURN MAP ON (CPU-MODE<B15> MAY BE
12 0685                                                       OFF)
12 0686                                                                ;FINIS OF COND 0
12 0687
12 0688
```

```
13 0689
13 0690
13 0691
13 0692                                         ;HERE FOR CONO 1
13 0693
13 0694                                         MAPCO1:
13 0695
13 0696                                         .REPEAT WAITS [
13 0697
13 0698                                                 PUSHJ[EXECLR] TNRM $
13 0699                                                         ;CLEAR EXEC ARS
13 0700
13m0701                                                       ];[
13m0701
13m0701 10402 010731170000000001416162225435456000             PUSHJ[EXECLR] TNRM $
13m0701                                                         ;CLEAR EXEC ARS
13m0701
13 0701                                                       ].REPEAT WAITS
13 0702
13 0703
13 0704                                         .REPEAT 1 - WAITS [
13 0705
13 0706                                                 PUSHJ[MEMADR-23BIT-ABS] $
13 0707                                                 SPEC[MAP-DISABLE] $
13 0708                                                 D[CONST N-CONO-1] DEST[MA] TNRM $
13 0709                                                         ;LOAD MA WITH MAGIC LOC, DISABLE MAP
13 0710                                                 DF/WT D[MEM] ALU[D+1] DEST[MEMSTO] PUSHJ[EXECLR] $
13 0711                                                         ;BUMP # CONO 1'S
13 0712                                                         ;CLEAR USER ARS
13 0713                                         MAP-ENABLE TNRM $
13 0714                                                           ];.REPEAT 1 - WAITS
13 0715
13 0716 10403 010731170000000001416162025431456000             JUMP[MAIN] $
13 0717                                                         ;ENABLE MAP
13 0718                                                         ;FINIS CONO 1
13 0719
13 0720
13 0721                                         ;HERE FOR CONO 3
13 0722
13 0723
13 0724                                         MAPCO3:
13 0725
13 0726                                         .REPEAT WAITS [
13 0727
13 0728                                                 PUSHJ[USRCLR] TNRM $
13 0729                                                         ;CLEAR USER ARS AS APPROPRIATE
13 0730
13m0731                                                       ];[
13m0731
13m0731 10404 010731170000000001416162225435456000             PUSHJ[USRCLR] TNRM $
13m0731                                                         ;CLEAR USER ARS AS APPROPRIATE
13m0731
13 0731                                                       ].REPEAT WAITS
13 0732
13 0733                                         .REPEAT 1 - WAITS [
13 0734
13 0735                                                 PUSHJ[MEMADR-23BIT-ABS] $
13 0736                                                 SPEC[MAP-DISABLE] $
13 0737                                                 D[CONST N-CONO-3] DEST[MA] TNRM $
13 0738                                                         ;LOAD MA WITH MAGIC LOC, DISABLE MAP
13 0739                                                 DF/WT D[MEM] ALU[D+1] DEST[MEMSTO] PUSHJ[USRCLR] $
13 0740                                                         ;BUMP # CONO 3'S
13 0741                                                         ;CLEAR EXEC ARS AS APPROPRIATE
13 0742                                         ;***                    ↑↑↑ User AR's???
13 0743                                         MAP-ENABLE TNRM $
13 0744
13 0745                                                           ];.REPEAT 1 - WAITS
13 0746
13 0747 10405 010731170000000001416162025431456000             JUMP[MAIN] $
13 0748                                                         ;ENABLE MAP
13 0749                                                         ;FINIS CONO 3
13 0750
13 0751
```

```
14 0752
14 0753
14 0754                                                 ;TURN OFF MAP
14 0755
14 0756 10406 010731170013060554161623654314160000     MAPOFF: MAP-DISABLE $
14 0757 10407 010730170013000314161623670354160000             SPEC[MAP-DISABLE] D[AMEM-ABS APR-MODE] DEST[Q] TNRM $
14 0758                                                                 ;Get current cpu status.
14 0759 10410 010651370000000001400362025575456000             D[CONST 1]  ALU[-D&Q] DEST[AR] JUMP[SETMODE] TNRM $
14 0760                                                                 ;Turn off bit and return.
14 0761
14 0762
14 0763                                                 ;HERE FOR COND 4 OR 5
14 0764
14 0765                                                 MAPC04:
14 0766 10411 010731170000210154161622254354560000     MAPC05: PUSHJ[MAPOFF] TNRM $
14 0767                                                                 ;SIMPLY TURN OFF MAP
14 0768 10412 540731170000060554161623254204160000             NEOI $
14 0769
14 0770
14 0771                                                 ;HERE FOR COND 6
14 0772
14 0773 10413 710731170000000002001623655354160000     MAPC06: D[LIT 1000] DEST[MA] TNRM $
14 0774                                                                 ;Ready for page 1.
14 0775 10414 010771170000000000000563655114170000             D[LIT 000000000000] ALU[NOTD] DEST[MAP-TAG] LONG $
14 0776                                                                 ;Disable comparator bit for page 1.
14 0777 10415 010731170000600002000543655114170000             D[LIT 300001000] DEST[MAP-MA] LONG $
14 0778                                                                 ;AND phys page adr of 1 (mapping 1:1)
14 0779                                                                 ; with full access (PREVENT bits off)
14 0780 10416 010770171400000000000162365535416000             D[NLIT 600000000000] ALU[NOTD] DEST[Q] TNRM $
14 0781                                                                 ;MASK OF MAPONE AND ONETOi FLAGS
14 0782 10417 010640170000060554161623661354160000             D[D%AMEMZ] ALU[D&Q] DEST[Q] TNRM $
14 0783                                                                 ;GET AMEMZ WITHOUT THOSE FLAGS
14 0784 10420 010631170400000000000104365535416000             D[LIT 200000000000] ALU[DORQ] DEST[D%AMEMZ] TNRM $
14 0785                                                                 ;CLEAR MAPONE AND SET ONETO1 FLAG
14 0786 10421 010730170000000314161623670354160000             D[AMEM-ABS APR-MODE] DEST[Q] TNRM $
14 0787                                                                 ;GET BITS, DO SPEC ENABLE
14 0788 10422 010631370000000001400362225575456000             D[CONST 1]  ALU[DORQ] DEST[AR] PUSHJ[SETMODE] TNRM $
14 0789                                                                 ;Turn on MAP ON bit
14 0790 10423 010731170000000014161620254314560000             JUMP[MAIN] $
14 0791                                                                 ;Return to macro-code.
14 0792
14 0793                                                 ;HERE FOR COND 7
14 0794
14 0795 10424 010731006000000000361620261214560000     MAPC07: D[D%AMEMZ] ROT[1] COND[SIGNOFF] JUMP[MAPC17] C550 $
14 0796                                                                 ;JUMP IF ONETO1 FLAG IS OFF
14 0797 10425 710731170000000002001623655354160000             D[LIT 1000] DEST[MA] TNRM $
14 0798                                                                 ;READY FOR PAGE 1
14 0799 10426 010771170100000000000563655114170000             D[LIT 040000000000] ALU[NOTD] DEST[MAP-TAG] LONG $
14 0800                                                                 ;CLOBBER MAP WD ASSOCIATED WITH PAGE 1
14 0801                                                                 ;(ENABLE COMPARATOR AND GIVE GARBAGE CONTEXT OF 0)
14 0802 10427 010770171400000000000162365535416000     MAPC17: D[NLIT 600000000000] ALU[NOTD] DEST[Q] TNRM $
14 0803                                                                 ;MASK OF MAPONE AND ONETO1 FLAGS
14 0804 10430 010640170000060554161623661354160000             D[D%AMEMZ] ALU[D&Q] DEST[Q] TNRM $
14 0805                                                                 ;GET AMEMZ WITHOUT THOSE FLAGS
14 0806 10431 010631171000000000000104365535416000             D[NLIT 400000000000] ALU[DORQ] DEST[D%AMEMZ] TNRM $
14 0807                                                                 ;SET MAPONE AND CLEAR ONETO1 FLAG
14 0808 10432 010730170000000314161623670354160000             D[AMEM-ABS APR-MODE] DEST[Q] TNRM $
14 0809                                                                 ;GET BITS, DO SPEC ENABLE
14 0810 10433 010631370000000001400362225575456000             D[CONST 1]  ALU[DORQ] DEST[AR] PUSHJ[SETMODE] TNRM $
14 0811                                                                 ;TURN ON MAP ENABLE BIT
14 0812 10434 540731170000060554161623254204160000             NEOI $
14 0813
14 0814
```

```
15 0815
15 0816
15 0817                                         ;INVALIDATE ALL USER MAP WORDS
15 0818                                         ;USER CONTEXTS ARE ODD
15 0819                                         ;ACCEPTS DEV-ADR/ MAPDEV
15 0820
15 0821 10435 010730002000000001416162026761456000     USRCLR: D[D%AMEM17] DEST[Q] COND[ZERO] JUMP[USRCL1] C550 $
15 0822                                                         ;GET COPY OF USER-CTXT IN Q AND JUMP IF ITS 0
15 0823 10436 010681370000060554005363655754160000             D[CONST 2] ALU[D+Q] DEST[AR D%AMEM17] TNRM $
15 0824                                                         ;BUMP OUR COPY OF USER CONTEXT
15 0825 10437 010731004000000005561620254214560000             D[AR] ROT[22.] COND[SIGNON] JUMP[USRCL1] C550 $
15 0826                                                         ;JUMP IF WRAPPING (USER-CTXT CAN IGNORE FORCE BIT)
15 0827 10440 010731170000060544160624254354170000             D[AR] ROT[35. - 19.] DEST[USER-CTXT] POPJ TNRM $
15 0828                                                         ;UPDATE USER-CTXT
15 0829                                                         ;RETURN, NOTHING TO DO
15 0830
15 0831 10441 010731170000060544002623655754170000     USRCL1: D[CONST 1] ROT[35. - 19.] DEST[USER-CTXT] TNRM $
15 0832                                                         ;SET USER-CTXT TO LOWEST VALID VALUE
15 0833                                                         ;USER CONTEXTS ARE ODD
15 0834 10442 010731170000060554003363655754160000             D[CONST 1] DEST[D%AMEM17] TNRM $
15 0835                                                         ;UPDATE OUR COPY OF USER-CTXT
15 0836 10443 010731370000060554161623646314160000             D[PC-FLAGS] DEST[AR] T300 $
15 0837 10444 010731170000060547402423655614170000             D[CONST 1] ROT[35. - 5] DEST[PC-FLAGS] T400 $
15 0838                                                         ;SELECT USER ADR SPACE
15 0839 10445 710241170000060554161623654354160000             ALU[0] DEST[MA] TNRM $
15 0840                                                         ;START AT MA = 0
15 0841 10446 010731170010060542202003655754170000             D[CONST 1] ROT[9. + LLOAD-ROT] DEST[LLOAD] TNRM $
15 0842                                                         ;512. CONSECUTIVE PAGES => TOUCH EVERY MAP WD (THANK U
15 0842                                                  HASHER)
15 0843 10447 010730070000211202Z03620255754560000             D[CONST 1] ROT[9.] DEST[Q] TNRM LOOP[. + 1] $
15 0844                                                         ;BUMP MA BY 1000 EACH TIME THROUGH LOOP
15 0845 10450 010771170100000000000563655114170000     USRCL2: D[LIT 040000000000] ALU[NOTD] DEST[MAP-TAG] LONG $
15 0846                                                         ;CLEAR A MAP WD (ENABLE COMPARATOR AND GARBAGE CONTEXT OF
15 0846                                                  0)
15 0847 10451 710601070000211214161620256714560000             D[MA] ALU[D+Q] DEST[MA] LOOP[USRCL2] $
15 0848                                                         ;LOOP OVER ALL USER MAP WDS
15 0849 10452 010731170000060554160424254214170000             D[AR] DEST[PC-FLAGS] T400 POPJ $
15 0850                                                         ;RETURN
15 0851
15 0852
```

```
16 0853
16 0854
16 0855                                         ;INVALIDATE ALL EXEC MAP WORDS AS APPROPRIATE
16 0856                                         ;CLEARS PAGE 1 IF IT IS BEING MAPPED (MAPONE=1)
16 0857                                         ;Exec contexts are even.
16 0858                                         ;Assumes DEV-ADR/ MAPDEV.
16 0859
16 0860 10453 0107300020000000014161620267214560000    EXECLR: D[D%AMEM16] DEST[Q] COND[ZERO] JUMP[EXECL1] CS50 $
16 0861                                                         ;Get copy of EXEC-CTXT in Q and jump if = 0.
16 0862 10454 0106013700000060554005343655754160000            D[CONST 2] ALU[D+Q] DEST[AR D%AMEM16] TNRM $
16 0863                                                         ;Bump software copy of exec context.
16 0864 10455 0107310060000000005361620254214560000            D[AR] ROT[21.] COND[SIGNOFF] CS50 JUMP[SET-CONTEXT] $
16 0865                                                         ;If not wrapping, go update hdwr context registers and
16 0865                                          return.
16 0866 10456 0107311700000605540053436557541600       EXECL1: D[CONST 2] DEST[D%AMEM16] TNRM $
16 0867                                                         ;Set exec ctxt to lowest valid value (exec contexts are
16 0867                                          EVEN).
16 0868 10457 7107311700000605541616236667541600               D[D%AMEM15] DEST[MA] TNRM $
16 0869                                                         ;START LOOP AT FIRST SWAPPING ADR
16 0870 10460 0107301700000605466236236667541600               D[D%AMEM15] ROT[27.] MASK[9.] DEST[Q] TNRM $
16 0871                                                         ;Q/ 1ST SWAPPING PAGE #
16 0872 10461 0116211700000000001777503655354160000            D[LIT 512. - 1] ALU[D-Q] DEST[HOLD] TNRM $
16 0873                                                         ;CALCULATE # PAGES TO DO
16 0874 10462 0107311700100605401600036547551700               D[MEM] ROT[LLOAD-ROT] DEST[LLOAD] TNRM $
16 0875                                                         ;511.-D%AMEM15 RES MON CONSECUTIVE PAGES => TOUCH EVERY
16 0875                                          MAP WD
16 0876 10463 0107301700000000020016236553541600               D[LIT 1000] DEST[Q] TNRM $
16 0877                                                         ;BUMP MA BY 1000 EACH TIME
16 0878 10464 0107711701000000000005636552141700       EXECL2: D[LIT 0400000000001] ALU[NOTD] DEST[MAP-TAG] CS50 $
16 0879                                                         ;Clear a map wd (enable comparator and garbage context of
16 0879                                          0)
16 0880 10465 7106010700002115141616202567145600               D[MA] ALU[D+Q] DEST[MA] LOOP[EXECL2] $
16 0881                                                         ;Loop over all EXEC map wds but the resident ones
16 0882 10466 0107300060000605541616242612141600               D[D%AMEM2] COND[SIGNOFF] DEST[Q] POPJ CS50 $
16 0883                                                         ;DONE IF NOT MAPPING PAGE 1
16 0884 10467 7107311700000000020016236553541600               D[LIT 1000] DEST[MA] TNRM $
16 0885                                                         ;SET UP FOR PAGE 1
16 0886 10470 0107711701000000000005636551141700               D[LIT 0400000000001] ALU[NOTD] DEST[MAP-TAG] LONG $
16 0887                                                         ;CLEAR MAP WD ASSOCIATED WITH PAGE 1
16 0888                                                         ;(ENABLE COMPARATOR AND GARBAGE CONTEXT OF 0)
16 0889 10471 0106511704000000000010402553145600               D[LIT 2000000000001] ALU[-D&Q] DEST[D%AMEM2] JUMP[SET-CONTEXT] $
16 0890                                                         ;Turn off ONETO1 flag, go update hardware context
16 0890                                          registers.
16 0891
16 0892
```

```
17 0893
17 0894
17 0895                                                 ;SET UP RESIDENT MONITOR PAGES 0,2:D%AMEM15-1 WITH 1:1 MAPPING (NO
17 0895                                                  MAPPING)
17 0896                                                 ;ACCEPTS DEV-ADR/ MAPDEV
17 0897
17 0898 10472 710241170000060554161623654374160000     FORCEM: ALU[0] DEST[MA] SHORT $
17 0899                                                                 ;READY FOR PAGE 0
17 0900                                                                 ;LET MAP-USER-SR SETTLE
17 0901 10473 010771170000000000000056365511417000             D[LIT 000000000000] ALU[NOTD] DEST[MAP-TAG] LONG $
17 0902                                                                 ;DISABLE COMPARATOR FOR PAGE 0
17 0903 10474 010731170000600000000543655114170000             D[LIT 000300000000] DEST[MAP-MA] LONG $
17 0904                                                                 ;AND PHYS PAGE ADR OF 0 (MAPPING 1:1)
17 0905                                                                 ;WITH FULL ACCESS (PREVENT BITS OFF)
17 0906 10475 010730170000060546623623666754160000             D[D%AMEM15] ROT[27.] MASK[9.] DEST[Q] TNRM $
17 0907                                                                 ;GET 1ST SWAPPING PAGE OF MONITOR
17 0908 10476 011611370000060554007623655754160000             D[CONST 2 + 1] ALU[Q-D] DEST[AR] TNRM $
17 0909 10477 010731170010060540160003654354170000             D[AR] ROT[LLOAD-ROT] DEST[LLOAD] TNRM $
17 0910                                                                 ;LOAD LOOP-CTR FOR PAGES 2:D%AMEM15-1
17 0911 10500 710731170000600004001623655354160000             D[LIT 300002000] DEST[MA] TNRM $
17 0912                                                                 ;Init MA to page 2, provide READ and WRITE PERMIT bits.
17 0913 10501 010730170000000002001623655354160000             D[LIT 1000] DEST[Q] TNRM $
17 0914                                                                 ;BUMP MA BY 1000 IN LOOP
17 0915 10502 010771170000000000000056365511417000     FORCE1: D[LIT 000000000000] ALU[NOTD] DEST[MAP-TAG] LONG $
17 0916                                                                 ;DISABLE COMPARATOR FOR PAGE
17 0917 10503 010731170000060554160543656514170000             D[MA] DEST[MAP-MA] LONG $
17 0918                                                                 ;SET MA TO PHY ADR WITH FULL ACCESS
17 0919 10504 710601070000212054161620256754560000             D[MA] ALU[D+Q] DEST[MA] LOOP[FORCE1] TNRM $
17 0920                                                                 ;LOOP
17 0921 10505 010731170000060554161624254354160000             POPJ TNRM $
17 0922                                                                 ;RETURN
17 0923
17 0924
```

```
18 0925
18 0926
18 0927                                                 ;Here from trap location
18 0928                                                 ;Save state in AMEM (hoping just for AR REFILL) and process trap.
18 0929                                                 ;MIPC + 1 is on JMEM stack courtesy of the PUSHJLC in the trap location.
18 0929
18 0930
18 0931 10506 010731170000005434103543643354160000     MAPTRP: D[MB-STATUS] MASK[33.]  DEST[AMEM-ABS STATE01] TNRM $
18 0932                                                         ;Save MB-STATUS03:35 (bit 00 used later as a flag)
18 0933
18 0934                                                  ;;Following is a bug trap. It should be left in forever.
18 0935 10507 010730570000000000000162365535416000             D[LABEL ILLTRP] DEST[ALU1 Q] TNRM $
18 0936                                                         ;Get illegal trap loc. in microcode
18 0937 10510 010660570000060551035627645354160000             D[JMEM] ROT[36. - MUA-ROT] MASK[14.] ALU[ALU1 D#Q] DEST[ALU1 Q]
                                                         TNRM $
18 0938 10511 010231402000212234161620254214560000             ALU[ALU1 Q] COND[ZERO] JUMP[.] C550 $
18 0939                                                         ;Hang if got trap at ILLTRP.
18 0940
18 0941 10512 010730570000060546207623643214160000             D[MB-STATUS] ROT[1 + 24.] MASK[3] DEST[ALU1 Q] C550 $
18 0942 10513 010661400000000014005620255614560000             D[CONST 2] ALU[ALU1 Q#D]   C550  -ZERO JUMP[MAPXFX] $
18 0943                                                         ;Jump if no FIXUP needed (bits 22:24 not = 010).
18 0944
18 0945 10514 010731170003060554161623654354160000             MU-POP TNRM $
18 0946                                                         ;Flush trap loc. from stack to expose loc. of fixup
                                                         routine.
18 0947 10515 010730570000060554161627645214160000             D[JMEM] DEST[ALU1 Q] C500 $
18 0948 10516 010631570000000000002007255514560000             D[CONST 1] ROT[MUA-ROT] ALU[ALU1 DORQ] LONG ODISP $
18 0949                                                         ;Call the loser's fixup routine. It will PUSHJ to
18 0950                                                         ; MAP-RET after restoring the MA to the failing address.
18 0950
18 0951                                                         ;This keeps the fixup routine adr on the stack !
18 0952                                                  ;A fix-up routine PUSHJs to here when done,
18 0953                                                  ; therefore its desired return adr is on stack.
18 0954 10517 010730570000060554161627645354160000     MAP-RET:D[JMEM] DEST[ALU1 Q] TNRM $
18 0955 10520 010601570000060540002043655754170000             D[CONST 1] ROT[MUA-ROT] ALU[ALU1 Q+D] DEST[JMEM] TNRM $
18 0956                                                         ;Compensate for decrementing of return address by MAPUTP.
18 0956
18 0957
18 0958 10521 010721170000060554000723655754170000     MAPXFX: ALU[-1] DEST[MERGE] TNRM $
18 0959                                                         ;Set up mapper to allow 23-bit absolute addressing.
18 0960 10522 010731170000005534161543654354160000             D[AR] DEST[AMEM-ABS STATE05] TNRM $
18 0961                                                         ;Save AR (needed below at MAPRD even in short case)
18 0962 10523 010730570000000110001623655354160000             D[LIT 000000044000] DEST[ALU1 Q] TNRM $
18 0963                                                         ;Mask of bits indicating dispatches for resuming (bits
                                                         21, 24)
18 0964 10524 010641402000000001416162024321456000             D[MB-STATUS] ALU[ALU1 D&Q] COND[ZERO] JUMP[MAPLNG] C550 $
18 0965                                                         ;If all off, do long save.
18 0966 10525 010730170000005434161623670354160000     MAPSHT: D[AMEM-ABS STATE01] DEST[Q] TNRM $
18 0967                                                         ;Get saved CTL!MB-STATUS
18 0968 10526 010631170000005430603543655754160000             D[CONST 1] ROT[35.] ALU[DORQ] DEST[AMEM-ABS STATE01] TNRM $
18 0969                                                         ;Say short state via B0 of CTL.
18 0970 10527 010731170000005454161543657754160000             D[IR] DEST[AMEM-ABS STATE02] TNRM $
18 0971                                                         ;Save IR
18 0972 10530 010731170000005474161543656754160000             D[MA] DEST[AMEM-ABS STATE03] TNRM $
18 0973                                                         ;Save MA
18 0974 10531 010730170000770000001623655354160000             D[LIT 000374000000] DEST[Q] TNRM $
18 0975                                                         ;Mask for just CPU-MODE bits
18 0976 10532 010640170000000030516162367035416000             D[AMEM-ABS APR-MODE] ROT[20.] ALU[D&Q] DEST[Q] TNRM $
18 0977                                                         ;Get just bits 10 thru 15 of CPU-MODE in Q.
18 0978 10533 010630170000005503613543646514160000             D[USER/INT] ROT[15.] MASK[5] ALU[DORQ] DEST[Q AMEM-ABS STATE04]
                                                         T400 $
18 0979                                                         ;Save CPU-MODE & MAP-USER-SR and ACTIVE bit...
18 0980 10534 010731170040060554002463655614170000             D[CONST MAPDEV] DEST[DEV-ADR] CLR-DEV-FROM-INTR C500 $
18 0981                                                         ;INSURE MAP DEVICE SELECTED
18 0982 10535 010231170000000041416124025435456000             ALU[Q] DEST[SAVE-MAP-SR] JUMP[MAPNTR] TNRM $
18 0983                                                         ;...and save no more state.
18 0984
18 0985 10536 010231170000005514161543654354160000     MAPLNG: ALU[Q] DEST[AMEM-ABS STATE04] TNRM $
18 0986                                                         ;Save Q
18 0987 10537 010731170000005454161543657754160000             D[IR] DEST[AMEM-ABS STATE02] TNRM $
18 0988                                                         ;Save IR
18 0989 10540 010731170000005474161543656754160000             D[MA] DEST[AMEM-ABS STATE03] TNRM $
18 0990                                                         ;Save MA
18 0991 10541 010731170020005574161543654755160000             SUPPRESS-FETCH-TRAPS D[MEM] DEST[AMEM-ABS STATE07] TNRM $
18 0992                                                         ;Store MEM (no recursive traps, please).
18 0993 10542 010730171777000000000263655354160000             SELECT-HOLD D[NLIT 777400000000] DEST[Q] TNRM $
18 0994                                                         ;Mask for AMEM-P0:9
18 0995 10543 010731170000005554161543654755160000             D[MEM] DEST[AMEM-ABS STATE06] TNRM $
18 0996                                                         ;Store HOLD
18 0997 10544 010641370000060545656162364435416000             D[AMEM-P] ROT[26. - AMEM-P-ROT] ALU[D&Q] DEST[AR] TNRM $
18 0998                                                         ;Get 'em positioned in AR
18 0999 10545 010730170000770000001623655354160000             D[LIT 000374000000] DEST[Q] TNRM $
18 1000                                                         ;Mask for just CPU-MODE bits
18 1001 10546 010640170000000030516162367035416000             D[AMEM-ABS APR-MODE] ROT[20.] ALU[D&Q] DEST[Q] TNRM $
18 1002                                                         ;Get just bits 10 thru 15 of CPU-MODE in Q.
18 1003 10547 010631370000060554161623654314160000             D[AR] ALU[DORQ] DEST[AR] $
18 1004                                                         ;Enter CPU-MODE bits into result.
18 1005 10550 010730170000000017407762365535416000             D[LIT 000000076037] DEST[Q] TNRM $
18 1006                                                         ;Mask for <DEV-ADR>B25!<MAP-USER-SR>B34!<ACTIVE>B35
18 1007 10551 010640170000006054376162364665416000             D[USER/INT] ROT[15.] ALU[D&Q] DEST[Q] T350 $
18 1008                                                         ;Keep just above masked.
18 1009 10552 010630170000005614161543654354160000             D[AR] ALU[DORQ] DEST[Q AMEM-ABS STATE10] TNRM $
18 1010                                                         ;STORE
18 1010                                                 <AMEM-P>B9!<LC>B10!<MBFF>B11!<CPU-MODE>B15!<UNUSED>B20!
18 1011                                                                 ; <DEV-ADR>B25!<SPARE=0>B30!<MAP-USER-SR>B34!
18 1012                                                                 ; <MAP-USER-SR-ACTIVE>B35
18 1013 10553 010731170000005634161543644754160000             D[LOOP-CTR] DEST[AMEM-ABS STATE11] TNRM $
18 1014                                                         ;Store LOOP COUNTER somewhere in STATE11 (assume it fills
18 1014                                                 word!)
18 1015 10554 010731170040060554002463655614170000             D[CONST MAPDEV] DEST[DEV-ADR] CLR-DEV-FROM-INTR C500 $
18 1016                                                         ;INSURE MAP DEVICE SELECTED
18 1017 10555 010231170000000041416124025435456000             ALU[Q] DEST[SAVE-MAP-SR] JUMP[MAPNTR] TNRM $
18 1018                                                         ;Now enter processing code.
18 1019
18 1020
```

```
19 1021
19 1022
19 1023                                         ;Restore state from AMEM only, resume execution of trapped code.
19 1024
19 1025
19 1026 10556 010731370000005434161623670354160000     MAPUTP: D[AMEM-ABS STATE01] DEST[AR] TNRM $
19 1027 10557 010731006000000014161620254214560000             D[AR] COND[SIGNOFF] JUMP[MAPULG] C550 $
19 1028                                                                 ;Jump if LONG RESTORE
19 1029
19 1030                                         ;SHORT STATE RESTORE
19 1031                                         ;Can only be here if going to DISPATCH to untrap rather than CONTINUE
19 1031                                          instr.
19 1032                                         ;THIS WILL CHANGE when store traps can do a short save !!!
19 1033
19 1034                                         ;Cases:  IF (ON IDISP+1), DF/WT ON IDISP+1, DFRQ+1 INSTR HAD MAPF[04],
19 1035
19 1036 10560 010731370000005504161623670354160000     MAPUSH: D[AMEM-ABS STATE04] ROT[36. - 20.] DEST[AR] TNRM $
19 1037 10561 010731170000000001416162225435456000             PUSHJ[SETMODE] TNRM $
19 1038                                                                 ;Restore CPU-MODE
19 1039 10562 610731170000005454161403670354160000             D[AMEM-ABS STATE02] DEST[IR-ALL] TNRM $
19 1040                                                                 ;Restore IR
19 1041 10563 010731002000000005003620254114560000             D[AR] ROT[20.] MASK[1] COND[ZERO] JUMP[MAPUS1] C600 $
19 1042                                                                 ;Skip loading MAP-USER-SR if was not active.
19 1043 10564 010731170000005510610403670354170000             D[AMEM-ABS STATE04] ROT[35.] MASK[4] DEST[MAP-USER-SR] TNRM $
19 1044                                                                 ;Restore MAP-USER-SR automagically making it active.
19 1045 10565 010731370000005434161623670354160000     MAPUS1: D[AMEM-ABS STATE01] DEST[AR] TNRM $
19 1046                                                                 ;Get MBSTATUS in AR.
19 1047 10566 010731170012060554161623654354160000             MAP-ENABLE TNRM $
19 1048                                                                 ;Enable mapping.
19 1049 10567 710721170003060554161463656354160000             D[PC] ALU[D-1] DEST[PC MA] MU-POP TNRM $
19 1050                                                                 ;Back up PC and load MA in case used, discard MIPC
19 1051 10570 010731004000000004161620656145600        0            D[CYTYPE] ROT[16.] C550 COND[SIGNON] JUMP[MAPUTI] $
19 1052                                                                 ;Consult flags for type of trap cycle, jump if instr
19 1052                                          fetch.
19 1053 10571 010731004000000005361620254214560000             D[AR] ROT[21.] COND[SIGNON] JUMP[MAPUTI] C550 $
19 1054                                                                 ;It was a DATA FETCH. jump if it wants to fetch instr
19 1054                                          again.
19 1055 10572 010731170000060554161503657754160000             D[IR] DEST[HOLD] TNRM $
19 1056                                                                 ;Data Fetch, does not want instr. refetched
19 1057 10573 540731170000060454161623254254160000             P-IDISP $
19 1058                                                                 ;Dispatch without instr fetch
19 1059 10574 010701170041606055415162365431416000     MAPUTI: ALU[D+0] SPEC[CLR-MAP-SR] $
19 1060                                                                 ;Leave temp. luser mode.. D+0 prevents CRY0 from being
19 1060                                          set !
19 1061 10575 010731170000060554161623654314160000             CONT $
19 1062 10576 550731112000076554161623254212160000             COND[-MA<20] ABORT-ADR-04 ID-COND[1]  MA-NEOI $
19 1063                                                                 ;Fetch instr. and dispatch.
19 1064                                                                 ;Trap to loc 04 if dispatching into AC's.
19 1065
19 1066
```

```
20 1067
20 1068
20 1069                                                 ;HERE FOR LONG RESTORE
20 1070
20 1071 10577 010731370000005604161623670354160000     MAPULG: D[AMEM-ABS STATE10] ROT[36. - 20.] DEST[AR] TNRM $
20 1072                                                         ;CPU-MODE BITS ALREADY POSITIONED IN STATE10
20 1073 10600 010731170000000014161622254354560000             PUSHJ[SETMODE] TNRM $
20 1074                                                         ;RESTORE CPU-MODE
20 1075                                                         ;Z INSTR BECAUSE OF FIELD CONFLICT
20 1076 10601 010730170000005634161623670354160000             D[AMEM-ABS STATE11] DEST[Q] TNRM $
20 1077 10602 010601170010060540002003655754170000             D[CONST 1] ROT[LLOAD-ROT] ALU[D+Q] DEST[LLOAD] TNRM $
20 1078                                                         ;Loop ctr. gets decremented (by hardware) during traps..
20 1078
20 1079                                                         ;RESTORE LOOP-CTR ALREADY POSITIONED IN STATE11
20 1080 10603 010731370000005614161623670354160000             D[AMEM-ABS STATE10] DEST[AR] TNRM $
20 1081 10604 010731002000000014003620254114560000             D[AR] MASK[1] COND[ZERO] JUMP[MAPUL1] C600 $
20 1082                                                         ;SKIP LOADING MAP-USER-SR IF WAS NOT ACTIVE
20 1083 10605 010731170000005610610403670354170000             D[AMEM-ABS STATE10] ROT[35.] MASK[4] DEST[MAP-USER-SR] TNRM $
20 1084                                                         ;RESTORE MAP-USER-SR AUTOMAGICALLY MAKING IT ACTIVE
20 1085 10606 010731170000005602560123670354170000     MAPUL1: D[AMEM-ABS STATE10] ROT[10. + AMEM-P-ROT]  DEST[AMEM-P] TNRM $
20 1086                                                         ;Restore AMEM-P
20 1087 10607 010731370000005534161623670354160000             D[AMEM-ABS STATE05] DEST[AR] TNRM $
20 1088                                                         ;Restore AR
20 1089 10610 010730170000005514161623670354160000             D[AMEM-ABS STATE04] DEST[Q] TNRM $
20 1090                                                         ;Restore Q
20 1091 10611 010731170012060554161623654354160000     MAP-ENABLE TNRM $
20 1092 10612 710731170012060554045623665754160000             MAP-ENABLE D[CYTYPE] MASK[18.] DEST[MA] TNRM $
20 1093                                                         ;Put failing adr into MA so we can re-do the operation
20 1093                                                  below.
20 1094 10613 010731170040005606412463670354170000             D[AMEM-ABS STATE10] ROT[26.] MASK[5] DEST[DEV-ADR] TNRM $
20 1095                                                         ;Restore DEV-ADR.  Can't depend on DEVICE AMEM now!!
20 1096 10614 640731170000000074161663657614160000             LOAD-PPC C550 $
20 1097                                                         ;Get the PPC loaded with mapped version of PC.
20 1098                                                           ;!!THIS DESTROYS THE IR !!
20 1099 10615 610731170000005454161403670354160000             D[AMEM-ABS STATE02] DEST[IR-ALL] TNRM $
20 1100                                                         ;Restore IR
20 1101 10616 010730570003060554161627645354160000             MU-POP D[JMEM] DEST[ALU1 Q] TNRM $
20 1102                                                         ;GET MIPC + 1
20 1103 10617 011611570002060540002043655754170000             D[CONST 1] ROT[MUA-ROT] ALU[ALU1 Q-D] DEST[JMEM] MU-PUSH TNRM $
20 1104                                                         ;Fixup MIPC and leave on stack for POPJ when resuming
20 1104                                                  execution
20 1105 10620 010731170000005434161503670354160000             D[AMEM-ABS STATE01] DEST[HOLD] TNRM $
20 1106                                                         ;Get MB-STATUS into HOLD
20 1107 10621 010731000000000001607620254615560000             D[MEM] ROT[1 + 6] MASK[3] COND[-ZERO] C550 JUMP[MAPUF1] $
20 1108                                                         ;Jump if status shows fetch-type map errors.
20 1109 10622 010731006000000000276162025461556000             D[MEM] ROT[11.] C550 COND[SIGNOFF] JUMP[MAPUTW1] $
20 1110                                                         ;Jump if no ECC error is indicated.
20 1111 10623 010731004000000000636162025461556000     MAPUF1: D[MEM] ROT[25.] C550 COND[SIGNON] JUMP[MAPUTW2] $
20 1112                                                         ;Jump if this was a write trap AND there was also a fetch
20 1112                                                  err.
20 1113                                                   ;Here to resume after a  FETCH failure (RMW too).
20 1114 10624 010731170000005554161503670354160000             D[AMEM-ABS STATE06] DEST[HOLD] TNRM $
20 1115                                                         ;Recover contents of HOLD.
20 1116 10625 010730570000005474161623670354160000             D[AMEM-ABS STATE03] DEST[ALU1 Q] TNRM $
20 1117                                                         ;Recover the saved MA, store temporarily in ALU1.
20 1118 10626 710231570000060554161624254354160000             DFRQ ALU[ALU1 Q] DEST[MA] TNRM POPJ $
20 1119                                                         ;Re-do the failing fetch, then restore the MA (which may
20 1119                                                  NOT
20 1120                                                         ; have contained the failing adr), and resume execution.
20 1120
20 1121                                                   ;Here to resume exectuion after a WRITE failure with no concurrent
20 1121                                                  fetch err.
20 1122 10627 010731170000005554161563670314160000     MAPUTW1:D[AMEM-ABS STATE06] DEST[MEMSTO] $
20 1123                                                         ;Re-do the WRITE that was trapped.
20 1124 10630 010731170000005574161503670354160000             D[AMEM-ABS STATE07] DEST[HOLD] TNRM $
20 1125                                                         ;Restore former contents of MEM bus.
20 1126 10631 010731170000060554161624254314160000     ILLTRP: NORM POPJ $
20 1127                                                         ;Return to MIPC saved on stack!
20 1128
20 1129                                                   ;Here if there was bad fetch status along with the write trap.  This
20 1129                                                  can
20 1130                                                   ; occur only when an IFRQ has been done before the write.  We restore
20 1130                                                  the
20 1131                                                   ; state of the world by re-doing both operations.
20 1132 10632 010731170000005554161563670304160000     MAPUTW2:IFRQ D[AMEM-ABS STATE06] DEST[MEMSTO] $
20 1133                                                         ;Re-do the WRITE that was trapped and the IFRQ preceeding
20 1133                                                  it.
20 1134 10633 010731170000060554161624254314160000             NORM POPJ $
20 1135
20 1136
20 1137
```

```
21 1138
21 1139
21 1140
21 1141                                                 ;SAVE AMEM STATE01 : STATE11 IN MEMORY AND CONTINUE STATE SAVE THERE
21 1142                                                 ;NECESSARY WHEN MACRO TRAP ABOUT TO BE TAKEN
21 1143                                                 ;MIPC ON STACK
21 1144
21 1145 10634 01073117001306055416162365431416000      MACTRP: MAP-DISABLE $
21 1146                                                 .REPEAT GRUNTSW [
21 1147                                                         SPEC[MAP-DISABLE] D[D%AMEM4] DEST[Q] TNRM $
21 1148                                                                 ;GET ADR OF PSB
21 1149                                                         D[LIT PSB567] ALU[DORQ] DEST[MA] TNRM $
21 1150                                                                 ;LOAD MA WITH ADR OF STATE PTR
21 1151                                                         DF/WT D[MEM] MASK[8.] ALU[D-1] DEST[AR Q] TNRM $
21 1152                                                                 ;LOAD Q AND AR WITH INDEX-1 INTO STATE PAGE TO USE
21 1153                                                                 ;IGNORE ALL BUT LOW 8 BITS FOR WRAP
21 1154                                                         D[LIT MMEMSTATESIZE] ALU[D+Q+1] DEST[AR] TNRM $
21 1155                                                                 ;Bump to pt to next block to use, Q/ current block - 1.
21 1156                                                         D[AR] MASK[8.] DEST[MEMSTO] $
21 1157                                                                 ;Mask to retain only 8 bits and write back to main mem.
21 1158                                                         D[D%AMEM5] ALU[D+Q] DEST[MA] TNRM $
21 1159                                                                 ;Load MA with adr-1 of State Block to start filling.
21 1160                                                                 ;ALL THIS "-1" CLEVERNESS TO MAKE 1 INST LOOP WORK RIGHT.
21 1160
21 1161                                                                 ;THIS BUMMED AN INSTR (D[MA] ALU[D-1] DEST[MA])
21 1162                                                         D[AMEM-ABS STATE01] DEST[AR] TNRM $
21 1163                                                         D[AR] COND[SIGNOFF] JUMP[MACLNG] C550 $
21 1164                                                                 ;JUMP IF LONG STATE SAVE IS DESIRED
21 1165                                                 MACSHT: D[AR] DEST[MEMSTO] MA+MA+1 $
21 1166                                                                 ;SHORT!  JUST SAVE CTL!MB-STATUS
21 1167                                                         D[AMEM-ABS STATE02] DEST[MEMSTO] MA+MA+1 $
21 1168                                                                 ;AND IR
21 1169                                                         D[AMEM-ABS STATE03] DEST[MEMSTO] MA+MA+1 $
21 1170                                                                 ;AND MA
21 1171                                                         D[AMEM-ABS STATE04] DEST[MEMSTO] MA+MA+1 $
21 1172                                                                 ;AND CPU-MODE & MAP-USER-SR ETC.
21 1173                                                         D[PC-FLAGS] MASK[LEFT] DEST[Q] T350 $
21 1174                                                                 ;SAVE PC FLAGS IN LEFT HALF
21 1175                                                         D[JMEM] MU-POP ROT[36. - MUA-ROT] MASK[14.] ALU[DORQ]
21 1175                                                  DEST[MEMSTO] MA+MA+1 $
21 1176                                                                 ;POP JMEM STACK GETTING MIPC
21 1177                                                                 ;WRITE <PC-FLAGS>B17!<MIPC>B35
21 1178                                                         D[PC] DEST[MEMSTO] MA+MA+1 JUMP[MACTR2] $
21 1179                                                                 ;WRITE PC
21 1180                                                                 ;AND JUMP (SHORT MEM BLOCK DONE)
21 1181
21 1182
21 1183                                                 ;HERE TO POP AMEM STATE INTO MEM AND CONTINUE LONG STATE SAVE
21 1184
21 1185                                                 MACLNG: D[LIT STATE01] DEST[AR] TNRM $
21 1186                                                         D[AR] ROT[AMEM-P-ROT] DEST[AMEM-P] TNRM $
21 1187                                                                 ;LOAD AMEM-P WITH START OF AMEM STATE BLOCK
21 1188                                                         D[CONST AMEMSTATESIZE - 1] ROT[LLOAD-ROT] DEST[LLOAD] TNRM $
21 1189                                                                 ;LOAD LOOP-CTR FOR FULL AMEM BLOCK DUMP
21 1190                                                         D[AMEM&P] DEST[MEMSTO] AMEM-P-INCR MA+MA+1 LOOP[.] $
21 1191                                                                 ;PULL FROM AMEM STATE BLOCK TO MAIN MEM STATE BLOCK
21 1192                                                         D[PC-FLAGS] MASK[LEFT] DEST[Q] T350 $
21 1193                                                                 ;SAVE PC FLAGS IN LEFT HALF
21 1194                                                         D[PC] MASK[18.] ALU[DORQ] DEST[MEMSTO] MA+MA+1 TNRM $
21 1195                                                                 ;WRITE PC
21 1196                                                         D[JMEM-P] DEST[MEMSTO] MA+MA+1 $
21 1197                                                                 ;WRITE POSITIONED JMEM-P (THE WAY IT WAS BEFORE TRAP)
21 1198                                                 MACTR1:
21 1199                                                   .repeat JMEMSIZE [
21 1200                                                         MA+MA+1 D[JMEM] MU-POP DEST[MEMSTO] $
21 1201                                                                 ;POP LAST JEMEMSIZE STACK LOCS INTO STATE BLOCK
21 1202                                                       ]
21 1203                                                   ];.repeat GRUNTSW
21 1204 10635 01073037000006055416162364631416000      MACTR2: D[PC-FLAGS] DEST[Q AR] T300 $
21 1205                                                                 ;Save flags in AR for trap code.
21 1206 10636 01065117002000000000042365521417000              D[LIT 010000000000] ALU[-D&Q] DEST[PC-FLAGS] T400 $
21 1207                                                                 ;Put machine in exec mode.
21 1208 10637 01073117001406055400040365575417000              REENABLE-TRAP1 SET-TEMP-EXEC  [ D[CONST 00] DEST[MAP-USER-SR]
21 1208                                                  ]TNRM $
21 1209                                                                 ;Let traps happen again, and make sure about the mode.
21 1210 10640 71073117000006055416162365575416000              D[CONST PGR70] DEST[MA] TNRM $
21 1211                                                                 ;Load MA with adr of trap instr
21 1212 10641 01024117000006055416072365431217000              DFRQ  ALU[0] DEST[MERGE] $
21 1213 10642 01073117001206055416162365431416000      MAP-ENABLE $
21 1214 10643 01073117000000001416162025431456000      JUMP[PGRTRP] $                                  . . .#
21 1215                                                                 ;Fetch the trap instr, enable mapping with 18-bit
21 1216                                                                 ;  virtual addresses, and enter trap code.
21 1217
21 1218
```

```
22 1219
22 1220
22 1221                                                 ;Restore STATE FROM BLOCK IN MEMORY
22 1222                                                 ;AND JUMP INTO TRAP UCODE JUST LIKE TRAP JUST HAPPENED
22 1223                                                 ;MAP BLKI
22 1224
22 1225                                                 MAPBKI:
22 1226                                                 .REPEAT 1 - GRUNTSW [
22 1227                                                         ILGIOT $
22 1228                                                                 ;Perhaps this should be a no-op, so it can be followed by
22 1228
22 1229                                                                 ;JRST 2,@PGRTRP so the same monitor can be run for both.
22 1229
22m1230                                                    ];[
22m1230 10644 540731170000060554161623254204160000              ILGIOT $
22m1230                                                                 ;Perhaps this should be a no-op, so it can be followed by
22m1230
22m1230                                                                 ;JRST 2,@PGRTRP so the same monitor can be run for both.
22m1230
22 1230                                                    ].REPEAT 1 - GRUNTSW
22 1231                                                 .REPEAT GRUNTSW [
22 1232                                                 MACUTP: ALU[-1] DEST[MERGE] TNRM $
22 1233                                                                 ;Set up mapper to allow extended absolute addressing.
22 1234                                                         D[CONST MAPDEV] DEST[DEV-ADR] CLR-DEV-FROM-INTR CS00 $
22 1235                                                                 ;INSURE MAP DEVICE SELECTED
22 1236                                                 .REPEAT 1 - WAITSFIX [
22 1237                                                         MAP-DISABLE $
22 1238                                                 ];.REPEAT 1 - WAITSFIX
22 1239                                                         SPEC[MAP-DISABLE] D[D%AMEM4] DEST[Q] TNRM $
22 1240                                                                 ;GET ADR OF PSB
22 1241                                                         D[LIT PSB567] ALU[DORQ] DEST[MA] TNRM $
22 1242                                                                 ;LOAD MA WITH ADR OF STATE PTR
22 1243                                                         DF/WT D[MEM] MASK[8.] DEST[Q] TNRM $
22 1244                                                                 ;LOAD Q WITH INDEX OF NEXT STATE BLOCK TO BE USED
22 1245                                                                 ;IGNORE ALL BUT LOW 8 BITS FOR WRAP
22 1246                                                         D[LIT MMEMSTATESIZE] ALU[Q-D] DEST[AR] TNRM $
22 1247                                                                 ;BACKUP TO PT TO THE BLOCK OF CONCERN (NEXT TO BE FILLED
22 1247                                                  TOO)
22 1248                                                         D[AR] MASK[8.] DEST[MEMSTO Q] $
22 1249                                                                 ;WRITE CLEAN INDEX TO MAIN MEM AND Q
22 1250                                                         D[D%AMEM5] ALU[DORQ] DEST[MA] TNRM $
22 1251                                                                 ;FORM ADR OF START OF BLOCK TO RESTORE FROM
22 1252                                                         DF/WT D[MEM] DEST[AR AMEM-ABS STATE01] MA+MA+1 $
22 1253                                                                 ;Read and store CTL!MBSTATUS into AMEM block.
22 1254                                                         DFRQ MA+MA+1 D[AR] COND[SIGNOFF] JUMP[MACULG] CS50 $
22 1255                                                                 ;Fetch 2nd word of state block. Jump if long restore.
22 1256                                                 MACUSH: DFRQ D[MEM] DEST[AMEM-ABS STATE02] MA+MA+1 $
22 1257                                                                 ;Restore IR to AMEM.
22 1258                                                         DFRQ D[MEM] DEST[AMEM-ABS STATE03] MA+MA+1 $
22 1259                                                                 ;Restore MA to AMEM.
22 1260                                                         DFRQ D[MEM] DEST[AMEM-ABS STATE04 AR] MA+MA+1 $
22 1261                                                                 ;Restore CPU-MODE & MAP-USER-SR etc to AMEM. and AR.
22 1262                                                         D[MEM] MASK[LEFT] DEST[PC-FLAGS]  T350 PUSHJ[USRRST] $
22 1263                                                                 ;RESTORE PC-FLAGS, call routine to restore Map-User-Sr.
22 1264                                                         DFRQ D[MEM] ROT[MUA-ROT] MU-PUSH DEST[JMEM] $
22 1265                                                                 ;Push MIPC onto stack (JMEM-P not important).
22 1266                                                         D[MEM] DEST[PC] TNRM $
22 1267                                                                 ;Restore MACRO PC.
22 1268                                                         D[AMEM-ABS STATE03] DEST[MA] TNRM $
22 1269                                                         D[AMEM-ABS STATE02] DEST[IR-ALL] TNRM $
22 1270                                                         JUMP[MAPNTR] TNRM $
22 1271                                                           ;Enter mapping algorithm as if we had just trapped...
22 1272
22 1273                                                 ;HERE TO RESTORE LONG MEM STATE BLOCK
22 1274
22 1275                                                 MACULG:
22 1276                                                 ;;;     D[LIT STATE02] DEST[AR] TNRM $
22 1277                                                 ;;;             ;START FILL FROM STATE02 (STATE01 DONE ALREADY)
22 1278                                                 ;;;     D[AR] ROT[AMEM-P-ROT] DEST[AMEM-P] TNRM $
22 1279                                                 ;;;             ;LOAD AMEM-P WITH START OF AMEM STATE BLOCK
22 1280                                                 ;;;     D[CONST AMEMSTATESIZE - 1 - 1] ROT[LLOAD-ROT] DEST[LLOAD] TNRM $
22 1280
22 1281                                                 ;;;             ;LOAD LOOP-CTR FOR FULL-1 AMEM BLOCK FILL
22 1282                                                 ;;;     DF/WT D[MEM] DEST[AMEM@P] AMEM-P-INCR MA+MA+1 LOOP[.] $
22 1283                                                 ;;;             ;WAIT FOR FETCH OF STATE WORD AND STORE IT IN AMEM BLOCK
22 1283
22 1284                                                 ;;;             ;INCREMENT AM AND AMEM-P AS WE GO
22 1285
22 1286                                                         DFRQ D[MEM] DEST[AMEM-ABS STATE02] MA+MA+1 $
22 1287                                                           ;Fetch 3rd word, store 2nd in AMEM.
22 1288                                                         DFRQ D[MEM] DEST[AMEM-ABS STATE03] MA+MA+1 $
22 1289                                                           ;Etc.
22 1290                                                         DFRQ D[MEM] DEST[AMEM-ABS STATE04] MA+MA+1 $
22 1291                                                         DFRQ D[MEM] DEST[AMEM-ABS STATE05] MA+MA+1 $
22 1292                                                         DFRQ D[MEM] DEST[AMEM-ABS STATE06] MA+MA+1 $
22 1293                                                         DFRQ D[MEM] DEST[AMEM-ABS STATE07] MA+MA+1 $
22 1294                                                         DFRQ D[MEM] DEST[AMEM-ABS STATE10 AR] MA+MA+1 $
22 1295                                                           ;Place copy in AR for call to USRRST.
22 1296                                                         DFRQ D[MEM] DEST[AMEM-ABS STATE11] MA+MA+1 $
22 1297                                                           ;Fetch 12th word (= PC-FLAGS,,PC)
22 1298                                                         D[MEM] DEST[PC] TNRM PUSHJ[USRRST] $
22 1299                                                                 ;Restore PC, call routine to restore Map-User-Sr.
22 1300                                                         DFRQ MA+MA+1 D[MEM] MASK[LEFT] DEST[PC-FLAGS]  T350 $
22 1301                                                                 ;Fetch 13th word (JMEM-P).
22 1302                                                         DFRQ MA+MA+1  D[MEM] DEST[JMEM-P Q] TNRM $
22 1303                                                           ;Fetch top word of saved micro-pdl. Restore JMEM-P, save copy.
22 1303
22 1304                                                         DFRQ MA+MA+1 D[MEM] DEST[JMEM] $
22 1305                                                           ;Fetch next micro-pdl word.  Restore first one.
22 1306                                                  .repeat JMEMSIZE - 2 [
22 1307                                                         DFRQ MA+MA+1  D[MEM] MU-POP DEST[JMEM] $
22 1308                                                           ;Fetch next word of saved micro-pdl. Back up JMEM-P and write
22 1309                                                           ; the previous word into JMEM.  Note that we are, in effect,
22 1310                                                           ; PUSHing backwards !
22 1311                                                    ]
22 1312                                                         D[MEM] MU-POP DEST[JMEM] $
22 1313                                                           ;Last saved mu-pdl word. We repair JMEM-P later.
22 1314                                                         D[AMEM-ABS STATE02] DEST[IR-ALL] TNRM $
22 1315                                                         D[AMEM-ABS STATE03] DEST[MA] TNRM $
22 1316                                                         D[AMEM-ABS STATE06] DEST[HOLD] TNRM $
```

```
22 1322                                                                 ;NOW JUMP INTO PROCESSING CODE JUST LIKE WE JUST GOT THE
22 1322                                                          TRAP
22 1323
22 1324                                                                 1:.REPEAT GRUNTSW
22 1325
22 1326
22 1327 10645 010731170000060554161Z4365435416000      USRRST: D[AR] DEST[SAVE-MAP-SR] TNRM $
22 1328                                                           ;The "AR loading" code needs this.
22 1329 10646 010731002000060554003624254214160000             D[AR] MASK[1] COND[ZERO] POPJ C550 $
22 1330                                                                 ;Skip loading MAP-USER-SR if was not active.
22 1331 10647 010731170000060550610404254214170000             D[AR] ROT[35.] MASK[4] DEST[MAP-USER-SR] C550 POPJ $
22 1332                                                                 ;Restore MAP-USER-SR automagically making it active.
22 1333
22 1334
```

```
23 1335
23 1336                                                 TRAP-ENTER: ;Puts trap type code in ALU1 Q and AR, failing address in MA.
23 1336
23 1337                                                             ; Call with MB-STATUS in AR, saved AR in STATE05.
23 1338 10650 0107Z100Z000000006405620Z5411456000              D[AR] ROT[1 + Z5.] MASK[2] ALU[D-1] LONG COND[ZERO] JUMP[MAPWR] $
23 1338
23 1339                                                               ;If bit Z5 is on and bit 24 is off, this is a WRITE trap.
23 1340 10651 0107310060000000021616202542145600O             D[AR] ROT[8.] COND[SIGNOFF] C550 JUMP[MAPIF] $
23 1341                                                               ;Not a Write. If cycle not started by DFRQ, itis an instr.
23 1341                                                      fetch.
23 1342 10652 0107310060000000017616202542145600O             D[AR] ROT[7.] COND[SIGNOFF] JUMP[MAPRD] C550 $
23 1343                                                               ;DFRQ. Definitely a Data Fetch if MI MEM WAIT was on during
23 1343                                                      fetch.
23 1344 10653 0107310060000000061616202542145600O             D[AR] ROT[24.] COND[SIGNOFF] JUMP[MAPRD] C550 $
23 1345                                                               ;No WAIT. If trap not from 1st uinst after IDISP, still a Data
23 1345                                                      Fetch.
23 1346 10654 7107Z11700000605541616236563541600O      MAPIF:  D[PC] ALU[D-1] DEST[MA] TNRM $
23 1347                                                               ;PC-1 is failing adr for instr fetch
23 1348 10655 0107307700000605444056242557541600O             D[CONST 2] ROT[18.] DEST[ALU1 Q AR] POPJ TNRM $
23 1349                                                               ;Instruction fetch trap (Q/ 2,,0 ie. XCT access)
23 1350
23 1351 10656 0107310060000000055616202542145600O      MAPRD:  D[AR] ROT[22.] COND[SIGNOFF] C550 JUMP[MAPRD2] $
23 1352                                                               ;Jump if not special case trap.
23 1353 10657 0107310040000000057616202542145600O             D[AR] ROT[23.] COND[SIGNON] C550 JUMP[MAPRDA] $
23 1354                                                               ;Jump if failing address is  claimed to be in the AR.
23 1355 10660 7107311700000000140456202577541560O             D[IR] MASK[18.] DEST[MA] TNRM JUMP[MAPRD2] $
23 1356                                                               ;IR, not MA, contains failing adr.
23 1357 10661 7107311700000005534161623670354160O      MAPRDA: D[AMEM-ABS STATE05] DEST[MA] TNRM CONT $
23 1358                                                               ;In AR (DF-FROM-AR in the cycle after the DF)
23 1359 10662 0107310040000000051616202542145600O      MAPRD2: D[AR] ROT[20.] COND[SIGNON] JUMP[MAPRMW] C550 $
23 1360                                                               ;Jump if READ-MODIFY-WRITE
23 1361 10663 0107307700000605444216242557541600O             D[CONST 10] ROT[18.] DEST[ALU1 Q AR] POPJ TNRM $
23 1362                                                               ;Simple fetch trap (Q/ 10,,0 ie. READ ACCESS)
23 1363 10664 0107307700000605444316242557541600O      MAPRMW: D[CONST 14] ROT[18.] DEST[ALU1 Q AR] POPJ TNRM $
23 1364                                                               ;Read Modify Write trap (Q/ 14,,0 ie. READ&WRT ACCESS)
23 1365
23 1366                                                 MAPWR:  ;On a WRITE, MA is failing adr for store.
23 1367 10665 0107307700000605444102636557141600O             SELECT-HOLD D[CONST 4] ROT[18.] DEST[ALU1 Q AR] $
23 1368                                                               ;Select the HOLD register as the MEM data.
23 1369                                                               ;Get code for SIMPLE WRITE TRAP (Q/ 4,,0 IE. WRT ACCESS)
23 1369
23 1370 10666 0107311700000055541615436547551600O             D[MEM] DEST[AMEM-ABS STATE06] TNRM $
23 1371                                                               ;Get store data to STATE, even if we have done a short
23 1371                                                      save.
23 1372 10667 0107311700000605541616242543541600O             POPJ TNRM $
23 1373
```

```
24 1374                                                 ;HANDLE AR REFILL OR PAGE FAULT
24 1375                                                 ;
24 1376                                                 ;EXPECTS THAT LEVEL 1 STATE (AMEM) IS SAVED AND LEVEL 2 STUFF NOT YET
24 1377                                                  DAMAGED
24 1377                                                 ;MIPC OF TRAP ON LAST JMEM STACK LOC
24 1378                                                 ; &&&  MA must still contain original value &&&
24 1379
24 1380
24 1381 10670 01073137000000054341616236703541600      MAPNTR: D[AMEM-ABS STATE01] DEST[AR] TNRM $
24 1382 10671 01073117000021521416162225431456000              NORM PUSHJ[TRAP-ENTER] $
24 1383                                                                 ;Get trap type code and failing MA.
24 1384
24 1385                                                   ;Here with trap type in left half ALU1 Q (coded bits)
24 1386                                                   ;MA contains failing adr.
24 1387 10672 01063172400000000140452602566145600              D[MA] MASK[18.] ALU[ALU1 DORQ] DEST[AR CYTYPE] COND[USER] C550
24 1387                                                  JUMP[MFUS] $
24 1388                                                                 ;AR has <type code bits,,failing MA>
24 1389                                                                 ;Jump if USER
24 1390 10673 01073017000006054500562365675416000              D[MA] ROT[20.] MASK[2] DEST[Q] TNRM $
24 1391                                                                 ;high order 2 bits
24 1392 10674 0116Z100200000001400762025551456000              D[CONST 3] ALU[D-Q] COND[ZERO] JUMP[MFHIEX] C600 $
24 1393                                                                 ;J if privately mapped part of exec addrs space.
24 1394 10675 01073017000000000220762025754560000              D[CONST MMAP / 1000] ROT[9.]  DEST[Q] JUMP[MFA1] TNRM $
24 1395                                                                 ;PAGE TAB # MMAP AND JUMP
24 1396
24 1397 10676 01073017000006055410162366135416000      MFUS:   D[D%AMEM2] MASK[32.] DEST[Q] TNRM $
24 1398                                                                 ;GET ADR LIMIT REG
24 1399                                                 ;;;     D[MA] MASK[18.] ALU[D-Q] COND[SIGNOFF] JUMP[MTRPAL] C600 $
24 1400                                                                 ;J IF MA PAST ADDRS LIMIT
24 1401 10677 01073017000000001416162026175456000              D[D%AMEM3] DEST[Q] JUMP[MFA1] TNRM $
24 1402                                                                 ;GET UPT ADR IN Q
24 1403
24 1404 10700 01073017000006055416162366235416000      MFHIEX: D[D%AMEM4] DEST[Q] TNRM $
24 1405                                                                 ;GET PSB ADR IN Q
24 1406 10701 01073117001306055416162365431416000      MFA1:   MAP-DISABLE $
24 1407 10702 71060117001306054662362365675416000              SPEC[MAP-DISABLE] D[MA] ROT[27.] MASK[9.] ALU[D+Q] DEST[MA] TNRM
24 1407                                                  $
24 1408                                                                 ;PAGE TABLE IN Q
24 1409                                                                 ;LOAD MA WITH ADR OF PT
24 1410 10703 61073117000006054303544365575416000              D[CONST 16] ROT[12.] DEST[IR-ADR] TNRM $
24 1411                                                                 ;PREPARE INITIAL PERMIT BITS
24 1412 10704 01073017001406055400762365575416000      MFA2:   REENABLE-TRAP1 D[CONST 3] DEST[Q] TNRM $
24 1413                                                                 ;FETCH PAGE TABLE ENTRY
24 1414                                                                 ;DON'T CLEAR IND PNTR. COUNT, WAIT FOR FETCH
24 1415 10705 03063017000006054456162365475316000              DF/WT D[MEM] ROT[18.] ALU[DORQ] DEST[Q] TNRM $
24 1416                                                                 ;GET PERMIT BITS
24 1417 10706 61064117000006055416144365775416000              D[IR] ALU[D&Q] DEST[IR-ADR] TNRM $
24 1418                                                                 ;AND THEM IN
24 1419 10707 01073100000000000260762025461556000              D[MEM] ROT[11.] MASK[3] COND[-ZERO] JUMP[MFTR1] C550 $
24 1420                                                                 ;CHECK FOR TRAP BITS AND JUMP IF SO
24 1421 10710 01073100200000000320362025461556000      MFB3:   D[MEM] ROT[13.] MASK[1] COND[ZERO] JUMP[MFTR2] C550 $
24 1422                                                                 ;J IF NO ACCESS PERMIT
24 1423 10711 01073000200000000040562025461556000              D[MEM] ROT[2] MASK[2] DEST[Q] COND[ZERO] JUMP[MFTYP0] C550 $
24 1424                                                                 ;GET TYPE CODE, J IF 0 <PRIVATE>
24 1425 10712 01021000200000001416162025421456000              ALU[Q-1] DEST[Q] COND[ZERO] JUMP[MFTYP1] C550 $
24 1426                                                                 ;J IF 1 <SHARED>
24 1427 10713 01021000000000001416162025421456000              ALU[Q-1] DEST[Q] COND[-ZERO] JUMP[MFTR3] C550 $
24 1428                                                                 ;J IF NOT 2 <ILLEGAL=3>
24 1429 10714 01073100000000001060362025761456000              D[IR] ROT[35.] MASK[1] COND[-ZERO] JUMP[MFTR4] C550 $
24 1430                                                                 ;TYPE 2 <INDIRECT>, J IF >2 INDR. PNTRS
24 1431 10715 61170117000006055416144365775416000              D[IR] ALU[D+1] DEST[IR-ADR] TNRM $
24 1432                                                                 ;COUNT IND. PNTRS
24 1433 10716 01073017000006054663362365475516000              D[MEM] ROT[27.] MASK[13.] DEST[Q] TNRM $
24 1434                                                                 ;GET PAGE TABLE #
24 1435 10717 01073117000006055416130365475516000              D[MEM] DEST[D%AMEM14] TNRM $
24 1436                                                                 ;SAVE PNTR IN D%AMEM
24 1437 10720 71060117000000004000162365535416000              D[LIT SPT] ALU[D+Q] DEST[MA] TNRM $
24 1438                                                                 ;FETCH PAGE TABLE PNTR FROM SPT
24 1439 10721 01073017000006055402362366635216000              DFRQ D[D%AMEM14] MASK[9.] DEST[Q] TNRM $
24 1440                                                                 ;GET PAGE #, START MEM FETCH
24 1441 10722 01073100000000000441162025461556000              D[MEM] ROT[18.] MASK[4] COND[-ZERO] JUMP[MFTR4] C550 $
24 1442                                                                 ;PAGE OUT-OF-CORE? J IF YES (TRAP)
24 1443 10723 01073117000006055403550365475516000              D[MEM] MASK[14.] DEST[HOLD] TNRM $
24 1444                                                                 ;NO, GET MEM PAGE # IN HOLD (NOTICE 14.!!) ***
24 1445 10724 71063117000021610236162025475556000              D[MEM] ROT[9.] ALU[DORQ] DEST[MA] JUMP[MFA2] TNRM $
24 1446                                                                 ;FORM MA FOR OF PTR AND TAKE INDIRECT LOOP JUMP
24 1447                                                                 ;MFA2 DOES DFRQ
24 1448
24 1449
```

```
25 1450
25 1451
25 1452                                                 ;HERE FOR SHARED PTR
25 1453
25 1454 10725 010730170000060546633623654755160000     MFTYP1: D[MEM] ROT[27.] MASK[13.] DEST[Q] TNRM $
25 1455                                                         ;GET SHARED PAGE #
25 1456 10726 710601170000000040001623655354160000             D[LIT SPT] ALU[D+Q] DEST[MA] TNRM $
25 1457                                                         ;SHARED PNTR FROM SPT
25 1458 10727 010730170000000003025520255752560000             DFRQ D[CONST 12] ROT[12.] DEST[Q] JUMP[MFTY00] TNRM $
25 1459                                                         ;FETCH PTR, BETTER BE PRIVATE TYPE!!
25 1460                                                         ;GET R-X MASK
25 1461
25 1462
25 1463                                                 ;HERE FOR PRIVATE PTR
25 1464                                                 ;MEM AVAILABLE
25 1465
25 1466 10730 010730170000060543025623655754160000     MFTYP0: D[CONST 12] ROT[12.] DEST[Q] TNRM $
25 1467                                                         ;GET R-X MASK
25 1468 10731 010641002000000001416162025761456000     MFTY00: D[IR] ALU[D&Q] COND[ZERO] JUMP[MFTY01] CS50 $
25 1469                                                         ;J IF NEITHER R NOR X
25 1470 10732 610631170000060554161443657754160000             D[IR] ALU[DORQ] DEST[IR-ADR] TNRM $
25 1471                                                         ;TURN ON R AND X
25 1472 10733 010770170000060545607623657754160000     MFTY01: D[IR] ROT[23.] MASK[3] ALU[NOTD] DEST[Q] TNRM $
25 1473                                                         ;GET RWX ENBL BITS, INVERTED
25 1474 10734 010731000000000044116202546155600        0       D[MEM] ROT[18.] MASK[4] COND[-ZERO] JUMP[MFTRS] CS50 $
25 1475                                                         ;J IF OUT-OF-CORE (TRAP)
25 1476 10735 010640000000000042076202542145600        0       D[AR] ROT[17.] MASK[3] ALU[D&Q] DEST[Q] COND[-ZERO] JUMP[MFTR6]
 CS50 $
25 1477                                                         ;J IF ILLEGAL ACCESS TYPE (RWX)
25 1478 10736 010730170000060554035303654755160000             D[MEM] MASK[14.] DEST[Q D%AMEM14] TNRM $
25 1479                                                         ;GET ABS PAGE # (NOTICE 14.!!) ***
25 1480                                                         ;SAVE PHYS PAGE # IN SCRATCH
25 1481 10737 710631170000000001000162365535416000             D[LIT CST0] ALU[DORQ] DEST[MA] TNRM $
25 1482                                                         ;GET CORE STATUS ENTRY
25 1483 10740 010770170000060542561623657712160000             DFRQ D[IR] ROT[10.] ALU[NOTD] DEST[Q] $
25 1484                                                         ;START FETCH
25 1485                                                         ;GET RWX DISABLES
25 1486 10741 010640170000060545251623655754160000             D[CONST 24] ROT[21.] ALU[D&Q] DEST[Q] TNRM $
25 1487                                                         ;GET ONLY R & X
25 1488 10742 010600170000060545211623655754160000             D[CONST 4] ROT[21.] ALU[D+Q] DEST[Q] TNRM $
25 1489                                                         ;MOVE X BIT LEFT 1
25 1490 10743 010630170000060542257623666354160000             D[D%AMEM14] ROT[9.] MASK[23.] ALU[DORQ] DEST[Q] TNRM $
25 1491                                                         ;OR IN PAGE ADDRS (NOTE THE Z3. BIT PHYS ADR!) ***
25 1492 10744 010731002000000000607620254161556000             D[MEM] ROT[3] MASK[3] COND[ZERO] JUMP[MFTR7] CS50 $
25 1493                                                         ;J IF CST AGE SAYS TRAP
25 1494 10745 010731002000000000400362025421456000             D[AR] ROT[16.] MASK[1] COND[ZERO] JUMP[MFA3] CS50 $
25 1495                                                         ;J IF NO WRT RQ
25 1496 10746 610731170000060554161443654354160000             D[AR] DEST[IR-ADR] TNRM $
25 1497                                                         ;SAVE ORRIGINAL MA IN IR
25 1498 10747 010731170000060554161303654354160000             D[AR] DEST[D%AMEM14] TNRM $
25 1499                                                         ;SAVE TRAP WORD IN D%AMEM
25 1500 10750 010641370000060546107623647354160000             D[MASK 35.] ROT[24.] ALU[D&Q] DEST[AR] TNRM $
25 1501                                                         ;TURN OFF WRT PREVENT, SAVE NEW MAP WORD
25 1502 10751 010730170000060546403623655754160000             D[CONST 1] ROT[26.] DEST[Q] TNRM $
25 1503                                                         ;GET MODIFICATION BIT
25 1504 10752 010630170000060554065623654755160000             D[MEM] MASK[26.] ALU[DORQ] DEST[Q] TNRM $
25 1505                                                         ;OR INTO CST WORD
25 1506 10753 010731170000060542223503660754160000     MFA5:   D[D%AMEM1] ROT[9.] MASK[9.] DEST[HOLD] TNRM $
25 1507                                                         ;GET JUST AGE
25 1508 10754 010631170000060546761563654755160000             D[MEM] ROT[27.] ALU[DORQ] DEST[MEMSTO] TNRM $
25 1509                                                         ;OR in age only and store CST0 entry.
25 1510                                                   .repeat 1 - WAITS [
25 1511                                                         D[CONST N-AR-REFILLS] DEST[MA] TNRM $
25 1512                                                         DF/WT D[MEM] ALU[D+1] DEST[MEMSTO] TNRM $
25 1513                                                         ;BUMP # AR REFILLS
25 1514                                                     ];.repeat 1 - WAITS
25 1515                                                 ;(Leave this label in for debugging/documentation)
25 1516 10755 010241170000060554160723654354170000     LOADAR: ALU[0] DEST[MERGE] TNRM $
25 1517                                                         ;Set virtual address merger to use context for left half.
25 1517
25 1518 10756 010731006000060550761623665354160000             D[SAVE-MAP-SR] ROT[35.] COND[SIGNOFF] TNRM $
25 1519                                                         ;Test whether the MAP-SR was active...
25 1520 10757 710731170000000014045620657754560000             D[IR] MASK[18.] DEST[MA] TNRM JUMPLC[LOADA0] $
25 1521                                                         ;Get original MA.  Jump if MAP-SR was active.
25 1522 10760 010731170000060550610403665214170000             D[SAVE-MAP-SR] ROT[36. - 1] MASK[4] DEST[MAP-SR] CS50 $
25 1523                                                         ;MAP-SR was active... reload it.
25 1524 10761 010730170000060540561623654354160000     LOADA0: D[AR] ROT[2] DEST[Q] TNRM $
25 1525                                                         ;Get XP and WP bits in new position (phys adr got shifted
 too)
25 1525
25 1526 10762 010641002000000000660362025561456000             D[CONST 1] ROT[27.] ALU[D&Q] COND[ZERO] JUMP[LOADA1] CS50 $
25 1527                                                         ;Jump if RP bit off in old position (bit 10 rotated Z)
25 1528 10763 010630170000000000600362025754560000             D[CONST 1] ROT[24.] ALU[DORQ] DEST[Q] JUMP[LOADA2] TNRM $
25 1529                                                         ;RP bit was on, turn on XR in new position (bit 11)
25 1530 10764 010650170000060546003623655754160000     LOADA1: D[CONST 1] ROT[24.] ALU[-D&Q] DEST[Q] TNRM $
25 1531                                                         ;WAS OFF, TURN OFF XR IN NEW POSITION (BIT11)
25 1532 10765 010640170001600000001623655354160000     LOADA2: D[LIT 000700000000] ALU[D&Q] DEST[Q] TNRM $
25 1533                                                         ;RETAIN JUST CORRECTLY POSITIONED ACCESS IN Q
25 1534 10766 010660170000600000001623655354160000             D[LIT 000300000000] ALU[D#Q] DEST[Q] TNRM $
25 1535                                                         ;Hardware wants Read and Write access complemented.
25 1536 10767 010731370000060554057623654354160000             D[AR] MASK[14. +  9.] DEST[AR] TNRM $
25 1537                                                         ;RETAIN Z3. BIT PHYS ADR IN AR ***
25 1538                                                         ;F2/3 AR HAD BEEN ONLY Z0. BIT ADR BUT FAKE BIGGER HERE!
25 1538
25 1539 10770 010631170000060554160543654114170000             D[AR] ALU[DORQ] DEST[MAP-MA] LONG $
25 1540                                                         ;ACCESS!ADR => MAP WD FOR MA
25 1541
25 1542 10771 010730124000000004541620267614560000             D[D%AMEM17] ROT[18.] MASK[LEFT] DEST[Q] COND[USER] JUMP[LOADA3]
 CS50 $
25 1543                                                         ;Load Q with positioned copy of USER-CTXT if USER...
25 1544 10772 010730170000060544541623667354160000             D[D%AMEM16] ROT[18.] MASK[LEFT] DEST[Q] TNRM $
25 1545                                                         ;...or copy of EXEC-CTXT if EXEC
25 1546 10773 010630170000060554045623656754160000     LOADA3: D[MA] MASK[18.] ALU[DORQ] DEST[Q] TNRM $
25 1547 10774 010661170077777777776563655114170000             D[LIT 037777777777] ALU[D#Q] DEST[MAP-TAG] LONG  $
25 1548                                                         ;CTXT!VADR => MAP WD FOR MA (Complement tag bits except
25 1549                                                         ; for bit 3, which is guaranteed to be 0 when we get
25 1549
 here!!)
25 1550 10775 010731170000213354161620254354560000             JUMP[MAPUTP] TNRM $
25 1551                                                         ;Go restore state, take appropriate completion action,
25 1552                                                         ;and resume trapped instruction.
25 1553
25 1554
```

```
26 1555
26 1556 10776 010731000000000002403620254615560000       MFA3:   D[MEM] ROT[10.] MASK[1] COND[-ZERO] JUMP[MFA4] C550 $
26 1557                                                           ;J IF MODIF. BIT ON
26 1558 10777 610731170000060554161403654354160000       MFA7:   D[AR] DEST[IR-ALL] TNRM $
26 1559                                                           ;SAVE ORIGINAL MA IN IR
26 1560 11000 010731170000060554161303654354160000               D[AR] DEST[D%AMEM14] TNRM $
26 1561                                                           ;SAVE TRAP WORD IN D%AMEM
26 1562 11001 010631370000060545603623655754160000               D[CONST 1] ROT[23.] ALU[DORQ] DEST[AR] TNRM $
26 1563                                                           ;TURN ON WRT-PREVENT, SAVE NEW MAP WORD
26 1564 11002 010730170000217274067620254755560000       MFA6:   D[MEM] MASK[27.] DEST[Q] JUMP[MFA5] TNRM $
26 1565                                                           ;GET CST ENTRY, GO DO
26 1566
26 1567
26 1568 11003 010731002000217765403620257614560000       MFA4:   D[IR] ROT[22.] MASK[1] COND[ZERO] JUMP[MFA7] C550 $
26 1569                                                           ;J IF NO WRT-PERMIT
26 1570 11004 610731170000060554161403654354160000               D[AR] DEST[IR-ALL] TNRM $
26 1571                                                           ;SAVE ORRIGINAL MA IN IR
26 1572 11005 010731170000060554161303654354160000               D[AR] DEST[D%AMEM14] TNRM $
26 1573                                                           ;SAVE TRAP WORD IN D%AMEM
26 1574 11006 010641370000220046107620247354560000               D[MASK 35.] ROT[24.] ALU[D&Q] DEST[AR] JUMP[MFA6] TNRM $
26 1575                                                           ;CLEAR WRT-PREVENT AND JUMP
26 1576
26 1577
```

```
27 1578
27 1579
27 1580                                                 ;TRAP STUFF
27 1581
27 1582                                                 ;TRAP BITS SET
27 1583
27 1584 11007 010731000000000002Z036Z025461556000      MFTR1:  D[MEM] ROT[9.] MASK[1] COND[-ZERO] JUMP[MFTR9] C550 $
27 1585                                                         ;J IF TRAP TO USER
27 1586 11010 010731000000000002403Z025461556000                D[MEM] ROT[10.] MASK[1] COND[-ZERO] JUMP[MFTR10] C550 $
27 1587                                                         ;J IF WRT TRAP
27 1588 11011 010731170000000014161Z025475556000                D[MEM] JUMP[MFTR2] TNRM $
27 1589                                                         ;TREAT BOTH "TRAP-TO-MON" CODES AS IMMEDIATE
27 1590
27 1591
27 1592 1101Z 010731002000216204003Z025421456000      MFTR10: D[AR] ROT[16.] MASK[1] COND[ZERO] JUMP[MFB3] C550 $
27 1593                                                         ;WRT RQ? J IF NO
27 1594 11013 010730170000060547511623655575416000             D[CONST 44] ROT[30.] DEST[Q] TNRM $
27 1595                                                         ;GET ERROR CODE BITS -- WRT TRAP
27 1596                                                 MFB5:
27 1597 11014 010630324000000014161Z025421456000       MFB4:   D[AR] ALU[DORQ] DEST[AR Q] COND[USER] JUMP[MFB41] C550 $
27 1598                                                         ;OR ERROR BITS IN, J IF USER MODE
27 1599 11015 010631370000060544403623655575416000             D[CONST 1] ROT[18.] ALU[DORQ] DEST[AR] TNRM $
27 1600                                                         ;TURN ON EXEC BIT
27 1601 11016 010730170000000001363623655535416000     MFB41:  D[LIT PSB571] DEST[Q] TNRM $
27 1602 11017 710631170000060554161623662354160000             D[D%AMEM4] ALU[DORQ] DEST[MA] TNRM $
27 1603                                                         ;ADD PSB (MON BASE TAB)
27 1604                                                         ;FORM PHYS ADR OF PSB CELL 571
27 1605 11020 010731170000021471416156025431456000             D[AR] DEST[MEMSTO] JUMP[MACTRP] $
27 1606                                                         ;STORE ERROR BITS THERE
27 1607                                                         ;NO NEED TO STORE DATA IN PSB572 ANYMORE!!
27 1608                                                         ;TAKE MACRO TRAP AFTER DOING MEM STATE SAVE
27 1609
27 1610
27 1611                                                 ;NO ACCESS PERMISSION TRAP
27 1612
27 1613 110Z1 010730170000Z2030750362025575456000      MFTR2:  D[CONST 41] ROT[30.] DEST[Q] JUMP[MFB5] TNRM $
27 1614
27 1615
27 1616                                                 ;ADR LIMIT TRAP
27 1617                                                 ;OR ILLEGAL PTR TRAP
27 1618
27 1619                                                 MTRPAL:
27 1620 1102Z 010730170000060547501623655575416000     MFTR3:  D[CONST 40] ROT[30.] DEST[Q] TNRM $
27 1621 11023 010630170000Z2030660362025575456000               D[CONST 1] ROT[27.] ALU[DORQ] DEST[Q] JUMP[MFB5] TNRM $
27 1622
27 1623
27 1624                                                 ;NOT IN CORE TRAP
27 1625
27 1626                                                 MFTR4:
27 1627 11024 010730170000Z2030744562025575456000      MFTR5:  D[CONST 22] ROT[30.] DEST[Q] JUMP[MFB5] TNRM $
27 1628
27 1629
27 1630                                                 ;ILLEGAL ACCESS TRAP
27 1631
27 1632 11025 010641002000000001401362025561456000     MFTR6:  D[CONST 5] ALU[D&Q] COND[ZERO] JUMP[MFTR61] C550 $
27 1633                                                         ;J IF NO R OR X ERROR
27 1634 11026 010630170000060546611623655575416000             D[CONST 4] ROT[27.] ALU[DORQ] DEST[Q] TNRM $
27 1635                                                         ;OR IN "R OR X ERROR" BIT
27 1636 11027 010641002000000001400562025561456000             D[CONST 2] ALU[D&Q] COND[ZERO] JUMP[MFTR62] C550 $
27 1637                                                         ;J IF NO W ERROR
27 1638 11030 010630170000060546605623655575416000     MFTR61: D[CONST 2] ROT[27.] ALU[DORQ] DEST[Q] TNRM $
27 1639                                                         ;OR IN W ERROR BIT
27 1640 11031 010650170000060554017623655575416000     MFTR62: D[CONST 7] ALU[-D&Q] DEST[Q] TNRM $
27 1641 11032 010630170000Z2030750162025575456000               D[CONST 40] ROT[30.] ALU[DORQ] DEST[Q] JUMP[MFB4] TNRM $
27 1642                                                         ;GROUP 2
27 1643
27 1644
27 1645                                                 ;CST AGE = 0 TRAP
27 1646
27 1647 11033 010730170000Z2030742162025575456000      MFTR7:  D[CONST 10] ROT[30.] DEST[Q] JUMP[MFB4] TNRM $
27 1648                                                         ;GROUP 0
27 1649
27 1650 11034 010730170000Z2030750562025575456000      MFTR9:  D[CONST 42] ROT[30.] DEST[Q] JUMP[MFB4] TNRM $
27 1651                                                         ;USER TRAP
27 1652
27 1653
27 1654                                                 ;TOO MANY INDIRECTIONS
27 1655
27 1656 11035 010730170000060547441623655575416000     MFTR41: D[CONST 20] ROT[30.] DEST[Q] TNRM $
27 1657                                                         ;GROUP 1
27 1658 11036 010630170000Z2030660362025575456000               D[CONST 1] ROT[27.] ALU[DORQ] DEST[Q] JUMP[MFB4] TNRM $
27 1659                                                         ;TOO MANY INDR. PNTRS
27 1660
27 1661
```

```
28 1662
28 1663
28 1664
28 1665                                                 ;ACCEPTS DEV-ADR/ MAPDEV
28 1666                                                 ;CLEAR ALL MAP WDS AND INIT CTXTS.
28 1667                                                 ;LEAVE MAP OFF
28 1668
28 1669
28 1670 11037 010731170013060554161623654314160000     MAPRST: MAP-DISABLE $
28 1671 11040 010730170013000314161623670354160000             SPEC[MAP-DISABLE] D[AMEM-ABS APR-MODE] DEST[Q] TNRM $
28 1672                                                                 ;GET CURRENT CPU-MODE
28 1673 11041 010651370000000001400362225575456000             D[CONST 1] ALU[-D&Q] DEST[AR] PUSHJ[SETMODE] TNRM $
28 1674                                                                 ;CLEAR BIT 35. (MAP OFF FOR SURE)
28 1675 11042 010241170000060554160623654314170000             ALU[0] DEST[USER-CTXT] $
28 1676                                                                 ;CLEAR USER CONTEXT REG (INVALIDATE ALL USER ARS)
28 1677 11043 010241170000060554161363654314160000             ALU[0] DEST[D%AMEM17] $
28 1678                                                                 ;CLEAR AMEM COPY OF USER CONTEXT REG
28 1679 11044 010241170000060554160603654314170000             ALU[0] DEST[EXEC-CTXT] $
28 1680                                                                 ;CLEAR EXEC CONTEXT REG (INVALIDATE ALL EXEC ARS)
28 1681 11045 010241170000060554161343654314160000             ALU[0] DEST[D%AMEM16] $
28 1682                                                                 ;CLEAR AMEM COPY OF EXEC CONTEXT REG
28 1683 11046 010731170000211643603322255754560000             D[CONST 1] ROT[15.] DEST[D%AMEM15] PUSHJ[FORCEM] TNRM $
28 1684                                                                 ;ASSUME PAGE 100 IS FIRST EXEC SWAPPING PAGE TIL TOLD
28 1684                                                  BETTER
28 1685                                                                 ;GET 1ST ADR OF SWAPPABLE MONITOR IN D%AMEM15
28 1686                                                                 ;AND SETS UP THE MAP TO FORCE 1:1 MAPPING FOR
28 1687                                                                 ;PAGES 0,2:<D%AMEM15-1>
28 1688 11047 010731170000211270603042255754560000             D[CONST 1] ROT[35.] DEST[D%AMEM2] PUSHJ[EXECLR] TNRM $
28 1689                                                                 ;SET MAPONE SO EXECLR CLEARS PAGE 1
28 1690                                                                 ;MEANS WE ARE MAPPING PAGE 1 TIL TOLD BETTER
28 1691                                                                 ;CLEAR EVERY MAP WD, PAGE 1 INCLUDED
28 1692                                                                 ;INIT VALID EXEC-CTXT
28 1693 11050 010731170000210734161622254354560000             PUSHJ[USRCLR] TNRM $
28 1694                                                                 ;CLEAR ALL USER MAP WDS
28 1695                                                                 ;INIT VALID USER-CTXT
28 1696 11051 010731170000060554161624254354160000             POPJ TNRM $
28 1697                                                                 ;RETURN
28 1698
28 1699
```

```
29 1700
29 1701 11052 010733570000000553416000377031416000      ECCTRP: D[AMEM-ABS STATE05] DEST[ALU1 AC0] $
29 1702                                                         ;Preserve STATE05 (in case we are inside a map trap !)
29 1703 11053 010731170000000553416154365435416000             D[AR] DEST[AMEM-ABS STATE05] TNRM $
29 1704                                                         ;Save AR (needed below at MAPRD)
29 1705 11054 010731370000006354161541364335416000             D[MB-STATUS] DEST[AMEM-ABS[ECC-TEMP1] AR] TNRM $
29 1706                                                         ;Put trap status in AR for TRAP-ENTER, and save in
29 1706                                                  ECC-TEMP1
29 1707 11055 010730570000060546207623654954116000     ECCTP1: D[AR] ROT[1 + 24.] MASK[3] DEST[ALU1 Q] TNRM $
29 1708 11056 010661402000000014005620255614560000             D[CONST 2] ALU[ALU1 Q=D]   C550  ZERO JUMP[ECCFX] $
29 1709                                                         ;Jump if FIXUP needed (bits 22:24 = 010).
29 1710 11057 010733570000215214160022395675456000             D[MA] DEST[ALU1 AC1] TNRM  PUSHJ[TRAP-ENTER] $
29 1711                                                         ;Save MA in our own private spot (not used by map trap).
29 1711
29 1712                                                         ;Get MA loaded with address of failing reference, and a
29 1713                                                         ; code in AR indicating the type of cycle trapped.
29 1714                                                 ECCFXZ:
29 1715 11060 640731170000000074161663657654116000     ECCUTP: LOAD-PPC $
29 1716                                                         ;Get PPC loaded with mapped version of PC.
29 1717 11061 010731004000006342560263670214160000             SELECT-HOLD D[AMEM-ABS ECC-TEMP1] ROT[10.] C500 COND[SIGNON] $
29 1718                                                         ;Put HOLD on MEM bus.  Check the Hard Error bit.
29 1719                                                         JUMPLC[ECCHRD]
29 1720 11062 010733570000000014160060754755556000               D[MEM] DEST[ALU1 AC3] TNRM $
29 1721                                                         ;JUMP IF HARD
29 1722                                                         ;Save contents of HOLD.
29 1723 11063 010730570000000300161623670314160000             D[AMEM-ABS APR-MODE] ROT[CPU-MODE-ROT] DEST[ALU1 Q] $
29 1724                                                         ; Get CPU mode bits.
29 1725 11064 010631570000060540004223655714170000             D[CONST 2] ROT[CPU-MODE-ROT] ALU[ALU1 DorQ] DEST[CPU-MODE] $
29 1726                                                         ;Enable ECC correction.
29 1727 11065 010231570000060554160223654212170000             DFRQ ALU[ALU1 Q] DEST[CPU-MODE] C550 $
29 1728                                                         ;Fetch the (corrected) data, disable correction again.
29 1729                                                 ;;;     SUPPRESS-FETCH-TRAPS D[MEM] DEST[MEMSTO  ALU1 Q] $
29 1730                                                         ;Attempt to re-write data with good ECC.
29 1731 11066 011700570000006234161623670354160000             D[AMEM-ABS ECC-SFTCNT] ALU[D+1] DEST[ALU1 Q] TNRM $
29 1732                                                         ;Increment count of SOFT ECC errors
29 1733 11067 010231570000006234161543654354160000             ALU[ALU1 Q] DEST[AMEM-ABS ECC-SFTCNT] TNRM $
29 1734                                                 .REPEAT 1 - WAITS [
29 1735                                                         D[AMEM-ABS APR-STATUS] DEST[ALU1 Q] TNRM $
29 1736                                                         ;GET APR CONI BITS
29 1737                                                         D[CONST 1] ROT[35. - 22.] ALU[ALU1 D&Q] COND[-ZERO] JUMP[ECCXIT]
29 1737                                                  C550 $
29 1738                                                         ;DON'T OVERWRITE LAST DATA UNTIL CONI BIT CLEARED
29 1739                                                         ;NON-RECORDED DATA IS LOST DUE TO SLOWNESS
29 1740                                                         ;COUNT IS MAINTAINED THOUGH SO LOSSAGE DETECTABLE
29 1741                                                         D[CONST 1] ROT[35. - 22.] ALU[ALU1 DORQ] DEST[AMEM-ABS
29 1741                                                  APR-STATUS] $
29 1742                                                         ;LIGHT APR CONI BIT FOR SOFT ECC INTERRUPT REQUEST
29 1743                                                         ;WILL BE DETECTED IN APRCHK
29 1744                                                 ];.REPEAT 1 - WAITS
29 1745                                                 .REPEAT WAITS [ ;Bit 22 is ILL MEM REF on KA-10!  Use bit 24 (SBUS error)
29 1745
29 1746                                                         ;(Bit 27 would also be acceptable)
29 1747                                                 D[AMEM-ABS APR-STATUS] DEST[ALU1 Q] TNRM $
29 1748                                                         ;GET APR CONI BITS
29 1749                                                         D[CONST 1] ROT[35. - 24.] ALU[ALU1 D&Q] COND[-ZERO] JUMP[ECCXIT]
29 1749                                                  C550 $
29 1750                                                         ;DON'T OVERWRITE LAST DATA UNTIL CONI BIT CLEARED
29 1751                                                 .repeat 0 []    ;NON-RECORDED DATA IS LOST DUE TO SLOWNESS
29 1752                                                         ;COUNT IS MAINTAINED THOUGH SO LOSSAGE DETECTABLE
29 1753                                                         D[CONST 1] ROT[35. - 24.] ALU[ALU1 DORQ] DEST[AMEM-ABS
29 1753                                                  APR-STATUS] $
29 1754                                                         ;LIGHT APR CONI BIT FOR SOFT ECC INTERRUPT REQUEST
29 1755                                                         ;WILL BE DETECTED IN APRCHK
29m1756                                                 ];[     ;Bit 22 is ILL MEM REF on KA-10!  Use bit 24 (SBUS error)
29m1756                                                         ;(Bit 27 would also be acceptable)
29m1756 11070 010730570000000054161623670354160000             D[AMEM-ABS APR-STATUS] DEST[ALU1 Q] TNRM $
29m1756                                                         ;GET APR CONI BITS
29m1756 11071 010641400000000026036202556145600 0              D[CONST 1] ROT[35. - 24.] ALU[ALU1 D&Q] COND[-ZERO] JUMP[ECCXIT]
29m1756                                                  C550 $
29m1756                                                         ;DON'T OVERWRITE LAST DATA UNTIL CONI BIT CLEARED
29m1756                                                 .repeat 0 []    ;NON-RECORDED DATA IS LOST DUE TO SLOWNESS
29m1756                                                         ;COUNT IS MAINTAINED THOUGH SO LOSSAGE DETECTABLE
29m1756 11072 010631570000000042603543655714160000             D[CONST 1] ROT[35. - 24.] ALU[ALU1 DORQ] DEST[AMEM-ABS
29m1756                                                  APR-STATUS] $
29m1756                                                         ;LIGHT APR CONI BIT FOR SOFT ECC INTERRUPT REQUEST
29m1756                                                         ;WILL BE DETECTED IN APRCHK
29 1756                                                 ].REPEAT WAITS
29 1757 11073 010730570000006354141623670354160000             D[AMEM-ABS ECC-TEMP1] MASK[LEFT] DEST[ALU1 Q] TNRM $
29 1758                                                         ;Get status bits, including ECC syndrome.
29 1759 11074 010631570000006254045543656714160000             D[MA] MASK[18.] ALU[ALU1 DORQ] DEST[AMEM-ABS ECC-SFTD1] $
29 1760                                                         ;Put failing address in right half
29 1761 11075 010731170000000014161622254354560000             PUSHJ[ECCPMA] TNRM $
29 1762                                                         ;GET PMA IN ALU1 Q
29 1763 11076 010231570000006274161543654314160000             ALU[ALU1 Q] DEST[AMEM-ABS ECC-SFTD2] $
29 1764                                                         ;PUT IN PLACE FAILING PHYS ADR
29 1765 11077 010730570000006055414162364625416000             D[PC-FLAGS] MASK[LEFT] DEST[ALU1 Q] T350 $
29 1766 11100 010631570000006314045543656314160000             D[PC] MASK[18.] ALU[ALU1 DORQ] DEST[AMEM-ABS ECC-SFTPC] $
29 1767                                                         ;ENTER FLAGS.,PC
29 1768                                                         ;NOTE THAT THE INT WON'T HAPPEN ON END-OF-INSTRUCTION AS
29 1768
29 1769                                                         ;IT SHOULD (WILL HAPPEN AT NEXT 60HZ TICK IN APRCHK).
29 1770                                                         ;ACCEPT THAT SFT INT WILL COME SOME # OF INSTRS AFTER THE
29 1770
29 1771                                                         ;OFFENDING INSTR.
29 1772                                                         ;FALL THRU...
29 1773                                                 ;;;;*** Fix interrupt at end-of-instruction.  Currently, screws up PI's.
29 1773
29 1774                                                 ;;;;*** I think this might cause the loss of clock interrupts. TVR/Feb83
29 1774
29 1775
29 1776                                                 ;HERE ANY ERROR RECORDING HAS BEEN DONE
29 1777
29 1778 11101 010431570014060554161503731754160000     ECCXIT: REENABLE-TRAP1 D[AC3] ALU[ALU1 A] DEST[HOLD] TNRM $
29 1779                                                           ;Prepare to return.  First, restore HOLD.
29 1780 11102 010731004000000003560300254214560000             SELECT-MB D[AR] ROT[14.] C550 COND[SIGNON] JUMP[ECCUDF] $
29 1781                                                           ;Test code from TRAP-ENTER, jump if trap is from a data fetch.
29 1781
29 1782 11103 010731170000060554160743654374160000             CLR-ECC-ERR SHORT $
29 1783 11104 540731170000060474161623254254160000             XCT-IDISP $
29 1784                                                           ;Trap was from an instruction fetch.  This is easy.
```

```
29 1789 11110 010431570000005534161541373035416000             D[AC0] ALU[ALU1 A] DEST[AMEM-ABS STATE0S] TNRM $
29 1790 11111 010730570000006554161623644161 6000             D[LOOP-CTR] DEST[ALU1 Q] C500 $
29 1791 11112 010501570010060540002003655754170 00            D[CONST 1] ROT[LLOAD-ROT] ALU[ALU1 D+Q] DEST[LLOAD] TNRM $
29 1792                                                                ;Loop ctr. gets decremented (by hardware) during traps..
29 1793
29 1794 11113 010730406000006341761623670354160 00            D[AMEM-ABS ECC-TEMP1] ROT[7] DEST[ALU1 Q] COND[SIGNOFF] TNRM $
29 1795                                                                  ;Find out from MB-STATUS whether trapped uinst. had MI MEM WAIT
29 1795                                                         on.
29 1796                                                                JUMPLC[ECCUD1]
29 1797 11114 010641402000000000440362065571456000                      D[CONST 1] ROT[35. - ( 24. - 7 )]  ALU[ALU1 D&Q] COND[ZERO] $
29 1798                                                                  ;Jump if MI MEM WAIT was on, test whether trap was from P-Idisp
29 1798                                                         + 1.
29 1799 11115 710431570000006055416162473075416000                    D[AC1]  ALU[ALU1 A] DEST[MA] TNRM  POPJLC $
29 1800                                                                  ;Restore MA, return if trap not from cycle after Principle
29 1800                                                         Idisp.
29 1801 11116 440731170000000034161623257654160 00                    RE-IDISP $
29 1802                                                        ECCUD1: ;The FAKE-WAIT's below inhibit the DFRQ in the uinst. we return
29 1802                                                         to.
29 1803                                                                ; (This is necessary since that uinst has DFRQ, WAIT, D[MEM])
29 1804 11117 710431570024060554161624730754160 00                    FAKE-DFWAIT D[AC1] ALU[ALU1 A] DEST[MA] TNRM  POPJLC $
29 1805                                                                  ;Restore MA, return if trap not from cycle after Principle
29 1805                                                         Idisp.
29 1806 11120 440731170024000034161623257654160 00                    FAKE-DFWAIT RE-IDISP $
29 1807
29 1808
29 1809                                                        ;HERE FOR HARD ERROR OR NXM
29 1810
29 1811 11121 010730570000006344415623670314160 00            ECCHRD: D[AMEM-ABS ECC-TEMP1] ROT[18.] MASK[6] DEST[ALU1 Q] $
29 1812                                                                  ;Hard error.  Extract syndrome bits.
29 1813 11122 010661402000000014015620255614560 00                    D[CONST 06] ALU[ALU1 D*Q] C550 COND[ZERO] JUMP[ECCNXM] $
29 1814                                                                  ;JUMP IF NXM
29 1815 11123 011700570000006134161623670354160 00                    D[AMEM-ABS ECC-HRDCNT] ALU[D+1] DEST[ALU1 Q] TNRM $
29 1816                                                                  ;Increment count of HARD ECC
29 1817 11124 010231570000006134161541365435416000                    ALU[ALU1 Q] DEST[AMEM-ABS ECC-HRDCNT] TNRM $
29 1818 11125 010730570000000114161623670354160 00                    D[AMEM-ABS PI-STATUS] DEST[ALU1 Q] TNRM $
29 1819                                                                  ;GET PI CONI BITS
29 1820 11126 010641400000000004003620255614560 00                    D[CONST 1] ROT[35. - 19.] ALU[ALU1 D&Q] COND[-ZERO] JUMP[ECCDIE]
29 1820                                                         C550 $
29 1821                                                                  ;DON'T OVERWRITE LAST DATA UNTIL CONI BIT CLEARED
29 1822 11127 010631570000000104003541365714160 00                    D[CONST 1] ROT[35. - 19.] ALU[ALU1 DORQ] DEST[AMEM-ABS PI-STATUS]
29 1822                                                         $
29 1823                                                                  ;LIGHT PI CONI BIT FOR HARD ECC INTERRUPT REQUEST
29 1824                                                                  ;WILL BE DETECTED IN APRCHK
29 1825 11130 010730570000006354141623670354160 00                    D[AMEM-ABS ECC-TEMP1] MASK[LEFT] DEST[ALU1 Q] TNRM $
29 1826                                                                  ;Get status bits, including ECC syndrome.
29 1827 11131 010631570000006154045543656714160 00                    D[MA] MASK[18.] ALU[ALU1 DORQ] DEST[AMEM-ABS ECC-HRDD1] $
29 1828                                                                  ;Put failing address in right half
29 1829 11132 010731170000000014161622254354560 00                    PUSHJ[ECCPMA] TNRM $
29 1830                                                                  ;GET PMA IN ALU1 Q
29 1831 11133 010231570000006174161541365431416000                    ALU[ALU1 Q] DEST[AMEM-ABS ECC-HRDD2] $
29 1832                                                                  ;PUT IN PLACE FAILING PHYS ADR
29 1833 11134 010730570000060554141623646254160 00                    D[PC-FLAGS] MASK[LEFT] DEST[ALU1 Q] T350 $
29 1834 11135 010631570000006214045543656314160 00                    D[PC] MASK[18.] ALU[ALU1 DORQ] DEST[AMEM-ABS ECC-HRDPC] $
29 1835                                                                  ;ENTER FLAGS,,PC
29 1836 11136 010770570000000114161623670354160 00                    D[AMEM-ABS PI-STATUS] ALU[NOTD] DEST[ALU1 Q] TNRM $
29 1837                                                                  ;GET NOT OF APR FLAGS KEPT IN PI CONI WORD
29 1838 11137 010641400000022202360762025561456000 00                 D[CONST 3] ROT[35. - 20.] ALU[ALU1 D&Q] COND[-ZERO] JUMP[ECCXIT]
29 1838                                                         C550 $
29 1839                                                                  ;JUMP IF PAR ERR INTS NOT ENABLED AND ON
29 1840 11140 010731370000000054007623670354160 00                    D[AMEM-ABS APR-STATUS] MASK[3.] DEST[AR] TNRM $
29 1841                                                                  ;APR CHANNEL
29 1842 11141 010731170000203431416162025431456000                    JUMP[PIGEN] $
29 1843                                                                  ;ABORT INTRUCTION AND GIVE ERROR INT IMMEDIATELY
29 1844                                                                  ;SINCE END-OF-INSTRUCTION WON'T CAUSE INTERRUPT
29 1845                                                                  ;AS IT SHOULD.
29 1846
29 1847
29 1848                                                        ;HERE IF LAST NXM OR HARD ECC ERROR NOT PROCESSED AND ANOTHER OCCURRED
29 1849                                                        ;CAN'T LET THESE GUYS GO UNNOTICED...
29 1850
29 1851 11142 010731370014000014155620255714560 00            ECCDIE: REENABLE-TRAP1 D[CONST 66] DEST[AR] JUMP[HLTLOC] $
29 1852                                                                  ;You die now, Yankee...
29 1853                                                                  ;DATA IN ECC-HRDD1,2 OR ECC-NXMD1,2 IS PREVIOUS ERROR
29 1853                                                         DATA
29 1854                                                                  ;DATA OF CURRENT TRAP NOT YET DISTRIBUTED TO THESE LOCS
29 1855
29 1856                                                        ;HERE FOR NXM
29 1857
29 1858 11143 011700570000006034161623670354160 00            ECCNXM: D[AMEM-ABS ECC-NXMCNT] ALU[D+1] DEST[ALU1 Q] TNRM $
29 1859                                                                  ;Increment count of NXMS
29 1860 11144 010231570000006034161541365435416000                    ALU[ALU1 Q] DEST[AMEM-ABS ECC-NXMCNT] TNRM $
29 1861 11145 010730570000000054161623670354160 00                    D[AMEM-ABS APR-STATUS] DEST[ALU1 Q] TNRM $
29 1862                                                                  ;GET APR CONI BITS
29 1863 11146 010641400000022304300362025561456000                    D[CONST 1] ROT[35. - 23.] ALU[ALU1 D&Q] COND[-ZERO] JUMP[ECCDIE]
29 1863                                                         C550 $
29 1864                                                                  ;DON'T OVERWRITE LAST DATA UNTIL CONI BIT CLEARED
29 1865 11147 010631570000000043003541365714160 00                    D[CONST 1] ROT[35. - 23.] ALU[ALU1 DORQ] DEST[AMEM-ABS
29 1865                                                         APR-STATUS] $
29 1866                                                                  ;LIGHT APR CONI BIT FOR NXM INTERRUPT REQUEST
29 1867                                                                  ;WILL BE DETECTED IN APRCHK
29 1868 11150 010730570000006354141623670354160 00                    D[AMEM-ABS ECC-TEMP1] MASK[LEFT] DEST[ALU1 Q] TNRM $
29 1869                                                                  ;Get status bits, including ECC syndrome.
29 1870 11151 010631570000006054045543656714160 00                    D[MA] MASK[18.] ALU[ALU1 DORQ] DEST[AMEM-ABS ECC-NXMD1] $
29 1871                                                                  ;Put failing address in right half
29 1872 11152 010731170000000014161622254354560 00                    PUSHJ[ECCPMA] TNRM $
29 1873                                                                  ;GET PMA IN ALU1 Q
29 1874 11153 010231570000006074161541365431416000                    ALU[ALU1 Q] DEST[AMEM-ABS ECC-NXMD2] $
29 1875                                                                  ;PUT IN PLACE FAILING PHYS ADR
29 1876 11154 010730570000060554141623646254160 00                    D[PC-FLAGS] MASK[LEFT] DEST[ALU1 Q] T350 $
29 1877 11155 010631570000006114045543656314160 00                    D[PC] MASK[18.] ALU[ALU1 DORQ] DEST[AMEM-ABS ECC-NXMPC] $
29 1878                                                                  ;ENTER FLAGS,,PC
29 1879                                                        .REPEAT WAITS (
29 1880                                                                D[AMEM-ABS APR-STATUS] MASK[3.] DEST[AR] $
29 1881                                                                  ;Get APR PI channel
29 1882                                                                D[AR] COND[-OBUS=0] JUMP[PIGEN] C550 $
29 1883                                                                  ;Abort instruction and take APR trap if enabled.
29 1883                                                         Instruction
29 1884                                                                  ;can't be restarted anyway.
29m1885                                                        );[
29m1885 11155 010731370000000054007623670354160 00                    D[AMEM-ABS APR-STATUS] MASK[3.] DEST[AR] $
```

```
29m1885                                                         ;can't be restarted anyway.
29 1885                                                 ].REPEAT WAITS
29 1886 11160 010731170000ZZZ034161620Z5431456000               JUMP[ECCXIT] $
29 1887                                                         ;LET PROCESS FIELD THIS PANIC INTERRUPT, DON'T STOP
29 1887                                                  MACHINE!
29 1888                                                         ;NOTE THAT THE INT WON'T HAPPEN ON END-OF-INSTRUCTION AS
29 1888
29 1889                                                         ;IT SHOULD (WILL HAPPEN AT NEXT 60H2 TICK IN APRCHK) SO
29 1890                                                         ;PROCEEDING IS DEFINITELY WRONG BUT WE HAVE TO FOR
29 1891                                                         ;MEMORY SIZING TO WORK.  ACCEPT THAT NXM INT WILL COME
29 1891                                                  SOME
29 1892                                                         ;# OF INSTRS AFTER THE OFFENDING INSTR.
29 1893
29 1894
```

```
30 1895
30 1896 11161 0107335700030605541600477453541600 0       ECCFX:  MU-POP D[JMEM] DEST[ALU1 AC2] TNRM $
30 1897                                                           ;Flush trap loc. from stack to expose loc. of fixup routine.
30 1898 11162 0107335700000605541500237567541600 0              D[MA] DEST[ALU1 AC1] TNRM $
30 1899                                                           ;Save original MA.
30 1900 11163 0107305700000605541616276452141600 0              D[JMEM] DEST[ALU1 Q] C500 $
30 1901 11164 0102315700000000014160007254114560 00             ALU[ALU1 Q] LONG ODISP $
30 1902                                                           ;Call the loser's fixup routine. It will go to either
30 1902                                                  ECC-PROCEED
30 1903                                                           ; or ECC-RET after loading MA with the failing address.
30 1904                                                           ;This keeps the fixup routine adr on the stack!
30 1905                                                  ECC-PROCEED:
30 1906 11165 0104315700020000141600403312145700 0              D[AC2] ALU[ALU1 A] DEST[JMEM] MU-PUSH  C550 JUMP[ECCFX1] $
30 1907                                                           ;A fix-up routine JUMP's here if it doesn't need to be returned
30 1907                                                  to...
30 1908                                                           ; recover the trap loc. so we can return there instead.
30 1909 11166 0107305700030605541616276453541600 0       ECC-RET:MU-POP D[JMEM] DEST[ALU1 Q] TNRM $
30 1910                                                           ;A fix-up routine PUSHJ's to here if it needs to be returned
30 1910                                                  to.
30 1911 11167 0106015700020605400020436557541700 0              D[CONST 1] ROT[MUA-ROT] ALU[ALU1 Q+D] DEST[JMEM] MU-PUSH TNRM  $
30 1911
30 1912                                                           ;Compensate for the decrementing of return address by ECCUTP.
30 1913 11170 0107313700000221405203620255754560 00    ECCFX1: D[CONST 1] ROT[35. - 14.] DEST[AR]  TNRM JUMP[ECCFX2] $
30 1914                                                           ;Fake a TRAP-ENTER code indicating that trap was from a data
30 1914                                                  fetch.
30 1915
30 1916
30 1917
```

```
31 1918
31 1919
31 1920                                                 ;GET PMA IN ALU1 Q
31 1921                                                 ;READ MAP WORD IF MAP ON, USE MA IF NOT
31 1922
31 1923 11171 010731006000000310761623670354160000     ECCPMA: D[AMEM-ABS APR-MODE] ROT[35.] COND[SIGNOFF] TNRM $
31 1924                                                         ;TEST MAP ON BIT (DOES NOT ACCOUNT FOR MAP-DISABLE
31 1924                                                  THOUGH)
31 1925 11172 010730570000060554101624656754160000             D[MA] MASK[32.] DEST[ALU1 Q] POPJLC TNRM $
31 1926                                                                 ;LOAD ALU1 Q WITH MA INCASE MAP OFF, POPJ IF MAP OFF
31 1927 11173 010731170000006374161543654354160000             D[AR] DEST[AMEM-ABS ECC-TEMP2] TNRM $
31 1928                                                                 ;SAVE AR FOR A WHILE
31 1929                                                                 ;SO CAN USE AR AND HOLD AS SCRATCH
31 1930 11174 010730570000060776001623655354160000             D[LIT 000030377000] DEST[ALU1 Q] TNRM $
31 1931                                                                 ;MASK FOR BITS 13,14 AND 19:26
31 1932 11175 010721170000060554000003655514170000             ALU[-1] DEST[EOBUS-NULL] LONG $
31 1933                                                                 ;charge up MC's Local Eobus to all 1's
31 1934                                                                 ;??? IS THIS NECESSARY???
31 1935 11176 010640570000060554161623641514160000             D[MAP-MA] ALU[ALU1 D&Q] DEST[Q] LONG $
31 1936                                                                 ;GET THOSE BITS FROM MAP-MA INTO ALU1 Q
31 1937 11177 010731370000000007161622241514560000             D[MAP-MA] ROT[28.] DEST[AR] PUSHJ[DECODE] LONG $
31 1938                                                                 ;DECODE BITS 28:31
31 1939 11200 010630570000060544761623654755160000             D[MEM] ROT[35. - 16.] ALU[ALU1 DORQ] DEST[Q] TNRM $
31 1940                                                                 ;CONTRIBUTE BITS 15 AND 16 OF PHYS ADR
31 1941 11201 010731370000000010161622241514560000             D[MAP-MA] ROT[32.] DEST[AR] PUSHJ[DECODE] LONG $
31 1942                                                                 ;DECODE BITS 32:35
31 1943 11202 010630570000060544361623654755160000             D[MEM] ROT[35. - 18.] ALU[ALU1 DORQ] DEST[Q] TNRM $
31 1944                                                                 ;CONTRIBUTE BITS 17 AND 18 OF PHYS ADR
31 1945 11203 010731370000006374161623670354160000             D[AMEM-ABS ECC-TEMP2] DEST[AR] TNRM $
31 1946                                                                 ;RESTORE AR
31 1947 11204 010630570000060554023624256754160000             D[MA] MASK[9.] ALU[ALU1 DORQ] DEST[Q] POPJ TNRM $
31 1948                                                                 ;ADD LOW ORDER 9 BITS FROM MA (WORD INDEX INTO PAGE)
31 1949                                                                 ;PMA IN ALU1 Q
31 1950
31 1951                                                 ;ACCEPTS 4 CODED LEFT JUSTIFIED BITS IN AR AND RETURNS 2 BIT # IN HOLD
31 1952
31 1953 11205 010731170000060554001503655754160000     DECODE: D[CONST 0] DEST[HOLD] TNRM $
31 1954 11206 010731006000060554161624254214160000     DECOD1: D[AR] COND[SIGNOFF] POPJ C550 $
31 1955 11207 010731370000060540361623654354160000             D[AR] ROT[1.] DEST[AR] TNRM $
31 1956 11210 011701170000224154161500254755560000             D[MEM] ALU[D+1] DEST[HOLD] JUMP[DECOD1] TNRM $
```

```
01 0061
01 0062                                                 .insert F4BASE
```

```
01 0001                                                 ;;  MAIN F4 MICROCODE -- FIRST PART
01 0002
01 0003
01 0004         4050      NORMAL = 4050           ;Beginning of ordinary code
01 0005
01 0006
01 0007                                                 ;FROM 0 TO 37, EVEN LOCS ARE TRAP0, ODD ARE TRAP1
01 0008
01 0009
01m0010                                                 .org[0]  [ xlist
01 0010 00000 540731170000060554161623254204160000        list ]NEOI $
01 0011                                                           ;SKIPs which skip come here. Fetch and dispatch on next instr.
01 0011
01 0012 00001 010731170000000001416162265431456000               PUSHJLC[TRAP1X]  $
01 0013 00002 710731170000000001416162065611456000               D[PC] DEST[MA] C150 JUMPLC[JMPFAIL] $
01 0014                                                            ;Failing JUMPs come here. Get address of instr. after the JUMP.
01 0014
01 0015 00003 610721170000000001416146065631456000               D[PC] ALU[D-1] DEST[PC] NORM JUMPLC[IOINT] $
01 0016                                                                 ;Fbus interrupts trap here (on EOI+1 cycles)
01 0017 00004 010731170000000001416162065431456000        :04    JUMPLC[ID-TO-AC] $       ;IDISP with PC<20                ;TRAM 04
01 0017
01 0018 00005 010731170000212154161622654314560000               PUSHJLC[MAPTRP]  $       ;MAP FAULT
01 0019 00006 610721170000000001416146065631456000               D[PC] ALU[D-1] DEST[PC] JUMPLC[ID-TO-AC] $       ;POPJ into an AC
01 0019
01 0020 00007 010731170000000001416162065431456000               JUMPLC[IMPURITY]  $      ;Impure thoughts trap to here.
01 0021
01 0022 00010 010731170000000001416162065431456000        :10    JUMPLC[PDL-TRAP] $       ;PDLOV detected at EOI           ;TRAM 10
01 0022
01 0023 00011 010731170000000001416162065431456000               JUMPLC[PCTRAP]  $        ;Some PC-FLAG trap bit set.
01 0024 00012 610231170000000001416140065431456000               ALU[Q] DEST[IR-ALL] JUMPLC[JRST0] $      ;JRST with non-zero AC
01 0024                                                 field
01 0025 00013 010731170000000001416162265431456000               PUSHJLC[TRAP1X]  $
01 0026
01 0027 00014 010731170000000314161620654314560000        :14    JUMPLC[.] $                                              ;TRAM 14
01 0027
01 0028 00015 010731170000221254161622654314560000               PUSHJLC[ECCTRP]  $       ;Main memory ECC errors...
01 0029 00016 010731170000000354161620654314560000               JUMPLC[.] $
01 0030 00017 610721170000000001416146065631456000               D[PC] ALU[D-1] DEST[PC] NORM JUMPLC[IOINT] $
01 0031                                                                 ;Fbus interrupts trap here (on EOI+1 cycles)
01 0032 00020 010731170000000041416162065431456000        :20    JUMPLC[.] $
01 0033 00021 010731170000000001416162265431456000               PUSHJLC[TRAP1X]  $
01 0034 00022 010731170000000045416162065431456000               JUMPLC[.] $
01 0035 00023 010731170000000001416162265431456000               PUSHJLC[TRAP1X]  $
01 0036
01 0037 00024 010731170000000051416162065431456000        :24    JUMPLC[.] $
01 0038 00025 010731170000000001416162265431456000               PUSHJLC[TRAP1X]  $
01 0039 00026 010731170000000055416162065431456000               JUMPLC[.] $
01 0040 00027 010731170000000001416162265431456000               PUSHJLC[TRAP1X]  $
01 0041
01 0042                                                 :30    D[IR] MASK[18.] ALU[D+XR] DEST[MA IR-ADR] IDISP-RQ  ID-HOLD-PC[1]
01 0042
01 0043 00030 750501170000140034045443117654160000               ID-IDX-CY[1] ID-IDX-ENB[0] ID-FROM-OBUS ID-EOI[0]
01 0043                                                 ID-PRINCIPLE[0]  $
01 0044                                                           ;INDEXING TRAPS HERE.
01 0045
01 0046 00031 010731170000000001416162265431456000               PUSHJLC[TRAP1X]  $
01 0047
01 0048                                                           D[IR] MASK[18.] ALU[D+XR] DEST[MA IR-ADR] IDISP-RQ  ID-HOLD-PC[1]
01 0048
01 0049 00032 750501170000140034045443117654160000               ID-IDX-CY[1] ID-IDX-ENB[0] ID-FROM-OBUS ID-EOI[0]
01 0049                                                 ID-PRINCIPLE[0]  $
01 0050                                                           ;INDEXING TRAPS HERE.
01 0051
01 0052 00033 010731170000000001416162265431456000               PUSHJLC[TRAP1X]  $
01 0053
01 0054 00034 010331170000000001416150045431256000        :34    DFRQ ALU[MEMAC] DEST[HOLD] JUMPLC[INDIR1] $
01 0055                                                            ;Indirect trap.
01 0056 00035 010731170000000001416162265431456000               PUSHJLC[TRAP1X]  $
01 0057 00036 010331170000000001416150045431256000               DFRQ ALU[MEMAC] DEST[HOLD] JUMPLC[INDIR1] $
01 0058                                                            ;Indirect trap.
01 0059 00037 010731170000000001416162265431456000               PUSHJLC[TRAP1X]  $
01 0060
01 0061
01m0062                                                  .use[NORMAL]  [ xlist
01 0062                                                  list ]
01 0063
01 0064                                                 INDIR1:  D[MEM] DEST[AR MA IR-23] LONG
01 0065 04050 750731370000060034161423254515160000               IDISP-RQ  ID-HOLD-PC[1] ID-FROM-OBUS ID-EOI[0] ID-PRINCIPLE[0]
01 0065                                                 $
01 0066                                                           ;Dispatch after fetching indirect pointer.
01 0067
01 0068
01 0069                                                 TRAP1X: ;Unimplemented TRAP1s come here and halt.
01 0070 04051 010731370000000001404562025571456000               D[CONST 22] DEST[AR] JUMP[HLTLOC] $
01 0071
01m0072                                                  .even
01m0072                                                 [:. \ 2 + .
01 0073                                                   ]JMPFAIL: ;Failing jumps come here.
01 0074 04052 550731170000064554161623254212160000               MA-NEOI $
01 0075                                                           ;END-OF-INST. DISPATCH, except get instr. with DF instead of IF.
01 0075
01 0076
01 0077
01 0078
```

```
                                                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                                   ;                                                             ;
                                                   ; FIXED ENTRY POINTS FOR INTERFACE WITH CONSOLE COMPUTER      ;
                                                   ; AND ANY BOOTSTRAPS IT MAY LOAD THAT MAY LOAD THIS CODE      ;
                                                   ;                                                             ;
                                                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                                    .org[4040] [ xlist
                                                      list ] ;Entry for START executing macro-code

04040 710731170000000014161620256314560000                 D[PC] DEST[MA] JUMP[MSTRT1] $
                                                            ;Macro start.  Back up PC.

                                                    :4041  ;Entry point for Memory Loading code
04041 010731170000000014161620254314560000                 JUMP[MLWAIT] $

                                                    :4042  ;Reset machine state.
04042 010731170000000014161620254314560000                 JUMP[RESET] $

                                                    :4043  ;Halts of all kinds end up here for convenience of observation.
04043 010731170000101074161620254314560000         HALTED: D[AR] JUMP[.] $
                                                           ;AR FLAG LIST:
                                                                   ;01 - RESET FINISHED
                                                                   ;11 - JRST 4
                                                                   ;22 - UNIMPLEMENTED TRAP 1
                                                                   ;33 -
                                                                   ;44 - Interrupt from illegal device.
                                                                   ;45 - Error in PI system
                                                                   ;55 -
                                                                   ;66 - ECC error.
                                                                   ;70 - MONITOR BOOTSTRAP READ OK
                                                                   ;71 - MONITOR BOOTSTRAP READ FAILED FOR SOME REASON
                                                                   ;72 - MICROCODE READ OK
                                                                   ;73 - MICROCODE READ FAILED FOR SOME REASON

                                                   :4044   ;Bootstrap Ucode from Tape
                                                    .repeat TBOOT [ CLR-DEV-FROM-INTR D[AR] DEST[ALU1 Q] JUMP[MBOOT] $ ]
04044 010730570040000014161620254314560000         [ CLR-DEV-FROM-INTR D[AR] DEST[ALU1 Q] JUMP[MBOOT] $ ]
                                                   :4045  ;Bootstrap Ucode from Disk
                                                    .repeat DBOOTF [


                                                           CLR-DEV-FROM-INTR JUMP[DBOOTA] LONG $
                                                                   ;APR-DATASW SET UP ALREADY, RESET DONE


                                                                        ]
                                                   [

04045 010731170040000014161620254114560000                 CLR-DEV-FROM-INTR JUMP[DBOOTA] LONG $
                                                                   ;APR-DATASW SET UP ALREADY, RESET DONE


                                                                        ]
                                                   ;4046 reserved for some other boot device

                                                   :4047

04047 010731170000000014161620254314560000         ITRAP:  NORM JUMP[ITRAP1] $
                                                                ;Illegal instrs. jump here for ease of debugging patches or
                                                    breaks.



                                                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;



                                                    .use[HILOC]
                                                   [ xlist
                                                     list ]

11211 010731170021060554161623654314160000         HLTLOC: SPEC[21] $       ;TURN OFF MACRO HISTORY
11212 010731170017101074161620254314560000                 SPEC[17] JUMP[HALTED] $ ;TURN OFF MICRO HISTORY AND GO AWAY

11213 010731170014000000160007260114560000         IOINT:  REENABLE-TRAP1 D[D%AMEM0] ROT[MUA-ROT] LONG ODISP $
                                                            ;I/O interrupt-- dispatch on its AMEM loc. 0

                                                   IMPURITY:
11214 710721170014000014161460256314560000                 D[PC] ALU[D-1] DEST[PC MA] REENABLE-TRAP1 NORM JUMP[MAIN] $
                                                            ;Re-fetch an instr. which was modified by its predecessor.




11215 010730170000060554141623646314160000         JMPAC:  D[PC-FLAGS] MASK[LEFT] DEST[Q] T300 $
                                                            ;We have just dispatched a Jump to an address <20 -- get PC
                                                   flags.
11216 010630170000400000040423655314170000                 D[LIT 000200000020]  ALU[DORQ] DEST[PC-FLAGS Q] $
                                                            ;Set PC-FLAG bit 10 (to trap after end-of-instr), and make
                                                   Q18:35>17
11217 640631170000000003416162325765416000                 D[IR] ALU[DORQ] EA-FROM-OBUS RE-IDISP $
                                                             ;Re-dispatch on the Jump that trapped (eff. adr. calc. already
                                                   done).
                                                             ;By ORing in Q, we insure taking the non-AC dispatch (RE-IDISP
                                                   does
                                                             ; not change any registers).  PC flag 10 will cause trap at end
                                                   of
                                                             ; Jump instr... see code at PCTRAP.


                                                   ID-TO-AC:       ;(We come here from a TRAP1 to loc. 04)
11220 010730170000060554161623646314160000                 D[PC-FLAGS] DEST[Q] T300 $
                                                            ;We have detected PC<20 at end-of-instr dispatch.
11221 010631170000060546202423655714170000                 D[CONST 1] ROT[35. - 10.] ALU[DORQ] DEST[PC-FLAGS] $
                                                            ;Set PC-FLAG bit 10 (as at JMPAC), and go execute new instr.
11222 611701170000000014161460256314560000                 D[PC] ALU[D+1] DEST[PC]  JUMP[PC<20] $
                                                            ;The aborted IDISP that came here didn't bump the PC... make it
                                                    right.

11223 010731004014060543761623646614160000         PCTRAP: REENABLE-TRAP1 D[USER/INT] ROT[15.]  C550 COND[SIGNON]  $
                                                              ;A PC-FLAG trap bit or the AROV TRAP flag was on at EOI.
11224 010730370042000014161620646314560000                 CLR-AROV-TRAP-FLAG D[PC-FLAGS] DEST[AR Q] T300  JUMPLC[AROV] $
                                                              ;Jump if PC FLAG AROV TRAP  is on.
11225 010731004000000002561620254214560000                 D[AR] ROT[10.] C550 COND[SIGNON] JUMP[PC<20] $
```

```
02 0178 11230 610721170041000014161460256314560000     PCTRPX: CLR-PC-TRAP-FLAGS D[PC] ALU[D-1] DEST[PC] JUMP[MAIN] $
02 0179                                                           ;Return to execution of trapped instr.
02 0180 11231 710721122000022461404562025621456000     PC<Z0:  D[PC] MASK[18.] ALU[D-1] DEST[MA] C550 COND[-EA<Z0] JUMP[PCTRPX]
                                                        $
02 0181                                                           ;PC was <Z0 at end of previous instr.  Is it still ?
02 0182 11232 010331104000000001416150005421456000               ALU[MEMAC] DEST[HOLD] C550 COND[INTRPT] JUMP[PC<Z0I] $
02 0183                                                             ;PC is still <Z0.  Get contents of appropriate AC, check
                                                         interrupts.
02 0184                                                 .REPEAT 1 - WAITS [       ;Some sort of bugtrap???
02 0185                                                 D[LIT 251] DEST[Q] $
02 0186                                                 D[MEM] ROT[9.] MASK[9.] ALU[D#Q] COND[-ZERO] JUMP[. + 2] $
02 0187                                                           CONT $
02 0188                                                 ];.REPEAT 1 - WAITS
02 0189 11233 540731170000060474161623254254160000               XCT-IDISP $
02 0190                                                             ;Execute the instr we just fetched from the AC.
02 0191 11234 540651170000060566202423255654170000     PC<Z0I: D[CONST 1] ROT[35. - 10.] ALU[-D&Q] DEST[PC-FLAGS] PC-HOLD-EOI $

02 0192                                                             ;An interrupt is waiting, so let it happen.  When it is
                                                         dismissed,
02 0193                                                             ; we will detect PC<Z0 again at MAIN.
02 0194 11235 610721170000Z0163416146ZZ5631456000       AROV:   D[PC] ALU[D-1] DEST[PC] PUSHJ[APRCII] $
02 0195                                                             ;Overflow trap.  Adjust PC, get APR status bits.
02 0196                                                 .REPEAT WAITS [
02 0197                                                 ;;Infamous core clobbering bug - a bug trap for something else.  May it
02 0198                                                 ;;rest in peace!!!
02 0199                                                 ;;      MAP-DISABLE D[PC] MASK[18.] DEST[Q] $
02 0200                                                 ;;        ;Turn off map so we can store safely into EXEC
02 0201                                                 ;;      D[LIT 123] DEST[MA] $
02 0202                                                 ;;        ;Where we'll store the losing PC
02 0203                                                 ;;      D[PC-FLAGS] MASK[LEFT] ALU[DORQ] DEST[MEMSTO] T350 $
02 0204                                                 ;;        ;Combine flags and PC, then store into abs. 123
02 0205                                                 ;;      MAP-ENABLE $
02m0206                                                 ];[
02w0206                                                 ;;Infamous core clobbering bug - a bug trap for something else.  May it
02m0206                                                 ;;rest in peace!!!
02m0206                                                 ;;      MAP-DISABLE D[PC] MASK[18.] DEST[Q] $
02m0206                                                 ;;        ;Turn off map so we can store safely into EXEC
02m0206                                                 ;;      D[LIT 123] DEST[MA] $
02m0206                                                 ;;        ;Where we'll store the losing PC
02m0206                                                 ;;      D[PC-FLAGS] MASK[LEFT] ALU[DORQ] DEST[MEMSTO] T350 $
02m0206                                                 ;;        ;Combine flags and PC, then store into abs. 123
02m0206                                                 ;;      MAP-ENABLE $
02 0206                                                 ].REPEAT WAITS
02 0207                                                 .REPEAT FZAROV [
02 0208                                                           D[AMEM-ABS APR-MODE] DEST[Q] $
02 0209                                                                   ; Fetch old APR state
02 0210                                                           D[CONST 20] ALU[-D&Q] DEST[AR] $
02 0211                                                                   ; Turn off arithmetic interrupts
02 0212                                                           CLR-AROV-TRAP-FLAG PUSHJ[SETMODE] $
02 0213                                                                   ; Set new mode
02 0214                                                           D[APRSTS] MASK[3] DEST[Q AR] COND[-OBUS=0] JUMP[PIGEN] C550 $
02 0215                                                             ;Generate a PI an the APR's channel.
02m0216                                                 ];[
02m0216 11236 010730170000000314161623670314160000               D[AMEM-ABS APR-MODE] DEST[Q] $
02m0216                                                                   ; Fetch old APR state
02m0216 11237 010651370000060554041623655714160000               D[CONST 20] ALU[-D&Q] DEST[AR] $
02m0216                                                                   ; Turn off arithmetic interrupts
02m0216 11240 010731170042000014161622254314560000               CLR-AROV-TRAP-FLAG PUSHJ[SETMODE] $
02m0216                                                                   ; Set new mode
02m0216 11241 010730200000203434007620261214560000               D[APRSTS] MASK[3] DEST[Q AR] COND[-OBUS=0] JUMP[PIGEN] C550 $
02m0216                                                             ;Generate a PI an the APR's channel.
02 0216                                                 ].REPEAT FZAROV
02 0217                                                 .REPEAT 1 - FZAROV [
02 0218                                                           D[AR] MASK[3] DEST[Q AR] COND[-OBUS=0] JUMP[PIGEN] C550 $
02 0219                                                             ;Generate a PI an the APR's channel.
02 0220                                                 ];.REPEAT 1 - FZAROV
02 0221 11242 010731170000000014161620254314560000               JUMP[MAIN] $
02 0222                                                             ;Otherwise, main.  (This might be a good place for a bugtrap if
02 0223                                                             ;a system is presumed to always have these things enabled.)
02 0224
02 0225
```

```
03 0226
03 0227
03 0228                                                         ;Ordinary code starts here
03 0229
03 0230 11243 010731370000Z24Z341116Z02557145600        ILLINT: D[CONST 44] DEST[AR] JUMP[HLTLOC] $
03 0231                                                           ;Come here if unknown device requests an interrupt.
03 0232
03 0233
03 0234 11244 010241170000060554156063654314170000     RESET:  ALU[0] DEST[JMEM-P] $
03 0235 11245 010241170000060554160123654314170000             ALU[0] DEST[AMEM-P] $
03 0236                                                                 ;RESET PDL PTRS
03 0237 11246 010241170000060554160423654314170000             ALU[0] DEST[PC-FLAGS] $
03 0238                                                                 ;SET EXEC
03 0239 11247 010241170000060554160723654314170000             ALU[0] DEST[MERGE] $
03 0240                                                                 ;Select 18-bit addressing.
03 0241 11250 010731170000060540406243655714170000             D[CONST 3] ROT[IDR-ROT] DEST[IDISP-REG] $
03 0242                                                                 ;SET IDISP BASE ADR
03 0243
03 0244 11251 010701170047060554161623654314160000             SPEC[BUS-RESET] $
03 0245                                                                 ;Reset all I/O hardware
03 0246 11252 010240170040060554160463654314170000             ALU[0] CLR-DEV-FROM-INTR DEST[DEV-ADR Q] $
03 0247                                                                 ;SELECT APR DEVICE
03 0248 11253 010731170010060540076003655714170000             D[CONST 31.] ROT[LLOAD-ROT] LLOAD $
03 0249                                                                   ;Init. the interrupt dispatch addresses of all devices to
03 0249                                                          ILLEGAL.
03 0250 11254 010731170000000022507003655314160000     RESET1: D[LIT ILLINT] DEST[D%AMEM0] NORM $
03 0251 11255 011200070040225314160460254314570000             ALU[Q+1] DEST[Q DEV-ADR] LOOP[RESET1] $
03 0252                                                                 ;Set all device interrupt dispatches to be illegal.
03 0253 11256 010241170040202374160462254314570000             ALU[0] DEST[DEV-ADR] PUSHJ[APRRST] $
03 0254                                                                 ;Init. device APR stuff.
03 0255 11257 010731170040220774002462254571457000             D[CONST 1] DEST[DEV-ADR] PUSHJ[MAPRST] NORM $
03 0256                                                                 ;Init MAP device
03 0257
03 0258 11260 010241370000000001416162225431456000             ALU[0] DEST[AR] PUSHJ[SETMODE] $
03 0259                                                                   ;Disable ECC traps...
03 0260                                                         .REPEAT 1 - WAITS [
03 0261                                                                 D[MASK 20.] DEST[MA] PUSHJ[MEMADR-Z3BIT-ABS] $
03 0262                                                                   ;Sweep lower part of memory to repair ECC errors...
03 0263                                                         ];.REPEAT 1 - WAITS
03 0264                                                         .REPEAT WAITS [
03 0265                                                                 D[MASK 21.] DEST[MA] PUSHJ[MEMADR-Z3BIT-ABS] $
03 0266                                                                   ;Sweep both parts of memory to repair ECC errors...
03m0267                                                         ];[
03m0267 11261 710731170000000001405362224731456000             D[MASK 21.] DEST[MA] PUSHJ[MEMADR-Z3BIT-ABS] $
03m0267                                                                   ;Sweep both parts of memory to repair ECC errors...
03 0267                                                         ].REPEAT WAITS
03 0268 11262 030731170000060554161563654713160000     RSTCCL: DF/WT D[MEM] DEST[MEMSTO] $
03 0269 11263 7107Z10060002Z5454057620Z5671456000               D[MA] MASK[Z3.] ALU[D-1] DEST[MA] COND[SIGNOFF] JUMP[RSTCCL] $
03 0270
03 0271 11264 010731370000000001402162225571456000             D[CONST 10]  DEST[AR] PUSHJ[SETMODE] NORM $
03 0272                                                                 ;Disable map, ECC off, EXTD ADR off,enable traps
03 0273
03 0274                                                         ;RESET ECC LOGGING AMEM
03 0275
03 0276 11265 010241170000000603416154365431416000             ALU[0] DEST[AMEM-ABS ECC-NXMCNT] NORM $
03 0277 11266 010241170000000605416154365431416000             ALU[0] DEST[AMEM-ABS ECC-NXMD1] NORM $
03 0278 11267 010241170000000607416154365431416000             ALU[0] DEST[AMEM-ABS ECC-NXMD2] NORM $
03 0279 11270 010241170000000611416154365431416000             ALU[0] DEST[AMEM-ABS ECC-NXMPC] NORM $
03 0280 11271 010241170000000613416154365431416000             ALU[0] DEST[AMEM-ABS ECC-HRDCNT] NORM $
03 0281 11272 010241170000000615416154365431416000             ALU[0] DEST[AMEM-ABS ECC-HRDD1] NORM $
03 0282 11273 010241170000000617416154365431416000             ALU[0] DEST[AMEM-ABS ECC-HRDD2] NORM $
03 0283 11274 010241170000000621416154365431416000             ALU[0] DEST[AMEM-ABS ECC-HRDPC] NORM $
03 0284 11275 010241170000000623416154365431416000             ALU[0] DEST[AMEM-ABS ECC-SFTCNT] NORM $
03 0285 11276 010241170000000625416154365431416000             ALU[0] DEST[AMEM-ABS ECC-SFTD1] NORM $
03 0286 11277 010241170000000627416154365431416000             ALU[0] DEST[AMEM-ABS ECC-SFTD2] NORM $
03 0287 11300 010241170000000631416154365431416000             ALU[0] DEST[AMEM-ABS ECC-SFTPC] NORM $
03 0288
03 0289
03 0290 11301 010731170000000001416162225431456000             PUSHJ[MEMADR-18BIT] $
03 0291
03 0292                                                           .repeat CTY [
03 0293                                                                 D[CONST 4] DEST[DEV-ADR] PUSHJ[CTYRST] NORM $
03m0294                                                                ]
03m0294                                                         [
03m0294 11302 010731170041000001401046225571457000             D[CONST 4] DEST[DEV-ADR] PUSHJ[CTYRST] NORM $
03 0294                                                                ] .repeat TIMER [
03 0295                                                                 D[CONST 5] DEST[DEV-ADR] PUSHJ[TIMRST] NORM $
03m0296                                                                ]
03m0296                                                         [
03m0296 11303 010731170041000001401246225571457000             D[CONST 5] DEST[DEV-ADR] PUSHJ[TIMRST] NORM $
03 0296                                                                ] .repeat TAPE [
03 0297                                                                 D[CONST 7] DEST[DEV-ADR] PUSHJ[TAPRST] NORM $
03m0298                                                                ]
03m0298                                                         [
03m0298 11304 010731170041000001401646225571457000             D[CONST 7] DEST[DEV-ADR] PUSHJ[TAPRST] NORM $
03 0298                                                                ] .repeat DSK [
03 0299                                                                 D[CONST 10] DEST[DEV-ADR] PUSHJ[DSKRST] NORM $
03m0300                                                                ]
03m0300                                                         [
03m0300 11305 010731170041000001402046225571457000             D[CONST 10] DEST[DEV-ADR] PUSHJ[DSKRST] NORM $
03 0300                                                                ] .repeat DLS [; 9 JAN 80  BO
03 0301                                                                 D[CONST DLSDEV] DEST[DEV-ADR] PUSHJ[DLSRST] NORM $
03 0302                                                                ]
03 0303                                                           .repeat DCA [; 29 JAN 81  TVR
03 0304                                                                 D[CONST DCA-UDEV] DEST[DEV-ADR] PUSHJ[DCARST] NORM $
03m0305                                                                ]
03m0305                                                         [; 29 JAN 81  TVR
03m0305 11306 010731170041000001404046225571457000             D[CONST DCA-UDEV] DEST[DEV-ADR] PUSHJ[DCARST] NORM $
03 0305                                                                ] .repeat VC [ ; 24 AUG 80  BO
03 0306                                                                 D[CONST VCDEV] DEST[DEV-ADR] PUSHJ[VCRST] NORM $
03m0307                                                                ]
03m0307                                                         [ ; 24 AUG 80  BO
03m0307 11307 010731170041000001403046225571457000             D[CONST VCDEV] DEST[DEV-ADR] PUSHJ[VCRST] NORM $
03 0307                                                                ] .repeat IMP [; 23JAN80 MLB
03 0308                                                                 D[CONST 16] DEST[DEV-ADR] PUSHJ[IMPRST] NORM $
03 0309                                                                ]
03 0310                                                           .repeat LPT [
03 0311                                                                 D[CONST LPT-UDEV] DEST[DEV-ADR] PUSHJ[LPTRST] NORM $
03m0312                                                                ];[
03m0312 11310 010731170041000001403246225571457000             D[CONST LPT-UDEV] DEST[DEV-ADR] PUSHJ[LPTRST] NORM $
03 0312                                                                ];LPT
```

```
                                               ]
                                             [;Mar80 TVR
11311 010731170040000014060462255714570000         D[CONST 30] DEST[DEV-ADR] PUSHJ[PANRST] NORM $
11312 010731170040000014060462255714570000         D[CONST 30] DEST[DEV-ADR] PUSHJ[GRNRST] NORM $
11313 010731170040000014060462255714570000         D[CONST 30] DEST[DEV-ADR] PUSHJ[ASRST] NORM $
                                               ]
11314 010731170040000014014462255714570000         D[CONST 6] DEST[DEV-ADR] PUSHJ[CLKRST] NORM $
11315 010731170040000014014462255714570000         D[CONST 6] DEST[DEV-ADR] PUSHJ[TYMRST] NORM $
11316 010731170040000014024462255714570000         D[CONST FNTDEV] DEST[DEV-ADR] PUSHJ[FNTRST] NORM $
                                             ;**** Why two calls to DLSRST??? TVR/Jun82
                                               .repeat DLS [
                                                   D[CONST DLSDEV] DEST[DEV-ADR] PUSHJ[DLSRST] $
                                                ]
11317 010731170000205354161622254314560000         PUSHJ[PI-RESET] $


11320 010731370000224234003620255714560000         D[CONST 1] DEST[AR] JUMP[HLTLOC] $


                                             ;Here is the service code for main memory loading by the CC.
11321 010731170000226434161620254314560000   MLWAIT: JUMP[.] $
                                                     ;wait for console computer to 'mil' a "D[LIT] DEST[Q]"
11322 100231170000060554161563654314160000   MLWRT:  ALU[Q] DEST[MEMSTO] SEL-PC LD-MA $
                                                     ;store the new data.
11323 010731170000060554161623654314160000           CONT $
                                                     ;prepare to fetch it (need adrs in MA in case we wait)
11324 030661000000000014161620254713560000           DF/WT D[MEM] ALU[D#Q] COND[-OBUS=0] JUMP[MLERR] $
                                                     ;Stored ok?
11325 000731170005226434161620254314560000           SPEC[PC+1] JUMP[MLWAIT] $
                                                     ;yes, do some more
11326 010731170000226554161620254314560000   MLERR:  JUMP[.] $
                                                     ;No, lose big.
11327 010731170000226554161620254314560000           JUMP[. - 1] $
                                                     ;Persist even if CC loads MI.

11330 550731170000060454161623254252160000   MSTRT1: DFRQ IDISP-RQ SEL-MA LD-MA ID-EOI[0]   $
                                                      ;Fetch and dispatch on first inst.




                                             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                             ;                                                                      ;
                                             ; COMMON EXIT POINTS FOR INSTRS. WHICH END WITH A STORE TO MEMORY     ;
                                             ;                                                                      ;
                                             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


                                             .pair
                                              [:. \ 2 + .
                                                ]SEOI:
11332 540731170000060554161623254254160000   MEMST:  EOI $
11333 540733170000060554161623054254360000           D[AR] DEST[MEMAC] EOI $
                                                      ;Here to finish a store and proceed.
                                             .pair
                                              [:. \ 2 + .
                                                ]FIX-SEOI:
11334 540731170500060554161623254255160000           TRP-CTL[TRAP-FIX RE-XCT] EOI $
11335 540733170500060554161623054255360000           TRP-CTL[TRAP-FIX RE-XCT] D[AR] DEST[MEMAC] EOI $
                                                      ;Here to finish a store and proceed, but call a fixup routine
                                                      ;if the store fails.
11336 010731170000060554161623654314160000   SNEOI:  CONT $
11337 540731170000060554161623254204160000           NEOI $

                                             ASEOI:
11340 540733170000060554161623254254160000   MEMSTA: D[AR] DEST[AC] EOI $
                                                      ;Here to finish a store and also store result in AC.
                                             CASEOI:
                                             MEMSTCA:
11341 540731170000060554160163254254160000           D[AR] COND-AC-STO EOI $
                                                      ;Here to finish a store and conditionally store result in AC.

                                             ;Here to store AR into loc. indiacted by MA and go on to next instr.
                                             ;Get here by     ... DEST[AR] MA<20 LBJUMP[SXEOI] $
                                              .pair
                                              [:. \ 2 + .
11342 010731170000226654161560254304560000      ]SXEOI:      IFRQ D[AR] DEST[MEMSTO] NORM JUMP[SEOI] $
                                                      ;MA addressed real mem.
11343 540733170000060554161623054204360000   MACSTO: D[AR] DEST[MEMAC] NEOI $
                                                      ;MA addresses an AC.

                                             MAIN: ;Come here to resume execution of macro-code at (PC).
11344 710731170014000014161622256314560000           REENABLE-TRAP1 D[PC] DEST[MA] PUSHJ[MEMADR-18BIT] $
                                                      ;Let traps happen again, put PC where we can use it, sel. 18-bit
                                              addrs
11345 550731112000076554161623254212160000           COND[-MA<20] ABORT-ADR-04 ID-COND[1]  MA-NEOI $
                                                             ;Fetch instr. and dispatch.
                                                             ;Trap to loc 04 if dispatching into AC's.

                                             ;Come here to update the AMEM copy of CPU-MODE reg. (data in AR)
                                             ;and CPU-MODE itself.  All changers of CPU-MODE should call this routine.


                                             SET-CONTEXT:  ;Special entry for updating context registers only.
11346 010731370000000314161623670314160000           D[AMEM-ABS APR-MODE] DEST[AR] $
                                                             ;Set mode to current value...
                                             SETMODE:      ;Normal entry.
11347 010731170000060540160223654114170000           D[AR] ROT[CPU-MODE-ROT] DEST[CPU-MODE] CYLEN[EOBUS] $
                                                             ;Set the CPU MODE register.
11350 010731170014000314161543654314160000           REENABLE-TRAP1 D[AR] DEST[AMEM-ABS APR-MODE] NORM $
                                                             ;Update software copy, enable traps if that mode bit is
                                              now on.
11351 010731004000000010761620254214560000           D[AR] ROT[35.] CS50 COND[SIGNON] JUMP[SETMD1] $
                                                             ;Jump if map is now on.
11352 010241170000060554160623654314170000           ALU[0] DEST[USER-CTXT] NORM $
                                                             ;Map is off. Set BOTH context registers to 0, in order
11353 010241170000060554160604254314170000           ALU[0] DEST[EXEC-CTXT] NORM     POPJ $
                                                             ; to cause memory references to be to the bottom 256k
                                              only.
11354 010731170000000764160623670314170000   SETMD1: D[AMEM-ABS MAP-UCTXT] ROT[35. - 19.] DEST[USER-CTXT] NORM $
                                                             ;Map is now on.  Set up the context registers.
```

```
03 0420 11357 010241170012ZZ2735416072025431457000              ALU[0] DEST[MERGE] MAP-ENABLE JUMP[CPOPJ] $
03 0421
03 0422                                                 MEMADR-23BIT-ABS:       ;Enter 23-bit absolute address mode.
03 0423 11360 010721170013ZZ2735400072025571457000              ALU[-1] DEST[MERGE] MAP-DISABLE JUMP[CPOPJ] $
03 0424
```

```
01 0062                                                         ;;; F4BASE.SLO contains the stuff needed by everbody...,
01 0063
01 0064                                                 .insert F4INST
```

```
01 0001
01m0002                                                 .use[NORMAL]
01m0002                                                  [ xlist
01 0003                                                    list ]
01 0004
01 0005                                                 ;;; UUOs and unassigned opcodes
01 0006
01 0007                                                 .define ILLOP [n] [.repeat n [
01 0008                                                         JUMP[ITRAP] $
01 0009                                                         JUMP[ITRAP] $
01 0010                                                    ]]
01 0011
01 0012
01 0013                                                 .define NO-OP [n] [.repeat n [
01 0014                                                         NEOI $
01 0015                                                         NEOI $
01 0016                                                    ]]
01 0017
01 0018
01 0019                                                 ; 0 is unassigned
01m0020                                                   .opcode[000]
01m0020                                                 [xlist
01m0021                                                    list  ]      ILLOP [1]
01m0021                                                 [.repeat 1 [
01m0021                                                         JUMP[ITRAP] $
01m0021                                                         JUMP[ITRAP] $
01m0021                                                    ]][
01m0021 06000 010731170000101174161620254314560000             JUMP[ITRAP] $
01m0021 06001 010731170000101174161620254314560000             JUMP[ITRAP] $
01 0022                                                    ]
01 0023                                                 ; LUUO dispatch (codes 001-0037)
01 0024                                                   .repeat 37 [
01 0025                                                         JUMP[LUUO] $
01 0026                                                         JUMP[LUUO] $
01 0027                                                 ]
01 0028                                                 [
01 0029 06002 010731170000000001416162025431456000             JUMP[LUUO] $
01 0030 06003 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06004 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06005 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06006 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06007 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06010 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06011 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06012 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06013 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06014 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06015 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06016 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06017 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06020 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06021 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06022 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06023 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06024 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06025 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06026 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06027 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06030 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06031 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06032 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06033 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06034 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06035 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06036 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06037 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06040 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06041 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06042 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06043 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06044 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06045 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06046 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06047 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06050 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06051 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06052 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06053 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06054 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06055 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06056 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06057 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06060 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06061 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                 ][
01m0027 06062 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06063 010731170000000001416162025431456000             JUMP[LUUO] $
```

```
01m0027 06067 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                         ][
01m0027 06070 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06071 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                         ][
01m0027 06072 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06073 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                         ][
01m0027 06074 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06075 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027                                                         ][
01m0027 06076 010731170000000001416162025431456000             JUMP[LUUO] $
01m0027 06077 010731170000000001416162025431456000             JUMP[LUUO] $
01 0027                                                         ]
01 0028                                                         ; MUUO dispatch (codes 040-077)
01m0029                                                           .opcode[040]
01m0029                                                         [xlist
01 0030                                                           list  ]  .repeat 40 [
01 0031                                                              JUMP[MUUO] $
01 0032                                                              JUMP[MUUO] $
01m0033                                                         ]
01m0033                                                         [
01m0033 06100 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06101 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06102 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06103 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06104 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06105 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06106 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06107 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06110 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06111 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06112 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06113 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06114 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06115 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06116 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06117 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06120 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06121 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06122 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06123 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06124 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06125 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06126 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06127 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06130 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06131 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06132 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06133 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06134 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06135 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06136 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06137 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06140 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06141 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06142 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06143 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06144 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06145 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06146 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06147 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06150 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06151 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06152 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06153 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06154 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06155 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06156 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06157 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06160 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06161 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06162 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06163 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06164 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06165 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06166 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06167 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06170 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06171 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033                                                         ][
01m0033 06172 010731170000000001416162025431456000                  JUMP[MUUO] $
01m0033 06173 010731170000000001416162025431456000                  JUMP[MUUO] $
```

```
01m0033 06176 010731170000000001416162025431456000             JUMP[MUUO] $
01m0033 06177 010731170000000001416162025431456000             JUMP[MUUO] $
01 0033                                                        ]
01 0034                                                        ; Other unassigned opcodes
01 0035
01m0036                                                           .opcode[107]  [xlist
01 0036                                                           list  ]                                          ;RDAMEM
01 0037 06216 010731170000000000160120257714570000                  D[IR] ROT[AMEM-P-ROT] DEST[AMEM-P] NORM JUMP[RDAMEM] $
01 0038                                                                ;Get eff. addr. to use as AMEM addr.
01 0039 06217 010731170000000000160120257714570000                  D[IR] ROT[AMEM-P-ROT] DEST[AMEM-P] NORM JUMP[RDAMEM] $
01m0040                                                           .reloc
01m0040                                                          [.USE[NORMAL]
01m0040                                                          [ xlist
01 0041                                                           list ]                                  ]RDAMEM:
01 0042                                                         .repeat waitsfix [
01 0043                                                                COND[USER] PUSHJ[CKIOTU] NORM $
01m0044                                                         ];[
01m0044 04053 010731124000000001416162225431456000                     COND[USER] PUSHJ[CKIOTU] NORM $
01 0044                                                         ].repeat waitsfix
01 0045 04054 540733170000060554161623274204160000                     D[AMEM-P] DEST[AC] NEOI $
01 0046                                                                  ;Return contents of AMEM loc. in AC.
01 0047
01 0048                                                         ;110-113        (MISSING FLOATING POINT STUFF)
01m0049                                                           .opcode[110]
01m0049                                                          [xlist
01 0050                                                           list  ]
01m0051                                                                ILLOP [4]
01m0051                                                          [.repeat 4 [
01m0051                                                                JUMP[ITRAP] $
01m0051                                                                JUMP[ITRAP] $
01m0051                                                           ]][
01m0051 06220 010731170000101174161620254314560000                    JUMP[ITRAP] $
01m0051 06221 010731170000101174161620254314560000                    JUMP[ITRAP] $
01m0051                                                           ][
01m0051 06222 010731170000101174161620254314560000                    JUMP[ITRAP] $
01m0051 06223 010731170000101174161620254314560000                    JUMP[ITRAP] $
01m0051                                                           ][
01m0051 06224 010731170000101174161620254314560000                    JUMP[ITRAP] $
01m0051 06225 010731170000101174161620254314560000                    JUMP[ITRAP] $
01m0051                                                           ][
01m0051 06226 010731170000101174161620254314560000                    JUMP[ITRAP] $
01m0051 06227 010731170000101174161620254314560000                    JUMP[ITRAP] $
01 0052                                                           ]
01 0053
01 0054                                                         ;114-117
01 0055                                                           .opcode[114]
01 0056                                                          [xlist
01 0056                                                           list  ]      ILLOP [4]
01 0057                                                          [.repeat 4 [
01 0058                                                                JUMP[ITRAP] $
01 0059                                                                JUMP[ITRAP] $
01m0060                                                           ]][
01m0060 06230 010731170000101174161620254314560000                    JUMP[ITRAP] $
01m0060 06231 010731170000101174161620254314560000                    JUMP[ITRAP] $
01m0060                                                           ][
01m0060 06232 010731170000101174161620254314560000                    JUMP[ITRAP] $
01m0060 06233 010731170000101174161620254314560000                    JUMP[ITRAP] $
01m0060                                                           ][
01m0060 06234 010731170000101174161620254314560000                    JUMP[ITRAP] $
01m0060 06235 010731170000101174161620254314560000                    JUMP[ITRAP] $
01m0060                                                           ][
01m0060 06236 010731170000101174161620254314560000                    JUMP[ITRAP] $
01m0060 06237 010731170000101174161620254314560000                    JUMP[ITRAP] $
01 0057                                                           ]
01 0058                                                         ;120-122        DMOVE,DMOVN,KIFIX
01 0059                                                           .opcode[120]
01 0060                                                          [xlist
01 0060                                                           list  ].repeat DMOVEFLG [
01 0061                                                                DF/WT D[MEM] DEST[AC] MA+MA+1 JUMP[DMOVE1] $
01 0062                                                                ACSEL[MA,AC] ALU[A] DEST[B] MA+MA+1 NORM JUMP[DMOVE2] $
01 0063
01 0064                                                                DF/WT MA+MA+1 D[MEM] DEST[AR] JUMP[DMOVN1] $
01 0065                                                                MA+MA+1 ALU[MEMAC] DEST[AR] JUMP[DMOVN2] $
01 0066
01 0067                                                           .reloc
01 0068                                                         DMOVE1: DF/IF D[MEM] ACSEL[AC+1] DEST[AC] NEOI $
01 0069                                                         DMOVE2: ALU[MEMAC] DEST[Q] SHORT $
01 0070                                                                ALU[Q] ACSEL[AC+1] DEST[AC] NEOI $
01 0071
01 0072                                                         DMOVN1: DF/WT D[MEM] ALU[0-D] ACSEL[AC+1] DEST[AC] CS00 COND[CRY0]
01 0072                                                          LBJUMP[DMOVN3] $
01 0073                                                         DMOVN2: ACSEL[AC,MA] ALU[0-AC] DEST[Q] CS00 COND[CRY0] LBJUMP[DMOVN4] $
01 0074
01 0075                                                          .even
01 0076                                                         DMOVN3: D[AR] ALU[NOTD] DEST[AC] NEOI $
01 0077                                                                D[AR] ALU[0-D] DEST[AC] NEOI $
01 0078                                                          .even
01 0079                                                         DMOVN4: D[AR] ALU[NOTD] DEST[AC] NORM JUMP[DMOVN5] $
01 0080                                                                D[AR] ALU[0-D] DEST[AC] NORM JUMP[DMOVN5] $
01 0081                                                         DMOVN5: ALU[Q] ACSEL[AC+1] DEST[AC] NEOI $
01 0082                                                         ];[
01 0083 06240 130733170000000001416162025471356000                    DF/WT D[MEM] DEST[AC] MA+MA+1 JUMP[DMOVE1] $
01 0084 06241 110433170000000001416162001431456000                    ACSEL[MA,AC] ALU[A] DEST[B] MA+MA+1 NORM JUMP[DMOVE2] $
01 0085
01 0086 06242 130731370000000001416162025471356000                    DF/WT MA+MA+1 D[MEM] DEST[AR] JUMP[DMOVN1] $
01 0087 06243 110331370000000001416162005431456000                    MA+MA+1 ALU[MEMAC] DEST[AR] JUMP[DMOVN2] $
01 0088
01m0089                                                           .reloc
01m0089                                                          [.USE[NORMAL]
01m0089                                                          [ xlist
01 0090 04055 560733170000060554161623154603160000                    list ]                                  ]DMOVE1:     DF/IF D[MEM]
01 0090                                                          ACSEL[AC+1] DEST[AC] NEOI $
01 0091 04056 010330170000060554161623454374160000             DMOVE2: ALU[MEMAC] DEST[Q] SHORT $
01 0092 04057 540233170000060554161623154204160000                    ALU[Q] ACSEL[AC+1] DEST[AC] NEOI $
01 0093
01 0094 04060 031713030000000001416162115461356000             DMOVN1: DF/WT D[MEM] ALU[0-D] ACSEL[AC+1] DEST[AC] CS00 COND[CRY0]
01 0094                                                          LBJUMP[DMOVN3] $
01 0095 04061 011320030000000001416162105421456000             DMOVN2: ACSEL[AC,MA] ALU[0-AC] DEST[Q] CS00 COND[CRY0] LBJUMP[DMOVN4] $
01 0096
01m0097                                                          .even
```

```
01 0101 04064 010773170000000001416162025431456000          ]DMOVN4:     D[AR] ALU[NOTD] DEST[AC] NORM JUMP[DMOVN5] $
01 0102 04065 011731700000000014161620254314560O0           D[AR] ALU[0-D] DEST[AC] NORM JUMP[DMOVN5] $
01 0103 04066 540233170000060554161623154204160O0          DMOVN5: ALU[Q] ACSEL[AC+1] DEST[AC] NEOI $
01 0082                                                     ].repeat DMOVEFLG
01 0083
01 0084                                                     .repeat 1 - DMOVEFLG [
01 0085                                                           ILLOP [2]
01 0086                                                     ];.repeat 1 - DMOVEFLG
01 0087
01m0088                                                      .opcode[123]
01m0088                                                     [xlist
01m0089                                                       list  ]      ILLOP [1]
01m0089                                                     [.repeat 1 [
01m0089                                                           JUMP[ITRAP] $
01m0089                                                           JUMP[ITRAP] $
01m0089                                                       ]][
01m0089 06246 010731170000101174161620254314560O0                JUMP[ITRAP] $
01m0089 06247 010731170000101174161620254314560O0                JUMP[ITRAP] $
01 0090                                                       ]
01 0091                                                     ;124-127        DMOVEM,DMOVNM,FIXR,FLTR
01 0092                                                      .opcode[124]
01 0093                                                     [xlist
01 0093                                                       list  ].repeat DMOVEFLG [
01 0094                                                           ALU[AC] DEST[MEMSTO] JUMP[DMVEM1] $                    ;DMOVEM
01 0095                                                           ACSEL[AC,MA] ALU[A] DEST[B] MA←MA+1 JUMP[DMVEM2] $
01 0096
01 0096
01 0097                                                           ACSEL[AC+1] ALU[0-AC] DEST[Q] COND[CRY0] LBJUMP[DMVNM1] $
01 0097                                                     ;DMOVNM
01 0098                                                           ACSEL[AC+1] ALU[0-AC] DEST[Q] COND[CRY0] LBJUMP[DMVNM1] $
01 0099
01 0100                                                      .reloc
01 0101                                                     DMVEM1: MA←MA+1 ACSEL[AC+1] ALU[AC] DEST[MEMSTO] JUMP[SNEOI] $
01 0102                                                     DMVEM2: ACSEL[AC+1] ALU[AC] DEST[Q] SHORT $
01 0103                                                           ALU[Q] DEST[MEMAC] NEOI $
01 0104                                                      .even
01 0105                                                     DMVNM1: ALU[NOTAC] DEST[MEMSTO AR] NORM COND[MA<Z0] LBJUMP[DMVNM3] $
01 0105
01 0106                                                           ALU[0-AC] DEST[MEMSTO AR] NORM COND[MA<Z0] LBJUMP[DMVNM3] $
01 0107                                                      .even
01 0108                                                     DMVNM3: MA←MA+1 ALU[Q] DEST[MEMSTO] JUMP[SNEOI] $
01 0109                                                           MA←MA+1 D[AR] DEST[MEMAC] NORM $
01 0110                                                           ALU[Q] DEST[MEMAC] NEOI $
01 0111                                                     ];[
01 0112 06250 010331170000000001416156025431456000               ALU[AC] DEST[MEMSTO] JUMP[DMVEM1] $                    ;DMOVEM
01 0113 06251 110433170000000001416162005431456000               ACSEL[AC,MA] ALU[A] DEST[B] MA←MA+1 JUMP[DMVEM2] $
01 0114
01 0114
01 0115 06252 011320030000000001416162115431456000               ACSEL[AC+1] ALU[0-AC] DEST[Q] COND[CRY0] LBJUMP[DMVNM1] $
01 0115                                                     ;DMOVNM
01 0116 06253 011320030000000001416162115431456000               ACSEL[AC+1] ALU[0-AC] DEST[Q] COND[CRY0] LBJUMP[DMVNM1] $
01 0117
01m0118                                                      .reloc
01m0118                                                     [.USE[NORMAL]
01m0118                                                     [ xlist
01 0119 04067 110331170000226754161560154314560O0            list ]                                  ]DMVEM1:     MA←MA+1
01 0119                                                     ACSEL[AC+1] ALU[AC] DEST[MEMSTO] JUMP[SNEOI] $
01 0120 04070 010330170000060554161623554374160O0          DMVEM2: ACSEL[AC+1] ALU[AC] DEST[Q] SHORT $
01 0121 04071 540233170000060554161623054204360O0                ALU[Q] DEST[MEMAC] NEOI $
01m0122                                                      .even
01m0122                                                     [:. \ Z + .
01 0123 04072 010371310000000001416156125431456000           ]DMVNM1:     ALU[NOTAC] DEST[MEMSTO AR] NORM COND[MA<Z0]
01 0123                                                     LBJUMP[DMVNM3] $
01 0124 04073 011321310000000001416156125431456000               ALU[0-AC] DEST[MEMSTO AR] NORM COND[MA<Z0] LBJUMP[DMVNM3] $
01m0125                                                      .even
01m0125                                                     [:. \ Z + .
01 0126 04074 110231170000226754161560254314560O0            ]DMVNM3:     MA←MA+1 ALU[Q] DEST[MEMSTO] JUMP[SNEOI] $
01 0127 04075 110733170000060554161623454314360O0                MA←MA+1 D[AR] DEST[MEMAC] NORM $
01 0128 04076 540233170000060554161623054204360O0                ALU[Q] DEST[MEMAC] NEOI $
01 0111                                                     ].repeat DMOVEFLG
01 0112                                                     .repeat 1 - DMOVEFLG [
01 0113                                                           ILLOP [2]
01 0114                                                     ];.repeat 1 - DMOVEFLG
01 0115
01 0116                                                     ;247 and 257 defined in f4aug if getting augment intructions
01 0117
01 0118                                                     .repeat 1 - AUGGYF [
01 0119                                                     .repeat 1 - WAITS [
01 0120                                                       .opcode[247]          ;KAFIX on appropriately modified KA-10's
01 0121                                                           ILLOP [1]
01 0122                                                     ];.repeat 1 - WAITS
01 0123
01 0124                                                       .opcode[257]
01 0125                                                           ILLOP [1]
01 0126
01m0127                                                                        ];[
01m0127                                                     .repeat 1 - WAITS [
01m0127                                                       .opcode[247]          ;KAFIX on appropriately modified KA-10's
01m0127                                                           ILLOP [1]
01m0127                                                     ];.repeat 1 - WAITS
01m0127
01m0127                                                       .opcode[257]
01m0127                                                     [xlist
01m0128                                                       list  ]      ILLOP [1]
01m0128                                                     [.repeat 1 [
01m0128                                                           JUMP[ITRAP] $
01m0128                                                           JUMP[ITRAP] $
01m0128                                                       ]][
01m0128 06536 010731170000101174161620254314560O0                JUMP[ITRAP] $
01m0128 06537 010731170000101174161620254314560O0                JUMP[ITRAP] $
01 0129                                                       ]                                                  .
01 0127                                                                        ].REPEAT 1 - AUGGYF
01 0128
```

```
02 0129
02m0130                                                 .reloc
02m0130                                                 [.USE[NORMAL]
02m0130                                                 [ xlist
02 0131 04077 710731170000000001410162225571456000       list ]                                    ]LUUO:       D[CONST 40]
02 0131                                                 DEST[MA] PUSHJ[UUOPJ] NORM  $
02 0132                                                         ;Setup MA for trap area and make mask for removing
02 0132                                                 index/indir.
02 0133 04100 010641170000060554161563657714160000     D[IR] ALU[D&Q] DEST[MEMSTO] NORM $
02 0134                                                         ;Store instruction which caused trap.
02 0135 04101 710731170000060554103623655714160000     D[CONST 41] DEST[MA] NORM $
02 0136 04102 010731170000000001416162025431256000     DFRQ JUMP[XCT11] $
02 0137                                                         ;Execute contents of (same space) location 41, usually a
02 0137                                                 JSR or
02 0138                                                         ;a HALT instruction
02 0139
02 0140                                                 ITRAP1:    ;Illegal instrs. come here from ITRAP.
02 0141 04103 710731170000000001414162025571456000     D[CONST 60] DEST[MA] JUMP[MUUO1] $
02 0142                                                   ;We will trap to loc. 60 of exec. mode.
02 0143
02 0144                                                 MUUO:      ;Monitor UUOs (opcodes 040 - 077) come here.
02 0145                                                            ;Instructions which have decided they are illegal also come
02 0145                                                 here.
02 0146 04104 710731170000060554101623655714160000     D[CONST 40] DEST[MA] $
02 0147                                                   ;Trap to location 40 of exec. mode.
02 0148 04105 010730370000060554161623646314160000     MUUO1:  D[PC-FLAGS] DEST[Q AR] $
02 0149                                                   ;Save PC flags in AR for storage by the trap instr.
02 0150 04106 010651170000000000740242225571457000     D[CONST 1] ROT[35. - 5] ALU[-D&Q] DEST[PC-FLAGS] PUSHJ[UUOPJ] $
02 0151                                                   ;Clear USER MODE and go get a mask for removing bits 13:17 from
02 0151                                                 inst.
02 0152 04107 010641170000060554161563657714160000     D[IR] ALU[D&Q] DEST[MEMSTO] NORM $
02 0153                                                         ;Store instruction which caused trap, without IDX or IND
02 0153                                                 bits.
02 0154 04110 110731170000000001416162025431456000     MA←MA+1 JUMP[PIMUUO] $
02 0155                                                         ;Get address of trap instruction.
02 0156                                                   ;PI traps come here.
02 0157 04111 010731170000060554161623654312160000     PIMUUO: DFRQ $
02 0158                                                         ;Fetch trap inst.
02 0159                                                 ;Map Fault traps come here.
02 0160 04112 710731170000060554161403654715160000     PGRTRP: D[MEM] DEST[IR-ALL MA] NORM $
02 0161                                                         ;Prepare for execution of trap inst.
02 0162 04113 010730170000000000211623655314160000     D[LIT 104] DEST[Q] NORM $
02 0163 04114 010661002000000002223620254715560000     D[MEM] ROT[9.] MASK[9.] ALU[D#Q] COND[OBUS=0] JUMP[MUJSYS] $
02 0164                                                         ;Jump if trap instruction is a JSYS.
02 0165 04115 010730170000000000055162365531416000     D[LIT 264] DEST[Q] NORM $
02 0166 04116 010661002000000002223620254715560000     D[MEM] ROT[9.] MASK[9.] ALU[D#Q] COND[OBUS=0] JUMP[MUJSR] $
02 0167                                                         ;Jump if trap instruction is a JSR.
02 0168                                                 ;;;+++  D[LIT 265] ALU[D#Q] COND[OBUS=0] JUMP[MUJSP] $
02 0169                                                 ;;;             ;Jump if trap instruction is a JSP (opcode 265)
02 0170                                                 ;+++ BLKI/BLKO would need to be handled here, if anyone ever cared.
02 0171
02 0172 04117 610721170000060554161463656374160000             D[PC] ALU[D-1] DEST[PC] SHORT $
02 0173                                                         ;Any other opcode will be XCTed and the regular code will
02 0173
02 0174                                                         ;resume.
02 0175 04120 540731170000060554160423254254170000     D[AR] DEST[PC-FLAGS] EOI $
02 0176                                                         ;Restore flags and dispatch
02 0177 04121 010730170000060554141623654314160000     MUJSYS: D[AR] MASK[LEFT] DEST[Q] NORM  $
02 0178                                                         ;Place original PC-FLAGS where JSYS wants them.
02 0179 04122 010731002000000000662362025771456000     D[IR] ROT[33] MASK[11] COND[OBUS=0] JUMP[JSYSA] $
02 0180                                                         ;Jump if it's an EXEC-type JSYS
02 0181 04123 010731170000000001416162025431456000     NORM JUMP[JSYS2] $
02 0182
02 0183 04124 010730170000000001414162025431456000     MUJSR:  D[AR] MASK[LEFT] DEST[Q] NORM JUMP[JSR1] $
02 0184                                                         ;Place original PC-FLAGS where JSR wants them.
02 0185
02 0186 04125 010730170000060545677624247314160000     UUOPJ:  D[MASK 37] ROT[27] DEST[Q] POPJ NORM $
02 0187
02 0188
```

```
03 0189
03 0190
03 0191                                         ;UMOVE group faked as special XCT.
03 0192
03m0193                                         .OPCODE[100]    [xlist
03 0193                                           list ]                                        ;UMOVE
03 0194 06200 010730126000000001416162125771456000       D[IR] DEST[Q] NORM COND[-USER] LBJUMP[UMOVX] NORM $
03 0195 06201 010730126000000001416162125771456000       D[IR] DEST[Q] NORM COND[-USER] LBJUMP[UMOVX] NORM $
03 0196                                                         ;GET UNMODIFIED (VIA NDX ETC TRAPS) INSTR IN Q
03 0197
03m0198                                         .OPCODE[101]    [xlist
03 0198                                           list ]                                        ;UMOVEI
03 0199 06202 010730126000000001416162125771456000       D[IR] DEST[Q] NORM COND[-USER] LBJUMP[UMOVX] NORM $
03 0200 06203 010730126000000001416162125771456000       D[IR] DEST[Q] NORM COND[-USER] LBJUMP[UMOVX] NORM $
03 0201                                                         ;GET UNMODIFIED (VIA NDX ETC TRAPS) INSTR IN Q
03 0202
03m0203                                         .OPCODE[102]    [xlist
03 0203                                           list ]                                        ;UMOVEM
03 0204 06204 010730126000000001416162125771456000       D[IR] DEST[Q] NORM COND[-USER] LBJUMP[UMOVX] NORM $
03 0205 06205 010730126000000001416162125771456000       D[IR] DEST[Q] NORM COND[-USER] LBJUMP[UMOVX] NORM $
03 0206                                                         ;GET UNMODIFIED (VIA NDX ETC TRAPS) INSTR IN Q
03 0207
03m0208                                         .OPCODE[103]    [xlist
03 0208                                           list ]                                        ;UMOVES
03 0209 06206 010730126000000001416162125771456000       D[IR] DEST[Q] NORM COND[-USER] LBJUMP[UMOVX] NORM $
03 0210 06207 010730126000000001416162125771456000       D[IR] DEST[Q] NORM COND[-USER] LBJUMP[UMOVX] NORM $
03 0211                                                         ;GET UNMODIFIED (VIA NDX ETC TRAPS) INSTR IN Q
03m0212                                          .reloc
03m0212                                         [.USE[NORMAL]
03m0212                                         [ xlist
03 0213                                           list ]                                    );LOAD MEM BUS WITH INSTR AND
03 0213                                         Q WITH MAGIC BITS FOR MERGE WITH
03 0214                                         ;SPECIAL XCT STUFF
03m0215                                          .pair
03m0215                                         [:. \ 2 + .
03 0216 04126 010731170000102114161620254314456000       1UMOVX:       JUMP[MUUO] NORM $
03 0217                                                         ;USER: LOSE
03 0218 04127 010600170200000000000162365531416000       D[LIT 1000000000000] ALU[D+Q] DEST[Q] NORM $
03 0219                                                         ;CONVERT TO MOVX GROUP
03 0220 04130 010651170000060544477503655714160000       D[CONST 37] ROT[18.] ALU[-D&Q] DEST[HOLD] NORM $
03 0221                                                         ;Addr calulation is already done. Don't confuse XCTP
03 0221                                         code.
03 0222 04131 010730170000000014013620255714560000       D[CONST 5] DEST[Q] JUMP[XCTMRG] NORM $
03 0223                                                         ;FAKE XCT 5,[MOVX AC,EA] (EXEC MODE EFFECTIVE ADR
03 0223                                         CALC!!!!!)
03 0224                                                         ;THIS IS WRONG ACCORDING TO BBN SPEC!!!!!! (WANTS 15 NOT
03 0224                                         5)
03 0225                                                         ;MERGE WITH SPECIAL XCT CODE
03 0226
03 0227
```

```
04 0228
04 0229
04m0230                                                 .opcode[104]   [xlist
04 0230                                                  list  ]                                                ;JSYS
04 0231 06210 010731002000000006623621257714560000             D[IR] ROT[33] MASK[11] COND[OBUS=0] LBJUMP[JSYS1] $
04 0232 06211 010731002000000006623621257714560000             D[IR] ROT[33] MASK[11] COND[OBUS=0] LBJUMP[JSYS1] $
04 0233
04m0234                                                  .reloc
04m0234                                                 [.USE[NORMAL]
04m0234                                                 [ xlist
04m0235                                                   list ]                                ]  .pair
04m0235                                                 [:. \ 2 + .
04 0236 04132 010730170000000001414162024631456000               ]JSYS1:      D[PC-FLAGS] MASK[LEFT] DEST[Q] JUMP[JSYS2] $ ; User JSYS
04 0236
04 0237                                                 .REPEAT 1 - WAITS [
04 0238                                                         D[PC-FLAGS] MASK[LEFT] DEST[Q] $        ; Exec JSYS
04 0239                                                 ];.REPEAT 1 - WAITS
04 0240                                                 .REPEAT WAITS [
04 0241                                                 ;EX JSYS is disabled in WAITS' BBN pager.  We simulate the same.
04 0242                                                 ;User program doing JSYS to EXEC, trap as illegal instruction
04 0243                                                 ;Exec doing JSYS is escape to funny WAITS instructions in KL (like the
                                                         checksum
04 0244                                                 ;   instruction.  NOTE: Unlike TENEX, WAITS cares what AC field is.
04 0245                                                         D[IR] MASK[17.] DEST[Q] COND[MA<20] JUMP[ITRAP] C550 $
04 0246                                                                 ;Illegal if effective address is an AC
04 0247                                                         D[PC-FLAGS] ROT[5] COND[OBUS<0] JUMP[MUUO] LONG $
04 0248                                                                 ;Skip if not in user mode.  Don't use USER condition as
                                                         we
04 0249                                                                 ;want to do it from EXEC using XCT mapped.
04 0250                                                         D[CONST 20] ALU[D-Q] COND[OBUS=0] JUMP[CHKINS] C550 $
04 0251                                                                 ;JSYS 20 = CHKINS (Checksum instruction)
04 0252                                                         JUMP[ITRAP] $
04m0253                                                 ];[
04m0253                                                 ;EX JSYS is disabled in WAITS' BBN pager.  We simulate the same.
04m0253                                                 ;User program doing JSYS to EXEC, trap as illegal instruction
04m0253                                                 ;Exec doing JSYS is escape to funny WAITS instructions in KL (like the
04m0253                                                  checksum
04m0253                                                 ;   instruction.  NOTE: Unlike TENEX, WAITS cares what AC field is.
04m0253 04133 010730110000101174043620257614560000             D[IR] MASK[17.] DEST[Q] COND[MA<20] JUMP[ITRAP] C550 $
04m0253                                                                 ;Illegal if effective address is an AC
04m0253 04134 010731004000102101361620246114560000             D[PC-FLAGS] ROT[5] COND[OBUS<0] JUMP[MUUO] LONG $
04m0253                                                                 ;Skip if not in user mode.  Don't use USER condition as
                                                         we
04m0253                                                                 ;want to do it from EXEC using XCT mapped.
04m0253 04135 011621002000000014041620255614560000             D[CONST 20] ALU[D-Q] COND[OBUS=0] JUMP[CHKINS] C550 $
04m0253                                                                 ;JSYS 20 = CHKINS (Checksum instruction)
04m0253 04136 010731170000101174161620254314560000             JUMP[ITRAP] $
04 0253                                                 ].REPEAT WAITS
04 0254 04137 010631370043060554045503656314160000     JSYSA:  CLR-HALF D[PC] MASK[18.] ALU[DORQ] DEST[HOLD AR] $
04 0255                                                                 ;Save old PC where we can store it from.
04 0256 04140 010641002000103047403620255714560000             D[CONST 1] ROT[36] ALU[D&Q] COND[OBUS=0] JUMP[. + 2] $
04 0257                                                                 ;If we are in user mode,
04 0258 04141 010731170000060554000403655714170000              SET-TEMP-EXEC  [ D[CONST 00] DEST[MAP-USER-SR] ]$
04 0258
04 0259                                                                 ; turn it off temporarily.
04 0260 04142 010730170000000002001623655314160000             D[LIT 1000] DEST[Q] NORM $
04 0261                                                                 ;Pointer word is at 1000+MA.
04 0262 04143 710631170000060554161623656714160000             D[MA] ALU[DORQ] DEST[MA] $
04 0263                                                                 ;Also, setup for map trap
04 0264 04144 010730170000060554161623646312160000             DFRQ D[PC-FLAGS] DEST[Q] $
04 0265                                                                 ;Fetch pointer word.
04 0266 04145 710731170000060554045463654715160000             D[MEM] MASK[18.] DEST[PC MA] $       . #
04 0267                                                                 ;RH is new PC... place in MA for fetch below.
04 0268                                                                 ;If we take a page trap here, store of PC is aborted.
04 0269                                                 .REPEAT WAITSFIX [
04 0270                                                         D[PC-FLAGS] ROT[5] C550 COND[OBUS<0] JUMP[JSYSA2] $
04 0271                                                                 ;If we came from exec mode, then we...
04 0272                                                         D[CONST 1] ROT[34] ALU[DORQ] DEST[Q] JUMP[JSYSA2] $
04 0273                                                                 ; ...TURN ON FLAG BIT 7
04 0274                                                         .ODD
04 0275                                                 JSYSA2: D[CONST 1] ROT[36] ALU[-D&Q] DEST[PC-FLAGS] PUSHJ[JSYSA3] $
04 0276                                                                 ; Turn off USER
04 0277                                                                 ; Prepare for map trap
04 0278                                                          NORM JUMP[ECC-PROCEED] $
04 0279                                                                 ;Nothing special for ECC trap
04 0280                                                 JSYSFX:  D[AR] DEST[PC-FLAGS] $
04 0281                                                                 ;Fixup code for either User or Exec JSYS
04 0282                                                                 ;Repair flags
04 0283                                                          D[AR] MASK[18.] DEST[PC] PUSHJ[MAP-RET] $
04 0284                                                                 ;And PC
04 0285                                                          jump[.] $
04 0286                                                                 ;We should never get here!!!
04 0287                                                 JSYSA3: DFRQ D[MEM] ROT[18.] MASK[18.] DEST[MA STRT-WRT] JUMP[FIX-SEQI] $
04 0287
04 0288                                                                 ;Fetch target instr. Store old PC via left half of
04 0288                                                  pointer.
04 0289                                                                 ;Do fixup if store fails
04 0290
04 0291                                                 ; User JSYS
04 0292                                                 JSYS2:  DFRQ D[PC] MASK[18.] ALU[DORQ] DEST[HOLD AR] JUMP[JSYS2A] $
04 0293                                                                 ;Fetch pointer, prepare to store PC & flags.
04 0294                                                         .ODD
04 0295                                                 JSYS2A: CLR-HALF D[MEM] MASK[18.] DEST[PC MA] NORM PUSHJ[JSYS2B] $
04 0296                                                                 ;Jump to (right half) +++ could be AC !!
04 0297                                                          NORM JUMP[ECC-PROCEED] $
04 0298                                                                 ;Nothing special for ECC trap
04 0299                                                          NORM JUMP[JSYSFX] $
04 0300                                                                 ;Fixup is same for User or Exec JSYS
04 0301                                                 JSYS2B: DFRQ D[MEM] ROT[18.] MASK[18.] DEST[MA STRT-WRT]
04 0302                                                 ;;;interacts buggily with old BLT                EA<20 LBJUMP[FIX-SEQI] $
04 0302
04 0303                                                                         EA<20 LBJUMP[SEQI] $ ;Works with old BLT.
04 0304                                                                 ;The DFRQ fetches the target instr.  Start the write and
04 0304                                                  exit.
04m0305                                                 ];[
04m0305 04146 010731004000000001361620246214560000             D[PC-FLAGS] ROT[5] C550 COND[OBUS<0] JUMP[JSYSA2] $
04m0305                                                                 ;If we came from exec mode, then we...
04m0305 04147 010630170000000000700362025571456000             D[CONST 1] ROT[34] ALU[DORQ] DEST[Q] JUMP[JSYSA2] $
04m0305                                                                 ; ...TURN ON FLAG BIT 7
04m0305                                                         .ODD
```

```
04m0306 04152 010731170000223534161620254314560000             NORM JUMP[ECC-PROCEED] $
04m0306                                                                 ;Nothing special for ECC trap
04m0306 04153 010731170000060554160423654314170000       JSYSFX:  D[AR] DEST[PC-FLAGS] $
04m0306                                                                 ;Fixup code for either User or Exec JSYS
04m0306                                                                 ;Repair flags
04m0306 04154 610731170000212374045462254314560000             D[AR] MASK[18.] DEST[PC] PUSHJ[MAP-RET] $
04m0306                                                                 ;And PC
04m0306 04155 010731170000103334161620254314560000              jump[.] $
04m0306                                                                 ;We should never get here!!!
04m0306 04156 710731170000226704445640254713560000       JSYSA3: DFRQ D[MEM] ROT[18.] MASK[18.] DEST[MA STRT-WRT] JUMP[FIX-SEQI] $
04m0306
04m0306                                                                 ;Fetch target instr. Store old PC via left half of
04m0306                                                  pointer.
04m0306                                                                 ;Do fixup if store fails
04m0306
04m0306                                                  ; User JSYS
04m0306 04157 010631370000000001404550025631256000      JSYS2:  DFRQ D[PC] MASK[18.] ALU[DORQ] DEST[HOLD AR] JUMP[JSYS2A] $
04m0306                                                                 ;Fetch pointer, prepare to store PC & flags.
04m0306                                                         .ODD
04m0306                                                  [:. + 1 - (. \ 2)
04m0307 04161 710731170041300001404546225471556000        ]JSYS2A:     CLR-HALF D[MEM] MASK[18.] DEST[PC MA] NORM PUSHJ[JSYS2B]
04m0307                                                  $
04m0307                                                                 ;Jump to (right half) +++ could be AC !!
04m0307 04162 010731170000022353416162025431456000              NORM JUMP[ECC-PROCEED] $
04m0307                                                                 ;Nothing special for ECC trap
04m0307 04163 010731170000103274161620254314560000              NORM JUMP[JSYSFX] $
04m0307                                                                 ;Fixup is same for User or Exec JSYS
04m0307                                                  JSYS2B: DFRQ D[MEM] ROT[18.] MASK[18.] DEST[MA STRT-WRT]
04m0307                                                  ;;;interacts buggily with old BLT                EA<20 LBJUMP[FIX-SEQI] $
04m0307
04m0307 04164 710731120000022664444564125471356000                             EA<20 LBJUMP[SEQI] $ ;Works with old BLT.
04m0307                                                                 ;The DFRQ fetches the target instr.  Start the write and
04m0307                                                  exit.
04 0305                                                 ].REPEAT WAITSFIX
04 0306                                                 .REPEAT 1 - WAITSFIX [
04 0307                                                         D[PC-FLAGS] ROT[5] C550 COND[OBUS<0] JUMP[. + 2] $
04 0308                                                                 ;If we came from exec mode, then we...
04 0309                                                         D[CONST 1] ROT[34] ALU[DORQ] DEST[Q] $
04 0310                                                                 ; ...TURN ON FLAG BIT 7
04 0311                                                         D[CONST 1] ROT[36] ALU[-D&Q] DEST[PC-FLAGS] $
04 0312                                                                 ; Turn off USER
04 0313                                                         DFRQ D[MEM] ROT[18.] MASK[18.] DEST[MA STRT-WRT] JUMP[SEQI] $
04 0314                                                                 ;Fetch target instr. Store old PC via left half of
04 0314                                                  pointer.                                        . #
04 0315
04 0316                                                 ; User JSYS
04 0317                                                 JSYS2:  DFRQ D[PC] MASK[18.] ALU[DORQ] DEST[HOLD AR] $
04 0318                                                                 ;Fetch pointer, prepare to store PC & flags.
04 0319                                                         CLR-HALF D[MEM] MASK[18.] DEST[PC MA] NORM $
04 0320                                                                 ;Jump to (right half) +++ could be AC !!
04 0321                                                         DFRQ D[MEM] ROT[18.] MASK[18.] DEST[MA STRT-WRT] EA<20
04 0321                                                  LBJUMP[SEQI] $
04 0322                                                                 ;The DFRQ fetches the target instr.  Start the write and
04 0322                                                  exit.
04 0323                                                 ];.REPEAT 1 - WAITSFIX
04 0324
04 0325
```

```


                                                          .REPEAT 1 - WAITS [
                                                          ;**** If you want ADJSP, you'll have to move XMOVE somewhere else!
                                                           TVR/Jun82

                                                            .opcode[105]
                                                           ;XMOVE
                                                                  JUMP[XMOVE1] $
                                                                  JUMP[XMOVE1] $

                                                                  DFRQ ALU[MEMAC] DEST[HOLD] JUMP[XMVEM1] $
                                                                  DFRQ ALU[MEMAC] DEST[HOLD] JUMP[XMVEM1] $

                                                            .reloc
                                                          XMOVE1: UIOTRP[MUUO] $
                                                                  ALU[AC] DEST[MA] PUSHJ[MEMADR-23BIT-ABS] $
                                                                  DFRQ D[IR] MASK[18.] DEST[MA] PUSHJ[MEMADR-18BIT] $
                                                                  D[MEM] SMAC $

                                                          XMVEM1: UIOTRP[MUUO] $
                                                                  ALU[AC] DEST[MA] PUSHJ[MEMADR-23BIT-ABS] $
                                                                  D[MEM] DEST[MEMSTO] $
                                                                  JUMP[MAIN] $


                                                          ];.REPEAT 1 - WAITS

                                                          .REPEAT WAITS [
                                                          ;*** What can i do instead of JUMP[MAIN] below?   TVR/Jun82
                                                             .opcode[105];        ADJSP
                                                          ADJSP:  D[IR] ROT[22] MASK[LEFT] ALU[D+AC] DEST[Q]
                                                                                  COND[OBUS<0] LBJUMP[ADJSP1] C550$
                                                                  D[IR] ROT[22] MASK[LEFT] ALU[D+AC] DEST[Q]
                                                                                  COND[OBUS<0] LBJUMP[ADJSP1] C550$
                                                                          ;Jump and skip if left result is negative
                                                             .reloc

                                                                  .even
                                                          ADJSP1: D[IR] MASK[22] ALU[D+AC] DEST[AR] JUMP[ADJSP2] NORM $   ;Add
                                                           right half
                                                                          ;Left result positive
                                                                  D[IR] MASK[22] ALU[D+AC] DEST[AR] NORM $
                                                                          ;Left result is negative.  Cheak right side.
                                                                          ;Add right half
                                                                  D[MASK 22] ALU[-D&Q] DEST[Q] SHORT $
                                                                          ;Isolate left half
                                                                  D[AR] MASK[22] ALU[DORQ] DEST[O-AC AR] NORM $
                                                                          ;Merge halves and store
                                                                  D[IR] COND[-OBUS18] JUMP[MAIN] C550 $
                                                                          ;If E was positive, it wasn't an overflow (just a bad
                                                           idea)
                                                                          ;Start fetching next instruction
                                                                  D[AR] COND[OBUS<0] JUMP[MAIN] C550 $
                                                                          ;If original was negative, we're OK.  Start doing next
                                                                          ;instruction if no sign changed in left half
                                                                  JUMP[ADJSPO] NORM $
                                                                          ;ADJSP got a PDLOV
                                                          ;ADJSP left result positive
                                                          ADJSP2: D[MASK 22] ALU[-D&Q] DEST[Q] SHORT $
                                                                          ;Isolate left half
                                                                  D[AR] MASK[22] ALU[DORQ] DEST[O-AC AR] $
                                                                          ;Merge halves and store
                                                                  D[IR] COND[OBUS18] JUMP[MAIN] $
                                                                          ;If E was negative, it wasn't an overflow (just a bad
                                                           idea)
                                                                          ;Start fetching next instruction
                                                                  D[AR] COND[-OBUS<0] JUMP[MAIN] C550 $
                                                                          ;If original was positive, we're OK.  Start doing next
                                                                          ;instruction if no sign changed in left half
                                                          ADJSPO: PUSHJ[PDLO] NORM $
                                                                          ;ADJSP got a PDLOV
                                                                  JUMP[MAIN] $

                                                            .opcode[106]
                                                                  DFRQ ALU[MEMAC] DEST[HOLD] COND[-USER] LBJUMP[XMVEM1] $
                                                                  DFRQ ALU[MEMAC] DEST[HOLD] COND[-USER] LBJUMP[XMVEM1] $


                                                            .reloc
                                                                  .EVEN
                                                          XMVEM1:
                                                          ];[
                                                          ;*** What can i do instead of JUMP[MAIN] below?   TVR/Jun82
                                                             .opcode[105]; [xlist
                                                             list  ]      ADJSP
                                                          ADJSP:  D[IR] ROT[22] MASK[LEFT] ALU[D+AC] DEST[Q]
06212 01050000400000000454162125761456000                                         COND[OBUS<0] LBJUMP[ADJSP1] C550$
                                                                  D[IR] ROT[22] MASK[LEFT] ALU[D+AC] DEST[Q]
06213 01050000400000000454162125761456000                                         COND[OBUS<0] LBJUMP[ADJSP1] C550$
                                                                          ;Jump and skip if left result is negative
                                                             .reloc
                                                           [.USE[NORMAL]
                                                           [ xlist
                                                             list ]                                   ]
                                                                  .even
                                                           [:. \ 2 + .
04166 01050137000000001404562025771456000                    ]ADJSP1:     D[IR] MASK[22] ALU[D+AC] DEST[AR] JUMP[ADJSP2] NORM $
                                                           ;Add right half
                                                                          ;Left result positive
04167 01050137000006055404562365771416000                         D[IR] MASK[22] ALU[D+AC] DEST[AR] NORM $
                                                                          ;Left result is negative.  Check right side.
                                                                          ;Add right half
04170 01065017000006055404562364737416000                         D[MASK 22] ALU[-D&Q] DEST[Q] SHORT $
                                                                          ;Isolate left half
04171 01063237000006055404562365431416000                         D[AR] MASK[22] ALU[DORQ] DEST[O-AC AR] NORM $
                                                                          ;Merge halves and store
04172 01073102200022711416162025761456000                         D[IR] COND[-OBUS18] JUMP[MAIN] C550 $
                                                                          ;If E was positive, it wasn't an overflow (just a bad
                                                           idea)
                                                                          ;Start fetching next instruction
04173 [illegible]                                                 D[AR] COND[OBUS<0] JUMP[MAIN] C550 $
```

```
05m0404                                                 ;ADJSP left result positive
05m0404 04175 010650170000060554045623647374160000     ADJSP2: D[MASK 22] ALU[-D&Q] DEST[Q] SHORT $
05m0404                                                         ;Isolate left half
05m0404 04176 010632370000060554045623654314160000             D[AR] MASK[22] ALU[DORQ] DEST[O-AC AR] $
05m0404                                                         ;Merge halves and store
05m0404 04177 010731020000227114161620257714560000             D[IR] COND[OBUS18] JUMP[MAIN] $
05m0404                                                         ;If E was negative, it wasn't an overflow (just a bad
05m0404                                                  idea)
05m0404                                                         ;Start fetching next instruction
05m0404 04200 010731006000227114161620254214560000             D[AR] COND[-OBUS<0] JUMP[MAIN] CS50 $
05m0404                                                         ;If original was positive, we're OK.  Start doing next
05m0404                                                         ;instruction if no sign changed in left half
05m0404 04201 010731170000000014161622254314560000     ADJSP0: PUSHJ[PDLO] NORM $
05m0404                                                         ;ADJSP got a PDLOV
05m0404 04202 010731170000227114161620254314560000             JUMP[MAIN] $
05m0404
05m0404                                                  .opcode[106]
05m0404                                                 [xlist
05m0405 06214 010331126000000014161501054312560000       list  ]       DFRQ ALU[MEMAC] DEST[HOLD] COND[-USER] LBJUMP[XMVEM1] $
05m0405
05m0405 06215 010331126000000014161501054312560000             DFRQ ALU[MEMAC] DEST[HOLD] COND[-USER] LBJUMP[XMVEM1] $
05m0405
05m0405                                                  .reloc
05m0405                                                 [.USE[NORMAL]
05m0405                                                 [ xlist
05m0406                                                   list ]                                    ]     .EVEN
05m0406                                                 [:. \ 2 + .
05m0407                                                   ]XMVEM1:
05 0402                                                 ].REPEAT WAITS
05 0403                                                 .REPEAT WAITSFIX {
05 0404                                                         COND[USER] PUSHJ[CKIOTU] $
05 0405                                                         ;Don't allow arbitrary writes unless in IOT-USER mode!!!
05 0405
05m0406                                                 };[
05m0406 04204 010731124000000014161622254314560000             COND[USER] PUSHJ[CKIOTU] $
05m0406                                                         ;Don't allow arbitrary writes unless in IOT-USER mode!!!
05m0406
05 0406                                                 ].REPEAT WAITSFIX
05 0407 04205 710331170000227414161622254314560000             ALU[AC] DEST[MA] PUSHJ[MEMADR-23BIT-ABS] $
05 0408 04206 010731170000060554161563654715160000             D[MEM] DEST[MEMSTO] $
05 0409 04207 010731170000227114161620254314560000             JUMP[MAIN] $
05 0410
```

```
06 0411                                                 ;;; Byte instructions
06 0412                                                   .opcode[133]   [xlist
06m0413                                                    list  ]                                                ;IBP
06 0413
06 0414 06266 030730315000000014161621254713560000             DF/WT R-M-W D[MEM] DEST[Q AR] COND[AC=0] LBJUMP[IBP1] $
06 0415 06267 010330314000000014161621054314560000             ALU[MEMAC] DEST[Q AR] COND[AC=0] LBJUMP[IBPA1] $
06 0416
06m0417                                                    .reloc
06m0417                                                  [.USE[NORMAL]
06m0417                                                  [ xlist
06m0418                                                    list ]                                 ]  .pair
06m0418                                                  [:. \ 2 + .
06 0419 04210 010731170000000014161620254314560000               ]IBP1:       JUMP[ADJBP] $           ; non-zero AC is an ADJBP
06 0420 04211 010731370000060543015623654314160000                    D[AR] ROT[12.] MASK[6] DEST[AR] $
06 0421                                                                ;Get S field
06 0422 04212 011611032000000007561561254304560000                    IFRQ D[AR] ROT[30.] ALU[Q-D] DEST[MEMSTO] COND[-CRY0]
                                                         LBJUMP[IBP2] $
06 0423                                                                ;Subtract from P (in place) and store, checking for
06 0423                                                  overflow
06m0424                                                    .pair
06m0424                                                  [:. \ 2 + .
06 0425 04214 540731170000060554161623254254160000               ]IBP2:       EOI $           ; No overflow, just store and disp
06 0426 04215 010640170000060554075623647314160000     IBP3:   D[MASK 30.] ALU[D&Q] DEST[Q] $
06 0427                                                                ;ptr will overflow, fix it up
06 0428 04216 011600170000060547511503655714160000             D[CONST 44] ROT[30.] ALU[D+Q+1] DEST[Q HOLD] $
06 0429                                                                ;New P of 44, add 1 to adr
06 0430 04217 010731000000000014045620254615560000             D[MEM] MASK[18.] COND[-ZERO] JUMP[IBP31] C550 $
06 0431                                                                ;JUMP IF ADR NOT 0 (DID NOT OVERFLOW)
06 0432 04220 011610170000060544403623655714160000             D[CONST 1] ROT[18.] ALU[Q-D] DEST[Q] NORM $
06 0433                                                                ;BACKUP LEFT HALF CAUSING WRAP-AROUND TO 0 FROM 777777
06 0434                                                                ;CARRY CAN'T PROPAGATE PAST BIT 0 CAUSE OF 44 IN B0-5
06 0435 04221 011611310000226647561561254304560000     IBP31:  IFRQ D[AR] ROT[30.] ALU[Q-D] S%AC  [ IFRQ DEST[MEMSTO AR]
                                                         COND[MA<20] LBJUMP[SEOI] ]$
06 0436                                                                ;Store it
06 0437
06 0438                                                 ; Same as above with ptr in an AC
06m0439                                                    .pair
06m0439                                                  [:. \ 2 + .
06 0440 04222 010731170000000014161620254314560000               ]IBPA1:      JUMP[ADJBP] $           ; non-zero AC is an ADJBP
06 0441 04223 010731370000060543015623654314160000             D[AR] ROT[12.] MASK[6] DEST[AR] $
06 0442                                                                ;Get S field
06 0443 04224 011613032000104327561620054304760000             IFRQ D[AR] ROT[30.] ALU[Q-D] DEST[MEMAC] COND[-CRY0] JUMP[IBP3] $
06 0443
06 0444                                                                ;Subtract from P (in place) and store, checking for
06 0444                                                  overflow
06 0445 04225 540731170000060554161623254254160000             EOI $
06 0446
06 0447 04226 010731170000102114161620254314560000     ADJBP:  JUMP[MUUO] $            ; Not implemented yet
06 0448                                                                ;WHEN IT IS, WATCH OUT FOR CARRY INTO
06 0448                                                  INDEX AC
06 0449                                                                ;WHILE INCREMENTING OR DECREMENTING ADR.
06 0449
06 0450
06 0451
```

```
07 0452
07m0453                                                  .opcode[134]  [xlist
07 0453                                                   list  ]                                                    ;ILDB
07 0454 06270 630730237000000001416142125471356000              DF/WT R-M-W D[MEM] DEST[Q AR IR-23] COND[-HALF] LBJUMP[ILDB1] $
07 0455 06271 610330236000000001416142105431456000              ALU[MEMAC] DEST[Q AR IR-23] COND[-HALF] LBJUMP[ILDBA1] $
07m0456                                                  .reloc
07m0456                                                  [.USE[NORMAL]
07m0456                                                  [ xlist
07 0457                                                   list ]                                  ]; Ptr in memory
07m0458                                                   .pair
07m0458                                                  [:. \ 2 + .
07 0459 04230 010731170000104643014500254314570000       ]ILDB1:      D[AR] ROT[14] MASK[6] DEST[MASKR] JUMP[. + 2] $ ; HALF
07 0459                                                  was set
07 0460 04231 010731370000000003014502254314570000              D[AR] ROT[14] MASK[6] DEST[AR MASKR] PUSHJ[IBPX] $
07 0461                                                                  ; Get S field and call common code
07 0462 04232 710731370000000001416162025471556000              D[MEM] DEST[AR MA] NORM JUMP[ILDB4] $
07 0463
07 0464                                                  ; Data in memory
07m0465                                                   .odd
07m0465                                                  [:. + 1 - (. \ 2)
07 0466 04233 011611170000000001414442254312570000       ]ILDB2:      DFRQ D[AR] ROT[6] MASK[6] ALU[Q-D] DEST[ROTR]
07 0466                                                  PUSHJ[ILDB3] $
07 0467                                                           ;36.-P to ROTR, fetch data, setup trap processing address.
07 0468 04234 010731170000223534161620254314560000               NORM JUMP[ECC-PROCEED] $
07 0469                                                           ;ECC trap -- nothing special.
07 0470 04235 010731170000000001416162225431456000               NORM PUSHJ[SET-HALF] $
07 0471                                                           ;MAP trap. Set the FirstPartDone flag.
07 0472 04236 010731170000212374161622254314560000               NORM PUSHJ[MAP-RET] $
07 0473                                                           ;We will not return from this PUSHJ, since RE-XCT is set at
07 0473                                                  ILDB3.
07 0474
07 0475 04237 540733170543060552121623254605160000      ILDB3:  TRP-CTL[TRAP-FIX RE-XCT] D[MEM] ROT[R] MASK[R] DEST[AC] CLR-HALF
07 0475                                                  NEOI $
07 0476                                                           ; Get and store byte. On MAP trap, re-xct entire instr.
07 0477
07 0478 04240 710731170000000001415162025431456000      ILDBAX: D[AR] DEST[MA] NORM JUMP[ILDB4] $
07 0479                                                           ;Get MA loaded with addr of data.
07 0480
07 0481                                                  ; Ptr in AC
07m0482                                                   .pair
07m0482                                                  [:. \ 2 + .
07 0483 04242 010731170000105003014500254314570000       ]ILDBA1:     D[AR] ROT[14] MASK[6] DEST[MASKR] JUMP[ILDBAX] $        ;
07 0483                                                  HALF was set
07 0484 04243 010731370000000003014502254314570000              D[AR] ROT[14] MASK[6] DEST[AR MASKR] PUSHJ[IBPXA] $
07 0485                                                                  ; Get S field and call common code
07 0486 04244 010731000000000004413622254214560000      ILDB4:  D[AR] ROT[18.] MASK[5] C550 COND[-OBUS=0] PUSHJ[BII] $
07 0487                                                           ; Test for indexing and indirection
07 0488                                                           ;fall thru
07 0489
07 0490                                                  ;jump here from special xct stuff
07 0491                                                  ;half set and bp indexing and indirection chased, map-user-sr setup
07 0492                                                  ;ma/ adr of data
07 0493                                                  ;ar/ bp
07 0494                                                  ;maskr/ S field
07 0495
07 0496 04245 010730112000104674111620255614560000      ILDB5:  D[CONST 36.] DEST[Q] C550 COND[-MA<20] JUMP[ILDB2] $
07 0497                                                           ;jump if data not in an AC
07 0498 04246 011611170000060541414443654314170000      ILDBA2: D[AR] ROT[6] MASK[6] ALU[Q-D] DEST[ROTR] $
07 0499                                                           ; Data in AC, 36.-P to ROTR
07 0500 04247 010331170000104774161500054314560000              ALU[MEMAC] DEST[HOLD] JUMP[ILDB3] $ ; Get data to useful place
07 0501
07 0502
07m0503                                                  .opcode[135]  [xlist
07 0503                                                   list  ]                                                    ;LDB
07 0504 06272 730730370000000001416142025471356000              DF/WT D[MEM] DEST[Q AR MA IR-23] JUMP[LDB1] $
07 0505 06273 710330370000000001416142005431456000              ALU[MEMAC] DEST[Q AR MA IR-23] JUMP[LDB1] $
07 0506
07m0507                                                  .reloc
07m0507                                                  [.USE[NORMAL]
07m0507                                                  [ xlist
07 0508 04250 010731170000105103014500254314570000       ]LDB1:       D[AR] ROT[14]
07 0508                                                  MASK[6] DEST[MASKR] NORM JUMP[ILDB4] $
07 0509                                                           ;Just like ILDB after the increment...
07 0510
07 0511
07m0512                                                  .opcode[136]  [xlist
07 0512                                                   list  ]                                                    ;IDPB
07 0513 06274 630730237000000001416142125471356000              DF/WT R-M-W D[MEM] DEST[Q AR IR-23] COND[-HALF] LBJUMP[IDPB1] $
07 0514 06275 610330236000000001416142105431456000              ALU[MEMAC] DEST[Q AR IR-23] COND[-HALF] LBJUMP[IDPBA1] $
07 0515
07m0516                                                  .reloc
07m0516                                                  [.USE[NORMAL]
07m0516                                                  [ xlist
07 0517                                                   list ]                                  ]; Ptr in memory
07m0518                                                   .pair
07m0518                                                  [:. \ 2 + .
07 0519 04252 010731170000105303014500254314570000       ]IDPB1:      D[AR] ROT[14] MASK[6] DEST[MASKR] JUMP[. + 2] $ ; HALF
07 0519                                                  was set
07 0520 04253 010731370000000003014502254314570000              D[AR] ROT[14] MASK[6] DEST[AR MASKR] PUSHJ[IBPX]$
07 0521                                                                  ; Get S field and call common code
07 0522 04254 710731370000000001416162025471556000              D[MEM] DEST[MA AR] NORM JUMP[IDPB4] $   ; Get back ptr
07 0523
07 0524                                                  ; Data in memory
07m0525                                                   .odd
07m0525                                                  [:. + 1 - (. \ 2)
07 0526 04255 010541370000000001412162224731256000       ]IDPB2:      DFRQ D[MASKR] ALU[D&AC] DEST[AR] PUSHJ[IDPB3] $
07 0527                                                           ;Get byte
07 0528 04256 010731170000223534161620254314560000              NORM JUMP[ECC-PROCEED] $
07 0529 04257 010731170000000001416162225431456000              NORM PUSHJ[SET-HALF] $
07 0530 04260 010731170000212374161622254314560000              NORM PUSHJ[MAP-RET] $
07 0531
07 0532 04261 010730171500060554161623654715160000      IDPB3:  RE-XCT R-M-W D[TRAP-FIX MEM] DEST[Q] $          ;Get detination
07 0532                                                  word
07 0533 04262 010650170000060552121623641731416000              D[MASKR] ROT[R] ALU[-D&Q] DEST[Q] $             ;Clear dest byte
07 0533
07 0534 04263 010631170043226652161560254304560000              IFRQ D[AR] ROT[R] ALU[DORQ] DEST[MEMSTO] CLR-HALF JUMP[SEOI] $
07 0535                                                           ;Do it and store it
07 0536
07 0537 04264 710731170000000001416162025431456000      IDPBAX: D[AR] DEST[MA] NORM JUMP[IDPB4] $
07 0538                                                           ;Get MA loaded with addr of data.
```

```
07 0543 04267 010731370000000000301450225431457000             D[AR] ROT[14] MASK[6] DEST[AR MASKR] PUSHJ[IBPXA] $
07 0544                                                                 ; Get S field and call common code
07 0545 04270 010731000000000000441362225421456000       IDPB4:  D[AR] ROT[18.] MASK[5] C550 COND[-DBUS=0] PUSHJ[BII] $
07 0546                                                                 ; Test for indexing and indirection
07 0547                                                                 ;fall thru
07 0548
07 0549                                                   ;jump here from special xct stuff
07 0550                                                   ;half set and bp indexing and indirection chased, map-user-sr setup
07 0551                                                   ;ma/ adr of data
07 0552                                                   ;ar/ bp
07 0553                                                   ;maskr/ S field
07 0554
07 0555 04271 010730112000105321414440254214457000       IDPBS:  D[AR] ROT[6] MASK[6] DEST[Q ROTR] C550 COND[-MA<Z0] JUMP[IDPBZ] $
07 0555
07 0556                                                                 ; Jump if data is in memory.
07 0557                                                   ; Data in AC
07 0558 04272 010541370000060554121623647314116000       IDPBAZ: D[MASKR] ALU[D&AC] DEST[AR] $   ;Get byte
07 0559 04273 010330170000060554161623454314116000             ALU[MEMAC] DEST[Q] $                              ;Get detination
                                                          word
07 0560 04274 010650170000060552121623647314116000             D[MASKR] ROT[R] ALU[-D&Q] DEST[Q] $               ;Clear dest byte
07 0560
07 0561 04275 540633170041306055216162305420436000             D[AR] ROT[R] ALU[DORQ] DEST[MEMAC] CLR-HALF NEOI $
07 0562                                                                  ;Do it and store it
07 0563
07m0564                                                          .opcode[137]   [xlist
07 0564                                                           list  ]                                          ;DPB
07 0565 06276 730730370000000014161420254713560000             DF/WT D[MEM] DEST[Q AR MA IR-23] JUMP[DPB1] $
07 0566 06277 710330370000000014161420054314560000             ALU[MEMAC] DEST[Q AR MA IR-23] JUMP[DPB1] $
07m0567                                                          .reloc
07m0567                                                   [.USE[NORMAL]
07m0567                                                   [ xlist
07 0568 04276 010731170000105603014500254314457000         ]DPB1:      D[AR] ROT[14]
07 0568                                                   MASK[6] DEST[MASKR] JUMP[IDPB4] $
07 0569
07 0570
```

```
@@ 0571
@@ 0572                                                 ;;; Routines for byte instructions
@@ 0573
@@ 0574                                                 ;ILDB and IDPB call here to increment ptr with S already in AR
@@ 0575
@@ 0576 04277 011611030000060547561564254214160000     IBPX:   D[AR] ROT[30.] ALU[Q-D] DEST[MEMSTO] CS50 COND[CRY0] POPJ $
@@ 0577                                                                 ;Subtract from P (in place) and store, checking for
@@ 0577                                                  overflow
@@ 0578 04300 010540170000060554075623647314160000            D[MASK 30.] ALU[D&Q] DEST[Q] $
@@ 0579                                                                 ; ptr will overflow, fix it up
@@ 0580 04301 011600170000060547511503655714160000            D[CONST 44] ROT[30.] ALU[D+Q+1] DEST[Q HOLD] $
@@ 0581                                                                 ;New P of 44, add 1 to adr
@@ 0582 04302 010731000000000014045620254615560000            D[MEM] MASK[18.] COND[-ZERO] JUMP[IBPX1] CS50 $
@@ 0583                                                                 ;JUMP IF ADR NOT 0 (DID NOT OVERFLOW)
@@ 0584 04303 011610170000060544403623655714160000            D[CONST 1] ROT[18.] ALU[Q-D] DEST[Q] NORM $
@@ 0585                                                                 ;BACKUP LEFT HALF CAUSING WRAP AROUND TO 0 FROM 777777
@@ 0586                                                                 ;CARRY CAN'T PROPAGATE PAST BIT 0 CAUSE OF 44 IN B0-5
@@ 0587 04304 011611370000060547561564254314160000     IBPX1:  D[AR] ROT[30.] ALU[Q-D] DEST[MEMSTO AR] POPJ $
@@ 0588                                                                 ;STORE IT
@@ 0589                                                                 ;MEM,AR/ INCED BP
@@ 0590
@@ 0591                                                 ; Same as above when ptr is in an AC
@@ 0592                                                 ; Clobbers HOLD now!!
@@ 0593
@@ 0594                                                 IBPXA:
@@ 0595                                                 .repeat waitsfix [
@@ 0596                                                 ;Please tell me why this could be flushed.  TVR/Feb83
@@ 0597                                                         D[MA] DEST[HOLD] SHORT $
@@ 0598                                                                 ;Save MA for possible use below.
@@m0599                                                  ];[
@@m0599                                                  ;Please tell me why this could be flushed.  TVR/Feb83
@@m0599 04305 010731170000060554161503656774160000             D[MA] DEST[HOLD] SHORT $
@@m0599                                                                 ;Save MA for possible use below.
@@ 0599                                                  ].repeat waitsfix
@@ 0600 04306 711613230000060547561624054214360000             D[AR] ROT[30.] ALU[Q-D] DEST[MEMAC AR MA] CS50 COND[CRY0] POPJ $
@@ 0600
@@ 0601                                                                 ;Subtract from P (in place) and store, checking for
@@ 0601                                                  overflow
@@ 0602 04307 010731370000060543015623655714160000            D[MA] ROT[12.] MASK[6] DEST[AR] $
@@ 0603                                                                 ; Attempt to be clever failed...
@@ 0604 04310 710731170000060554161623654715160000            D[MEM] DEST[MA] $
@@ 0605                                                                 ; so restore what was smashed
@@ 0606 04311 010640170000060554075623647314160000            D[MASK 30.] ALU[D&Q] DEST[Q] $
@@ 0607                                                                 ; ptr will overflow, fix it up
@@ 0608 04312 011600170000060547511503655714160000            D[CONST 44] ROT[30.] ALU[D+Q+1] DEST[Q HOLD] $
@@ 0609                                                                 ;New P of 44, add 1 to adr
@@ 0610 04313 010731000000000014045620254615560000            D[MEM] MASK[18.] COND[-ZERO] JUMP[IBPXA1] CS50 $
@@ 0611                                                                 ;JUMP IF ADR NOT 0 (DID NOT OVERFLOW)
@@ 0612 04314 011610170000060544403623655714160000            D[CONST 1] ROT[18.] ALU[Q-D] DEST[Q] NORM $
@@ 0613                                                                 ;BACKUP LEFT HALF CAUSING WRAP AROUND TO 0 FROM 777777
@@ 0614                                                                 ;CARRY CAN'T PROPAGATE PAST BIT 0 CAUSE OF 44 IN B0-5
@@ 0615 04315 711613370000060547561504054314360000     IBPXA1: D[AR] ROT[30.] ALU[Q-D] DEST[MEMAC AR MA HOLD] POPJ $
@@ 0616                                                                 ;STORE IT
@@ 0617                                                                 ;MEM,AC,AR,MA/ INCED BP
@@ 0618
@@ 0619                                                 ; Handle byte-ptr indexing and indirection
@@ 0620 04316 010730002000000004411620254214560000     BII:    D[AR] ROT[18.] MASK[4] DEST[Q] CS50 COND[OBUS=0] JUMP[BIIH1] $
@@ 0621                                                                 ;Test index=0
@@ 0622 04317 710501170000000001416162011671456000            D[MA] ALU[D+XR] DEST[MA] NORM JUMP[BIIHA] $
@@ 0623                                                                 ;Do indexing (IDX field of ptr was copied to IR earlier)
@@ 0623
@@m0624                                                  .even
@@m0624                                                  [:. \ 2 + .
@@ 0625 04320 010731006000060543361624254214160000       ]BIIHA:       D[AR] ROT[13.]  CS50 COND[SIGNOFF] POPJ $ ; Leave if no e
@@ 0625                                                  bit
@@m0626                                                  .odd
@@m0626                                                  [:. + 1 - (. \ 2)
@@ 0627                                                    ]BIIH1:     ;EA should still be same as MA
@@ 0628 04321 010331170000000001416150205431256000           DFRQ ALU[MEMAC] DEST[HOLD] PUSHJ[BIIHZ] $
@@ 0629                                                                   ;Fake up AC refs, provide trap fixup address.
@@ 0630 04322 010731170000223534161620254314560000           JUMP[ECC-PROCEED] $       ;ECC trap comes here.
@@ 0631 04323 010731170000000001416162225431456000            PUSHJ[SET-HALF] $        ;Map trap comes here.
@@ 0632 04324 010731170000212374161622254314560000           PUSHJ[MAP-RET] $
@@ 0633
@@ 0634 04325 710730170503060554057423654715160000     BIIHZ:  TRP-CTL[RE-XCT TRAP-FIX] D[MEM] MASK[23.] DEST[Q MA IR-23] MU-POP
@@ 0634                                                  $
@@ 0635                                                                 ;Get indirect word, repair stack.
@@ 0636 04326 010731002000060544413624254615160000           D[MEM] ROT[18.] MASK[5] CS50 COND[OBUS=0] POPJ $
@@ 0637                                                                 ;If no idx or ind, exit
@@ 0638 04327 010731370000060543031623654314160000           D[AR] ROT[12.] MASK[12.] DEST[AR] NORM $ ;Get original P and S
@@ 0639 04330 010630306000010634161616202542145600 0          D[AR] ROT[24.] ALU[DORQ] DEST[Q AR] CS50 COND[-INTRPT] JUMP[BII]
@@ 0639                                                  $
@@ 0640                                                                 ; Loop if no interrupt is waiting.
@@ 0641 04331 010731170000000001416162225431456000           PUSHJ[SET-HALF] NORM $
@@ 0642                                                                 ;Intrpt. waiting.   Set HALF (BIS) flag in PC.
@@ 0643 04332 610721170000227114161614602563145600000        D[PC] ALU[D-1] DEST[PC] NORM JUMP[MAIN] $
@@ 0644                                                                 ;Back up PC (so instr. will be re-executed) and take
@@ 0644                                                  intrpt.
@@ 0645
```

```
09 0646
09m0647                                                  .opcode[200]  [xlist
09 0647                                                   list  ]                                        ;MOVE
09 0648 06400 560733170000060554161623254603160000            DF/IF D[MEM] DEST[AC] NEOI $
09 0649 06401 540433170000060554161623014204160000            ACSEL[MA,AC] ALU[A] DEST[B] NEOI $
09 0650                                                  ;MOVEI
09 0650                                                        D[IR] MASK[18.] DEST[AC] NEOI $
09 0651 06402 540733170000060554045623257604160000            D[IR] MASK[18.] DEST[AC] NEOI $
09 0652 06403 540733170000060554045623257604160000
09 0653                                                        IFRQ ALU[AC] DEST[MEMSTO] NORM JUMP[SEQI] $
09 0654 06404 010331170000226654161560254304560000      ;MOVEM
09 0654                                                        ACSEL[AC,MA] ALU[A] DEST[B] NEOI $
09 0655 06405 540433170000060554161623054204160000
09 0656                                                        DF/IF R-M-W D[MEM] DEST[MEMSTO AR] JUMP[CASEQI]  $
09 0657 06406 030731371000227034161560254703560000      ;MOVES
09 0657                                                        ACSEL[MA,AC] ALU[A] COND-AC-STO NEOI $
09 0658 06407 540431170000060554160163014204160000
09 0659
09 0660
09m0661                                                  .opcode[204]  [xlist
09 0661                                                   list  ]                                        ;MOVS
09 0662 06410 560733170000060544561623254603160000            DF/IF D[MEM] ROT[18.] DEST[AC] NEOI $
09 0663 06411 010331370000000014161620054314560000            ALU[MEMAC] DEST[AR] NORM JUMP[MOVS1] $
09 0664
09 0665 06412 540733170000060544541623257604160000            D[IR] MASK[LEFT] ROT[18.] DEST[AC] NEOI $
09 0665                                                  ;MOVSI
09 0666 06413 540733170000060544541623257604160000            D[IR] MASK[LEFT] ROT[18.] DEST[AC] NEOI $
09 0667
09 0668 06414 010331370000000014161620254314560000            ALU[AC] DEST[AR] NORM JUMP[MOVSM1] $
09 0668                                                  ;MOVSM
09 0669 06415 010331370000000014161620254314560000            ALU[AC] DEST[AR] NORM JUMP[MOVSM2] $
09 0670
09 0671 06416 030731370000227024561560254703560000            DF/IF D[MEM] ROT[18.] DEST[MEMSTO AR] JUMP[CASEQI]       $
09 0671                                                  ;MOVSS
09 0672 06417 010331370000000014161620054314560000            ALU[MEMAC] DEST[AR] NORM JUMP[MOVSS1] $
09 0673
09m0674                                                   .reloc
09m0674                                                  [.USE[NORMAL]
09m0674                                                  [ xlist
09 0675 04333 540733170000060544561623254204160000       list ]                                  ]MOVS1:           D[AR]
09 0675                                                 ROT[18.] DEST[AC] NEOI $
09 0676 04334 010731170000226644561560254304560000     MOVSM1: IFRQ D[AR] ROT[18.] DEST[MEMSTO] NORM JUMP[SEQI] $
09 0677 04335 540733170000060544561623054204360000     MOVSM2:      D[AR] ROT[18.] DEST[MEMAC] NEOI $
09 0678 04336 010731370000227064560160254304560000     MOVSS1: IFRQ D[AR] ROT[18.] DEST[AR] COND-AC-STO NORM JUMP[MACSTO] $
09 0679
09 0680
09m0681                                                  .opcode[210]  [xlist
09 0681                                                   list  ]                                        ;MOVN
09 0682 06420 561713170044060554161623254603160000            DF/IF D[MEM] ALU[0-D] DEST[AC]    SET-PC-FLAGS NEOI $
09 0683 06421 541423170044060554161623014204160000            ACSEL[MA,AC] ALU[0-A] DEST[B]     SET-PC-FLAGS NEOI $
09 0684
09 0685 06422 541713170044060554045623257604160000            D[IR] MASK[18.] ALU[0-D] DEST[AC] SET-PC-FLAGS NEOI $   ;MOVNI
09 0686 06423 541713170044060554045623257604160000            D[IR] MASK[18.] ALU[0-D] DEST[AC] SET-PC-FLAGS NEOI $
09 0687
09 0688                                                        IFRQ ALU[0-AC] DEST[MEMSTO]       SET-PC-FLAGS
09 0689 06424 011321170044226654161560254304560000                                            NORM JUMP[SEQI] $
09 0689                                                 ;MOVNM
09 0690 06425 541423170044060554161623054204160000            ACSEL[AC,MA] ALU[0-A] DEST[B]     SET-PC-FLAGS NEOI $
09 0691
09 0692                                                        DF/IF R-M-W  D[MEM] ALU[0-D] DEST[MEMSTO AR] SET-PC-FLAGS
09 0693 06426 031711371044227034161560254703560000                                                    JUMP[CASEQI] $
09 0693                                                 ;MOVNS
09 0694                                                        IFRQ ACSEL[MA,AC] ALU[0-A] DEST[AR]    SET-PC-FLAGS
09 0695 06427 011421370044227074160160014304560000                                    COND-AC-STO NORM JUMP[MACSTO] $
09 0696
09 0697
09m0698                                                  .opcode[214]  [xlist
09 0698                                                   list  ]                                        ;MOVM
09 0699 06430 030731204000000014161621254713560000            DF/WT D[MEM] DEST[AR]   NORM  OBUS<0 LBJUMP[MOVM1] $
09 0700 06431 010331204000000014161621054314560000            ALU[MEMAC]   DEST[AR]   NORM  OBUS<0 LBJUMP[MOVM1] $
09 0701
09 0701                                                 ;MOVMI
09 0702 06432 540733170000060554045623257604160000            D[IR] MASK[18.] DEST[AC]  NEOI $
09 0703 06433 540733170000060554045623257604160000            D[IR] MASK[18.] DEST[AC]  NEOI $
09 0704
09 0705 06434 010331004000000014161621254314560000            ALU[AC]                  NORM  OBUS<0 LBJUMP[MOVMM1] $
09 0705                                                 ;MOVMM
09 0706 06435 010331004000000014161621254314560000            ALU[AC]                  NORM  OBUS<0 LBJUMP[MOVMM2] $
09 0707
09 0708 06436 030731205000000014161621254713560000            DF/WT R-M-W D[MEM] DEST[AR]     NORM  OBUS<0 LBJUMP[MOVMS1] $
09 0708                                                 ;MOVMS
09 0709 06437 010331204000000014161621054314560000            ALU[MEMAC]   DEST[AR]   NORM  OBUS<0 LBJUMP[MOVMS2] $
09 0710                                                                                                  ...#
09m0711                                                   .reloc
09m0711                                                  [.USE[NORMAL]
09m0711                                                  [ xlist
09m0712                                                    list ]                                  ]  .pair
09m0712                                                  [:. \ Z + .
09 0713 04340 540733170000060554161623254204160000       ]MOVM1:      D[AR]          DEST[AC] NEOI $
09 0714 04341 541713170000060554161623254204160000            D[AR] ALU[0-D] DEST[AC] NEOI $
09 0715 04342 010331170000226654161560254304560000     MOVMM1: IFRQ ALU[AC]            DEST[MEMSTO] JUMP[SEQI] $
09 0716 04343 011321170044226654161560254304560000            IFRQ ALU[0-AC]           DEST[MEMSTO] SET-PC-FLAGS JUMP[SEQI] $
09 0717 04344 010331370000227074161620254304560000     MOVMM2:      ALU[AC]            DEST[AR]# JUMP[MACSTO] IFRQ $
09 0718 04345 011321370044227074161620254304560000                 ALU[0-AC]          DEST[AR] SET-PC-FLAGS JUMP[MACSTO] IFRQ $
09 0718
09 0719 04346 010731170000227034161560254304560000     MOVMS1: IFRQ D[AR]              DEST[MEMSTO] JUMP[CASEQI] $
09 0720 04347 011711370044227034161560254304560000            IFRQ D[AR] ALU[0-D]      DEST[MEMSTO AR] SET-PC-FLAGS JUMP[CASEQI]
09 0720                                                  $
09 0721 04350 540731170000060554160163254204160000     MOVMS2: IFRQ D[AR]              COND-AC-STO  NEOI $
09 0722 04351 011711370044227074160160254304560000            IFRQ D[AR] ALU[0-D] DEST[AR] COND-AC-STO SET-PC-FLAGS
09 0722                                                  JUMP[MACSTO] $
09 0723
```

```
10 0724
10 0725                                                 ;;; Integer multiply
10 0726
10m0727                                                   .opcode[ZZ0]   [xlist
10 0727                                                    list  ]                                                    ;IMUL
10 0728 06440 030730170000000001416162025471356000             DF/WT D[MEM] DEST[Q] JUMP[IMUL1] NORM $
10 0729 06441 010330170000000001416162005431456000             ALU[MEMAC] DEST[Q] JUMP[IMUL1] $
10 0730
10m0731                                                   .opcode[ZZ1]   [xlist
10 0731                                                    list  ]                                                    ;IMULI
10 0732 06442 010730170000000001404562025771456000             D[IR] MASK[18.] DEST[Q] JUMP[IMUL1] NORM $
10 0733 06443 010730170000000001404562025771456000             D[IR] MASK[18.] DEST[Q] JUMP[IMUL1] NORM $
10 0734
10m0735                                                   .opcode[ZZZ]   [xlist
10 0735                                                    list  ]                                                    ;IMULM
10 0736 06444 030730170000000001416162025471356000             DF/WT D[MEM] DEST[Q] JUMP[IMUL2] NORM $
10 0737 06445 010330170000000001416162005431456000             ALU[MEMAC] DEST[Q] JUMP[IMUL2] $
10 0738
10m0739                                                   .opcode[ZZ3]   [xlist
10 0739                                                    list  ]                                                    ;IMULB
10 0740 06446 030730171000000001416162025471356000             DF/WT R-M-W D[MEM] DEST[Q] JUMP[IMUL3] NORM $
10 0741 06447 010330170000000001416162005431456000             ALU[MEMAC] DEST[Q] JUMP[IMUL3] $
10 0742
10m0743                                                   .opcode[ZZ4]   [xlist
10 0743                                                    list  ]                                                    ;MUL
10 0744 06450 030730170000000001416162025471356000             DF/WT D[MEM] DEST[Q] JUMP[MUL1] NORM $
10 0745 06451 010330170000000001416162005431456000             ALU[MEMAC] DEST[Q] JUMP[MUL1] $
10 0746
10m0747                                                   .opcode[ZZ5]   [xlist
10 0747                                                    list  ]                                                    ;MULI
10 0748 06452 010730170000000001404562025771456000             D[IR] MASK[18.] DEST[Q] JUMP[MUL1] NORM $
10 0749 06453 010730170000000001404562025771456000             D[IR] MASK[18.] DEST[Q] JUMP[MUL1] NORM $
10 0750
10m0751                                                   .opcode[ZZ6]   [xlist
10 0751                                                    list  ]                                                    ;MULM
10 0752 06454 030730171000000001416162025471356000             DF/WT R-M-W D[MEM] DEST[Q] JUMP[MUL2] NORM $
10 0753 06455 010330170000000001416162005431456000             ALU[MEMAC] DEST[Q] JUMP[MUL2] $
10 0754
10m0755                                                   .opcode[ZZ7]   [xlist
10 0755                                                    list  ]                                                    ;MULB
10 0756 06456 030730171000000001416162025471356000             DF/WT R-M-W D[MEM] DEST[Q] JUMP[MUL3] NORM $
10 0757 06457 010330170000000001416162005431456000             ALU[MEMAC] DEST[Q] JUMP[MUL3] $
10 0758
10m0759                                                   .reloc
10m0759                                                  [.USE[NORMAL]
10m0759                                                  [ xlist
10 0760 04352 010242370000000001416162225431456000       list ]                                ]IMUL1:      ALU[0] DEST[AR
10 0760                                                  O-AC] PUSHJ[DOIMUL] NORM $       ;0 TO AC,AC TO AR
10 0761 04353 540233170000060554161623254204160000             ALU[Q] DEST[AC] NEOI $
10 0762
10 0763 04354 010242370000000001416162225431456000       IMUL2:  ALU[0] DEST[AR O-AC] PUSHJ[DOIMUL] NORM $       ;0 TO AC
10 0764 04355 010733170000060554161623654314160000             D[AR] DEST[AC] NORM $
10 0765 04356 010231310000226654161561254304560000             ALU[Q] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<Z0] LBJUMP[SEOI] ]$
10 0766
10 0767 04357 010242370000000001416162225431456000       IMUL3:  ALU[0] DEST[AR O-AC] PUSHJ[DOIMUL] NORM $
10 0768 04360 010233310000226654161561254304560000             ALU[Q] DEST[AC] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<Z0]
10 0768                                                  LBJUMP[SEOI] ]$
10 0769
10 0770 04361 010242370000000001416162225431456000       MUL1:   ALU[0] DEST[AR O-AC] PUSHJ[DOMMUL] NORM $
10 0771 04362 540233170000060554161623154204160000             ACSEL[AC+1] ALU[Q] DEST[AC] NEOI $
10 0772
10 0773 04363 010242370000000001416162225431456000       MUL2:   ALU[0] DEST[AR O-AC] PUSHJ[DOMMUL] NORM $
10 0774 04364 010732310000226654161561254304560000             D[AR] DEST[O-AC] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<Z0]
10 0774                                                  LBJUMP[SEOI] ]$
10 0775
10 0776 04365 010242370000000001416162225431456000       MUL3:   ALU[0] DEST[AR O-AC] PUSHJ[DOMMUL] NORM $
10 0777 04366 010233170000060554161623554314160000             ACSEL[AC+1] ALU[Q] DEST[AC] NORM $
10 0778 04367 010331310000226654161561254304560000             ALU[AC] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<Z0] LBJUMP[SEOI] ]$
10 0778
10 0779
```

```
11 0780
11 0781                                                 ;;; integer multiply work routines
11 0782
11 0783 04370 010731170000000001416162225431456000     DOIMUL: PUSHJ[DOMUL] NORM $
11 0784 04371 010331002000060554161624254214116000             ALU[AC] C550 COND[OBUS=0] POPJ $; NO OV IF 0
11 0785 04372 010371002000060554161624254214116000             ALU[NOTAC] C550 COND[OBUS=0] POPJ $; NO OV IF -1
11 0786                                                 .repeat 1 - waitsfix [
11 0787                                                 MSETOV: D[PC-FLAGS] DEST[AC] $; GET FLAGS
11 0788                                                 MSTOV1: D[CONST 1] ROT[43] ALU[DORAC] DEST[PC-FLAGS] NORM POPJ $; SET OV
11 0789
11 0790                                                 ];.repeat 1 - waitsfix
11 0791                                                 .repeat waitsfix [      ;You'll need the JUMP[SETOVX], but is the C550
                                                         correct?
11 0792                                                 MSETOV: D[PC-FLAGS] DEST[AC] C550 $
11 0793                                                         ;Get flags
11 0794                                                 MSTOV1: D[CONST 1] ROT[43] ALU[DORAC] DEST[PC-FLAGS] NORM JUMP[SETOVX] $
11m0795
11m0795                                                         ;Go cause trap if enabled.
11m0795 04373 010733170000060554161623646214116000     ];[     ;You'll need the JUMP[SETOVX], but is the C550 correct?
11m0795                                                 MSETOV: D[PC-FLAGS] DEST[AC] C550 $
11m0795 04374 010531170000000001060242025571457000             ;Get flags
11m0795                                                 MSTOV1: D[CONST 1] ROT[43] ALU[DORAC] DEST[PC-FLAGS] NORM JUMP[SETOVX] $
11m0795
11m0795                                                         ;Go cause trap if enabled.
11 0795                                                 ].repeat waitsfix
11 0796
11 0797 04375 010661000000000001416162025421456000     DOMMUL: D[AR] ALU[D*Q] C550 COND[-OBUS=0] JUMP[DOMUL] $; NO OV IF DIFF
11 0797                                                  OPERS
11 0798 04376 010661000000000010603620255614456000             D[CONST 1] ROT[43] ALU[D*Q] C550 COND[-OBUS=0] JUMP[DOMUL] $
11 0799                                                         ;Jump if not -Z**35
11 0800 04377 010731170000107674161622254314456000             PUSHJ[MSETOV] NORM $;SET OV
11 0801 04400 010243170000060554161623654374116000             ALU[0] DEST[AC] SHORT $
11 0802
11 0803                                                  .define MULM [] [  ;One bit of a multiply...
11 0804                                                         ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. + 2] $
11 0804
11 0805                                                           ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11 0806                                                         D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. +
11 0806                                                  1] $
11 0807                                                           ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11 0808                                                    ]
11 0809 04401 010304024000000001416762125431456000     DOMUL:  ALU[AC+0] DEST[D4] ENDCONN[M/D] norm COND[Q0-35] LBJUMP[DOMUL0] $
11 0809
11 0810                                                          ;Enter the multiply "loop".
11 0811 04402 010304024000000035416762125431456000     DOMULF: ALU[AC+0] DEST[D4] ENDCONN[M/D] norm COND[Q0-35] LBJUMP[DOMUL0 +
11 0811                                                  14.] $
11 0812                                                          ;Multiply "loop" for floating point... 7 fewer iterations.
11m0813                                                  .even
11m0813                                                  [:. \ 2 + .
11m0814                                                    ]DOMUL0:      .REPEAT 35. [ MULM ]
11m0814                                                 [ MULM  [  ;One bit of a multiply...
11m0814 04404 010304024000110154167621254374456000             ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. + 2] $
11m0814
11m0814                                                           ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04405 010504024000110154167621254374456000             D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. +
11m0814                                                  1] $
11m0814                                                           ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                    ]][ MULM  [  ;One bit of a multiply...
11m0814 04406 010304024000110214167621254374456000             ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. + 2] $
11m0814
11m0814                                                           ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04407 010504024000110214167621254374456000             D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. +
11m0814                                                  1] $
11m0814                                                           ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                    ]][ MULM  [  ;One bit of a multiply...
11m0814 04410 010304024000110254167621254374456000             ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. + 2] $
11m0814
11m0814                                                           ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04411 010504024000110254167621254374456000             D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. +
11m0814                                                  1] $
11m0814                                                           ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                    ]][ MULM  [  ;One bit of a multiply...
11m0814 04412 010304024000110314167621254374456000             ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. + 2] $
11m0814
11m0814                                                           ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04413 010504024000110314167621254374456000             D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. +
11m0814                                                  1] $
11m0814                                                           ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                    ]][ MULM  [  ;One bit of a multiply...
11m0814 04414 010304024000110354167621254374456000             ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. + 2] $
11m0814
11m0814                                                           ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04415 010504024000110354167621254374456000             D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. +
11m0814                                                  1] $
11m0814                                                           ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                    ]][ MULM  [  ;One bit of a multiply...
11m0814 04416 010304024000110414167621254374456000             ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. + 2] $
11m0814
11m0814                                                           ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04417 010504024000110414167621254374456000             D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. +
11m0814                                                  1] $
11m0814                                                           ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                    ]][ MULM  [  ;One bit of a multiply...
11m0814 04420 010304024000110454167621254374456000             ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. + 2] $
11m0814
11m0814                                                           ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04421 010504024000110454167621254374456000             D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. +
11m0814                                                  1] $
11m0814                                                           ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                    ]][ MULM  [  ;One bit of a multiply...
11m0814 04422 010304024000110514167621254374456000             ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. + 2] $
11m0814
11m0814                                                           ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04423 010504024000110514167621254374456000             D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. +
11m0814                                                  1] $
11m0814                                                           ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                    ]][ MULM  [  ;One bit of a multiply...
11m0814 04424 010304024000110554167621254374456000             ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. + 2] $
11m0814
11m0814                                                           ;Here if multiplier bit is a 0.  Test next mulitplier bit.
```

```
11m0814
11m0814 04427 0105040240001106141676212543745600 0          ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814                                                      D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. +
11m0814                                                    1] $
11m0814
11m0814                                                      ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                 ]][ MULM  [  ;One bit of a multiply...
11m0814 04430 0103040240001106541676212543745600            ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. + 2] $
11m0814
11m0814                                                      ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04431 0105040240001106541676212543745600 0          D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. +
11m0814                                                    1] $
11m0814                                                      ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                 ]][ MULM  [  ;One bit of a multiply...
11m0814 04432 0103040240001107141676212543745600            ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. + 2] $
11m0814
11m0814                                                      ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04433 0105040240001107141676212543745600 0          D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. +
11m0814                                                    1] $
11m0814                                                      ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                 ]][ MULM  [  ;One bit of a multiply...
11m0814 04434 0103040240001107541676212543745600            ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. + 2] $
11m0814
11m0814                                                      ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04435 0105040240001107541676212543745600 0          D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. +
11m0814                                                    1] $
11m0814                                                      ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                 ]][ MULM  [  ;One bit of a multiply...
11m0814 04436 0103040240001110141676212543745600            ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. + 2] $
11m0814
11m0814                                                      ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04437 0105040240001110141676212543745600 0          D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. +
11m0814                                                    1] $
11m0814                                                      ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                 ]][ MULM  [  ;One bit of a multiply...
11m0814 04440 0103040240001110541676212543745600            ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. + 2] $
11m0814
11m0814                                                      ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04441 0105040240001110541676212543745600 0          D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. +
11m0814                                                    1] $
11m0814                                                      ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                 ]][ MULM  [  ;One bit of a multiply...
11m0814 04442 0103040240001111141676212543745600            ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. + 2] $
11m0814
11m0814                                                      ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04443 0105040240001111141676212543745600 0          D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. +
11m0814                                                    1] $
11m0814                                                      ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                 ]][ MULM  [  ;One bit of a multiply...
11m0814 04444 0103040240001111541676212543745600            ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. + 2] $
11m0814
11m0814                                                      ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04445 0105040240001111541676212543745600 0          D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. +
11m0814                                                    1] $
11m0814                                                      ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                 ]][ MULM  [  ;One bit of a multiply...
11m0814 04446 0103040240001112141676212543745600            ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. + 2] $
11m0814
11m0814                                                      ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04447 0105040240001112141676212543745600 0          D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. +
11m0814                                                    1] $
11m0814                                                      ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                 ]][ MULM  [  ;One bit of a multiply...
11m0814 04450 0103040240001112541676212543745600            ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. + 2] $
11m0814
11m0814                                                      ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04451 0105040240001112541676212543745600 0          D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. +
11m0814                                                    1] $
11m0814                                                      ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                 ]][ MULM  [  ;One bit of a multiply...
11m0814 04452 0103040240001113141676212543745600            ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. + 2] $
11m0814
11m0814                                                      ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04453 0105040240001113141676212543745600 0          D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. +
11m0814                                                    1] $
11m0814                                                      ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                 ]][ MULM  [  ;One bit of a multiply...
11m0814 04454 0103040240001113541676212543745600            ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. + 2] $
11m0814
11m0814                                                      ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04455 0105040240001113541676212543745600 0          D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. +
11m0814                                                    1] $
11m0814                                                      ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                 ]][ MULM  [  ;One bit of a multiply...
11m0814 04456 0103040240001114141676212543745600            ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. + 2] $
11m0814
11m0814                                                      ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04457 0105040240001114141676212543745600 0          D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. +
11m0814                                                    1] $
11m0814                                                      ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                 ]][ MULM  [  ;One bit of a multiply...
11m0814 04460 0103040240001114541676212543745600            ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. + 2] $
11m0814
11m0814                                                      ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04461 0105040240001114541676212543745600 0          D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. +
11m0814                                                    1] $
11m0814                                                      ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                 ]][ MULM  [  ;One bit of a multiply...
11m0814 04462 0103040240001115141676212543745600            ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. + 2] $
11m0814
11m0814                                                      ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04463 0105040240001115141676212543745600 0          D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. +
11m0814                                                    1] $
11m0814                                                      ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                 ]][ MULM  [  ;One bit of a multiply...
11m0814 04464 0103040240001115541676212543745600            ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. + 2] $
11m0814
11m0814                                                      ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04465 0105040240001115541676212543745600 0          D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[00-35] LBJUMP[. +
11m0814                                                    1] $
11m0814                                                      ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                 ]][ MULM  [  ;One bit of a multiply...
```

```
11m0814                                                                 ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                              ]][ MULM  [  ;One bit of a multiply...
11m0814 04470 010304024000111654167621254374560000                     ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. + 2] $

11m0814                                                                 ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04471 010504024000111654167621254374560000                     D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. +
11m0814                                                              1] $
11m0814                                                                 ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                              ]][ MULM  [  ;One bit of a multiply...
11m0814 04472 010304024000111714167621254374560000                     ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. + 2] $

11m0814                                                                 ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04473 010504024000111714167621254374560000                     D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. +
11m0814                                                              1] $
11m0814                                                                 ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                              ]][ MULM  [  ;One bit of a multiply...
11m0814 04474 010304024000111754167621254374560000                     ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. + 2] $

11m0814                                                                 ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04475 010504024000111754167621254374560000                     D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. +
11m0814                                                              1] $
11m0814                                                                 ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                              ]][ MULM  [  ;One bit of a multiply...
11m0814 04476 010304024000112014167621254374560000                     ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. + 2] $

11m0814                                                                 ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04477 010504024000112014167621254374560000                     D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. +
11m0814                                                              1] $
11m0814                                                                 ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                              ]][ MULM  [  ;One bit of a multiply...
11m0814 04500 010304024000112054167621254374560000                     ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. + 2] $

11m0814                                                                 ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04501 010504024000112054167621254374560000                     D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. +
11m0814                                                              1] $
11m0814                                                                 ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                              ]][ MULM  [  ;One bit of a multiply...
11m0814 04502 010304024000112114167621254374560000                     ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. + 2] $

11m0814                                                                 ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04503 010504024000112114167621254374560000                     D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. +
11m0814                                                              1] $
11m0814                                                                 ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                              ]][ MULM  [  ;One bit of a multiply...
11m0814 04504 010304024000112154167621254374560000                     ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. + 2] $

11m0814                                                                 ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04505 010504024000112154167621254374560000                     D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. +
11m0814                                                              1] $
11m0814                                                                 ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                              ]][ MULM  [  ;One bit of a multiply...
11m0814 04506 010304024000112214167621254374560000                     ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. + 2] $

11m0814                                                                 ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04507 010504024000112214167621254374560000                     D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. +
11m0814                                                              1] $
11m0814                                                                 ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11m0814                                                              ]][ MULM  [  ;One bit of a multiply...
11m0814 04510 010304024000112254167621254374560000                     ALU[AC+0] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. + 2] $

11m0814                                                                 ;Here if multiplier bit is a 0.  Test next mulitplier bit.
11m0814 04511 010504024000112254167621254374560000                     D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] CMUL COND[Q0-35] LBJUMP[. +
11m0814                                                              1] $
11m0814                                                                 ;Here if multiplier bit is a 1.  Test next mulitplier bit.
11 0814                                                              ]]
11 0815 04512 010304170000000001416762025431456000                     ALU[AC+0] DEST[D4] ENDCONN[M/D] JUMP[DOMUL1] $ ;Last add and
11 0815                                                              shift
11 0816 04513 010504170000060554167623654314160000                     D[AR] ALU[AC+D] DEST[D4] ENDCONN[M/D] $
11 0817
11 0818 04514 011517170000060554161623654314160000                     D[AR] ALU[AC-D] DEST[D7] $ ;Multiplier was negative, subtract
11 0819 04515 010331004000000001416162125431456000                     ALU[AC] COND[OBUS<0] LBJUMP[DOMUL4] $
11 0820
11 0821                                                              ;Multiplier was positive
11 0822 04516 010437170000060554161623654314160000                  DOMUL1: ALU[SH-AC] DEST[D7] NORM $ ;DEST AC SH LFT
11 0823 04517 010331004000000001416162125431456000                     ALU[AC] COND[OBUS<0] LBJUMP[DOMUL4] $
11 0824
11m0825                                                               .pair
11m0825                                                              [:. \ 2 + .
11 0826 04520 010640170000060554107624247314160000                    ]DOMUL4:     D[MASK 35.] ALU[D&Q] DEST[Q] POPJ NORM $
11 0827                                                                            ;Positive.  Turn off Q0, which was copied into AC35.
11 0828 04521 010630170000060550603624255714160000                     D[CONST 1] ROT[35.] ALU[DORQ] DEST[Q] POPJ NORM $
11 0829                                                                            ;Negative.  Set Q0 so the number is negative (Q0 was
11 0830                                                                            ;copied into AC35).
11 0831
```

```
12 0832
12 0833
12 0834                                                 ;-------------------------------------------------------------------------
12 0834                                                 ------
12 0835                                                 ;
12 0836                                                 ;       Integer Divide
12 0837                                                 ;
12 0838                                                 ;-------------------------------------------------------------------------
12 0839                                                 ------
12m0840                                                         .opcode[230]    [xlist
12 0840                                                   list  ]                                       ;IDIV
12 0841 06460 030731170000000001416162025431256000             DF/WT JUMP[IDIV9] $
12 0842 06461 010331170000000001416150005431456000             ALU[MEMAC] DEST[HOLD] JUMP[IDIV9] NORM $
12 0843
12m0844                                                         .opcode[231]    [xlist
12 0844                                                   list  ]                                       ;IDIVI
12 0845 06462 010731170000000001404550025771456000             D[IR] MASK[18.] DEST[HOLD] JUMP[IDIV9]  NORM $ ;IMMD
12 0846 06463 010731170000000001404550025771456000             D[IR] MASK[18.] DEST[HOLD] JUMP[IDIV9]  NORM $ ;IMMD
12 0847
12m0848                                                         .opcode[232]    [xlist
12 0848                                                   list  ]                                       ;IDIVM
12 0849 06464 030731171000000001416162025431356000             DF/WT R-M-W JUMP[IDIVMA] $
12 0850 06465 010331170000000001416150005431456000             ALU[MEMAC] DEST[HOLD] JUMP[IDIVMA] NORM $
12 0851
12m0852                                                         .opcode[233]    [xlist
12 0852                                                   list  ]                                       ;IDIVB
12 0853 06466 030731171000000001416162025431356000             DF/WT R-M-W JUMP[IDIVBA] $
12 0854 06467 010331170000000001416150005431456000             ALU[MEMAC] DEST[HOLD] JUMP[IDIVBA] NORM $
12 0855
12 0856
12m0857                                                         .opcode[234]    [xlist
12 0857                                                   list  ]                                       ;DIV
12 0858 06470 030731170000000001416162025431256000             DF/WT JUMP[DIV9] $
12 0859 06471 010331170000000001416150005431456000             ALU[MEMAC] DEST[HOLD] JUMP[DIV9] NORM $
12 0860
12m0861                                                         .opcode[235]    [xlist
12 0861                                                   list  ]                                       ;DIVI
12 0862 06472 010731170000000001404550025771456000             D[IR] MASK[18.] DEST[HOLD] JUMP[DIV9] $ ;IMMD
12 0863 06473 010731170000000001404550025771456000             D[IR] MASK[18.] DEST[HOLD] JUMP[DIV9] $ ;IMMD
12 0864
12m0865                                                         .opcode[236]    [xlist
12 0865                                                   list  ]                                       ;DIVM
12 0866 06474 030731171000000001416162025431356000             DF/WT R-M-W JUMP[DIVMA] $
12 0867 06475 010331170000000001416150005431456000             ALU[MEMAC] DEST[HOLD] JUMP[DIVMA] NORM $
12 0868
12m0869                                                         .opcode[237]    [xlist
12 0869                                                   list  ]                                       ;DIVB
12 0870 06476 030731171000000001416162025431356000             DF/WT R-M-W JUMP[DIVBA] $
12 0871 06477 010331170000000001416150005431456000             ALU[MEMAC] DEST[HOLD] JUMP[DIVBA] NORM $
12 0872
12m0873                                                  .reloc
12m0873                                                  [.USE[NORMAL]
12m0873                                                  [ xlist
12 0874                                                    list ]                                  ]
12 0875       00000037600077776000176740001040000        MEMST1 = SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<Z0] LBJUMP[SEQI] ]$
12 0875                                                  ;Store to memory. (Cause macro expansion now, rather
12 0876                                                                    ;than in macro call.)
12 0877
12 0878
12 0879                                                 .DEFINE DMACFN [AA A2 D1 D2 OP1]
12 0880                                                 [       ALU[AA] D[AR] DEST[O-AC AR] NORM $
12 0881                                                         ACSEL[AC+1] D[AR] ALU[A2] DEST[D1 D2] OP1 $
12 0882                                                         ALU[AC] SMAC $
12 0883                                                   ]
12 0884
12 0885                                                 .DEFINE DMACZ []
12 0886                                                 [       D[MEM] ALU[D#AC] DEST[AR] PUSHJ[DODDIV] NORM $
12 0887                                                   ]
12 0888
12 0889                                                 .DEFINE DMAC1 []
12 0890                                                 [       ALU[-1] DEST[AC] JUMP[. + 2] NORM $
12 0891                                                         ALU[0] DEST[AC] NORM $
12 0892                                                         D[MEM] ALU[D#AC] DEST[AR] NORM $
12 0893                                                         D[AR] ROT[1] MASK[1] DEST[IR-ADR] NORM $
12 0894                                                         ALU[Q] DEST[AR] PUSHJ[DODIV] NORM $
12 0895                                                   ]
12 0896
12 0897 04522 010330206000000001416162125421456000     IDIV9:  ALU[AC] DEST[Q AR] COND[-OBUS<0] LBJUMP[IDIV1] CYLEN[CS00] $
12 0898                                                                 ;LO PART TO Q, CHECK SIGN
12m0899                                                          .pair
12m0899                                                  [:. \ 2 + .
12m0900                                                    ]IDIV1:      DMAC1
12m0900
12m0900 04524 010723170000112554001620255714560000     [       ALU[-1] DEST[AC] JUMP[. + 2] NORM $
12m0900 04525 010243170000060554161623654314160000             ALU[0] DEST[AC] NORM $
12m0900 04526 010561370000060554161623654715160000             D[MEM] ALU[D#AC] DEST[AR] NORM $
12m0900 04527 610731170000060540203443654314160000             D[AR] ROT[1] MASK[1] DEST[IR-ADR] NORM $
12m0900 04530 010231370000000001416162225431456000             ALU[Q] DEST[AR] PUSHJ[DODIV] NORM $
12m0901                                                   ]     DMACFN[Q D AC AR NEOI]
12m0901
12m0901 04531 010232370000060554161623654314160000     [       ALU[Q] D[AR] DEST[O-AC AR] NORM $
12m0901 04532 540733370000060554161623154204160000             ACSEL[AC+1] D[AR] ALU[D] DEST[AC AR] NEOI $
12m0901 04533 010331310000226654161561254304560000             ALU[AC] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<20] LBJUMP[SEQI] ]$
12m0901
12 0902                                                   ]
12 0903 04534 010330206000000001416162125421456000     IDIVMA: ALU[AC] DEST[Q AR] COND[-OBUS<0] LBJUMP[IDIV2]  CYLEN[CS00] $
12 0904                                                                 ;LO PART TO Q, SAVE AC, CHECK SIGN
12m0905                                                          .pair
12m0905                                                  [:. \ 2 + .
12m0906                                                    ]IDIV2:      DMAC1
12m0906
12m0906 04536 010723170000113014001620255714560000     [       ALU[-1] DEST[AC] JUMP[. + 2] NORM $
12m0906 04537 010243170000060554161623654314160000             ALU[0] DEST[AC] NORM $
12m0906 04540 010561370000060554161623654715160000             D[MEM] ALU[D#AC] DEST[AR] NORM $
12m0906 04541 610731170000060540203443654314160000             D[AR] ROT[1] MASK[1] DEST[IR-ADR] NORM $
12m0906 04542 010231370000000001416162225431456000             ALU[Q] DEST[AR] PUSHJ[DODIV] NORM $
12m0907                                                   ]     DMACFN[D Q MEMSTO AR MEMST1]
12m0907
12m0907 04543 010732370000060554161623654314160000     [       ALU[D] D[AR] DEST[O-AC AR] NORM $
```

```
12m0910                                                  .pair
12m0910                                                 [:. \ 2 + .
12m0911                                                   ]IDIV3:      DMAC1
12m0911
12m0911 04550 010723170000011325400162025571456000      [       ALU[-1] DEST[AC] JUMP[. + 2] NORM $
12m0911 04551 010243170000060554161623654314160000              ALU[0] DEST[AC] NORM $
12m0911 04552 010561370000060554161623654716160000              D[MEM] ALU[D#AC] DEST[AR] NORM $
12m0911 04553 610731170000060540203443654314160000              D[AR] ROT[1] MASK[1] DEST[IR-ADR] NORM $
12m0911 04554 010231700000000014161622254314560000              ALU[Q] DEST[AR] PUSHJ[DODIV] NORM $
12m0912                                                    ]     DMACFN[Q D AC AR NORM]
12m0912
12m0912 04555 010232370000060554161623654314160000      [       ALU[Q] D[AR] DEST[D-AC AR] NORM $
12m0912 04556 010733370000060554161623554314160000              ACSEL[AC+1] D[AR] ALU[D] DEST[AC AR] NORM $
12m0912 04557 010331310000226654161561254304560000              ALU[AC] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<20] LBJUMP[SEOI] ]$
12m0912
12 0913                                                    ]
12 0914 04560 010330170000060554161623554314160000      DIV9:   ACSEL[AC+1] ALU[AC] DEST[Q] NORM $ ;LO PART
12m0915                                                          DMAC2
12m0915
12m0915 04561 010561370000000014161622254715560000      [       D[MEM] ALU[D#AC] DEST[AR] PUSHJ[DODDIV] NORM $
12m0916                                                    ]     DMACFN[Q D AC AR NEOI]
12m0916
12m0916 04562 010232370000060554161623654314160000      [       ALU[Q] D[AR] DEST[D-AC AR] NORM $
12m0916 04563 540733370000060554161623154204160000              ACSEL[AC+1] D[AR] ALU[D] DEST[AC AR] NEOI $
12m0916 04564 010331310000226654161561254304560000              ALU[AC] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<20] LBJUMP[SEOI] ]$
12m0916
12 0917                                                    ]
12 0918 04565 010330170000060554161623554314160000      DIVMA:  ALU[AC] ACSEL[AC+1] DEST[Q] NORM $
12m0919                                                          DMAC2
12m0919
12m0919 04566 010561370000000014161622254715560000      [       D[MEM] ALU[D#AC] DEST[AR] PUSHJ[DODDIV] NORM $
12m0920                                                    ]     DMACFN[D Q MEMSTO AR MEMST1]
12m0920
12m0920 04567 010732370000060554161623654314160000      [       ALU[D] D[AR] DEST[D-AC AR] NORM $
12m0920 04570 010231310000226654161561154304560000              ACSEL[AC+1] D[AR] ALU[Q] DEST[MEMSTO AR] MEMST1 $
12m0920 04571 010331310000226654161561254304560000              ALU[AC] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<20] LBJUMP[SEOI] ]$
12m0920
12 0921                                                    ]
12 0922 04572 010330170000060554161623554314160000      DIVBA:  ALU[AC] ACSEL[AC+1] DEST[Q] NORM $
12m0923                                                          DMAC2
12m0923
12m0923 04573 010561370000000014161622254715560000      [       D[MEM] ALU[D#AC] DEST[AR] PUSHJ[DODDIV] NORM $
12m0924                                                    ]     DMACFN[Q D AC AR NORM]
12m0924
12m0924 04574 010232370000060554161623654314160000      [       ALU[Q] D[AR] DEST[D-AC AR] NORM $
12m0924 04575 010733370000060554161623554314160000              ACSEL[AC+1] D[AR] ALU[D] DEST[AC AR] NORM $
12m0924 04576 010331310000226654161561254304560000              ALU[AC] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<20] LBJUMP[SEOI] ]$
12m0924
12 0925                                                    ]
12 0926 04577 610731170000060540203443654374160000      DODDIV: D[AR] ROT[1] MASK[1] DEST[IR-ADR] SHORT $
12 0927 04600 010435370000060554003623654314160000              ALU[SH-AC] DEST[AR DS] MASK[1] NORM $
12 0928 04601 010731002000114074003620254214560000              D[AR] MASK[1] COND[OBUS=0] JUMP[. + 2] C550 $
12 0929 04602 010630170000000010603620255714560000              D[CONST 1] ROT[43] ALU[DORQ] DEST[Q] JUMP[DODIV] NORM $
12 0930 04603 010640170000000001410762024731456000              D[MASK 43] ALU[D&Q] DEST[Q] NORM JUMP[DODIV] $
12 0931
12 0932                                                  ;-----------------------------------------------------------------------
12 0932                                                  ------
12 0933                                                  ;
12 0934                                                  ;       Single Precision Divide (and continuation of double precision)
12 0935                                                  ;
12 0936                                                  ;       (Reminder: Quotient, Remainder = Dividend / Divisor)
12 0937                                                  ;
12 0938                                                  ;Where we get here:
12 0939                                                  ;    MEM        36 bit signed divisor
12 0940                                                  ;    AC         High order dividend (0 or -1 for single precision)
12 0941                                                  ;    Q          Low order dividend
12 0942                                                  ;    IR<35>     Sign of dividend XOR sign of divisor
12 0943                                                  ;    AR         Sign of dividend and original contents of AC
12 0944                                                  ;
12 0945                                                  ;-----------------------------------------------------------------------
12 0945                                                  ------
12 0946 04604 010731006000000001416162025461556000      DODIV:  D[MEM] COND[-OBUS<0] JUMP[DODIV1] C550 $
12 0947                                                          ;Jump if divide by positive number
12 0948 04605 011711170000060554161503654775716000              D[MEM] ALU[0-D] DEST[HOLD] SHORT $
12 0949                                                          ;Take absolute value
12 0950 04606 010331006000000001416162025411456000      DODIV1: ALU[AC] COND[-OBUS<0] JUMP[DODIV2] CYLEN[C450] $
12 0951                                                          ;Check sign of high order word.  If positive, we're ready
12 0951                                                  to go
12 0952 04607 011ZZ000200000000141616202542145600               ALU[0-Q] DEST[Q] COND[OBUS=0] JUMP[DODIV3] CYLEN[C500] $
12 0953                                                          ;Double precision negate, low order word
12 0954 04610 010373170000000001416162025431456000              ALU[NOTAC] DEST[AC] JUMP[DODIV2] NORM $
12 0955                                                          ;High order word, no carry
12 0956 04611 011323170000060554161623654314160000      DODIV3: ALU[0-AC] DEST[AC] NORM $
12 0957                                                          ;High order word, with carry
12 0958 04612 010731170010060540110003655714170000      DODIV2: D[CONST 44] ROT[LLOAD-ROT] LLOAD NORM $
12 0959                                                          ;LOOP 37 TIMES
12 0960                                                  ;Now have:
12 0961                                                  ;   MEM         Absolute value of divisor
12 0962                                                  ;   AC          Absolute value of high order dividend
12 0963                                                  ;   Q           Absolute value of low order dividend
12 0964                                                  ;   R           Repeat count for division
12 0965 04613 011511004000000001416162025461556000              D[MEM] ALU[AC-D] COND[OBUS<0] JUMP[DODIV7] C550 $
12 0966                                                          ;Jump if not no divide case
12 0967 04614 010733170000060554161623646374160000              D[PC-FLAGS] DEST[AC] SHORT $
12 0968                                                          ;Get ready to set flags
12 0969 04615 010533170000010770560362225574156000              D[CONST 1] ROT[23.] ALU[DORAC] DEST[AC] PUSHJ[MSTOV1] NORM $
12 0970                                                          ;Set no divide
12 0971 04616 010733170000060554161624254314160000              D[AR] DEST[AC] POPJ NORM $
12 0972                                                          ;Fix clobbered AC
12 0973
12 0974 04617 010731170000000001416162225431456000      DODIV7: NORM PUSHJ[DODIV4] $
12 0975 04620 010731006000011445416162025421456000              D[AR] COND[-OBUS<0] JUMP[. + 2] C550 $; J IF DIVIDEND WAS +
12 0976 04621 011323170000060554161623654374160000              ALU[0-AC] DEST[AC] SHORT $
12 0977 04622 010731002000060554003624257614160000              D[IR] MASK[1] COND[OBUS=0] POPJ C550 $ ;LEAVE IF RESULT SHOULD BE
12 0977                                                  +
12 0978 04623 011220170000060554161624254314160000              ALU[0-Q] DEST[Q] POPJ NORM $
12 0979                                                  ;On completion:
12 0980                                                  ;   MEM         Absolute value of divisor
12 0981                                                  ;   AC          Remainder
12 0982                                                  ;   Q           Quotient
12 0983                                                  ;   AR(0)      Sign of result
```

```
12 0988                                                          ;For ordinary floating divide, do 8 fewer steps than fixed
12 0988                                                        point.
12 0989 04625 010731170000000045416162025431456000                    NORM JUMP[DODIV4 + 18.] $
12 0990                                                          ;For floating divide long, do 9 fewer steps than fixed point.
12 0991
12 0992                                                        DODIV4:
12m0993                                                          .even
12m0993                                                        [:. \ 2 + .
12 0994                                                          ]  .repeat 36. [
12 0995                                                              D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12 0995                                                        T300 $
12 0996                                                              D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12 0996                                                        T300 $
12m0997                                                          ]
12m0997                                                        [
12m0997 04626 011516004000114614167621254715560000                  D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                        T300 $
12m0997 04627 010506004000114614167621254715560000                  D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                        T300 $
12m0997                                                          ][
12m0997 04630 011516004000114654167621254715560000                  D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                        T300 $
12m0997 04631 010506004000114654167621254715560000                  D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                        T300 $
12m0997                                                          ][
12m0997 04632 011516004000114714167621254715560000                  D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                        T300 $
12m0997 04633 010506004000114714167621254715560000                  D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                        T300 $
12m0997                                                          ][
12m0997 04634 011516004000114754167621254715560000                  D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                        T300 $
12m0997 04635 010506004000114754167621254715560000                  D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                        T300 $
12m0997                                                          ][
12m0997 04636 011516004000115014167621254715560000                  D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                        T300 $
12m0997 04637 010506004000115014167621254715560000                  D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                        T300 $
12m0997                                                          ][
12m0997 04640 011516004000115054167621254715560000                  D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                        T300 $
12m0997 04641 010506004000115054167621254715560000                  D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                        T300 $
12m0997                                                          ][
12m0997 04642 011516004000115114167621254715560000                  D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                        T300 $
12m0997 04643 010506004000115114167621254715560000                  D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                        T300 $
12m0997                                                          ][
12m0997 04644 011516004000115154167621254715560000                  D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0? LBJUMP[. + 2]
12m0997                                                        T300 $
12m0997 04645 010506004000115154167621254715560000                  D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                        T300 $
12m0997                                                          ][
12m0997 04646 011516004000115214167621254715560000                  D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                        T300 $
12m0997 04647 010506004000115214167621254715560000                  D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                        T300 $
12m0997                                                          ][
12m0997 04650 011516004000115254167621254715560000                  D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                        T300 $
12m0997 04651 010506004000115254167621254715560000                  D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                        T300 $
12m0997                                                          ][
12m0997 04652 011516004000115314167621254715560000                  D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                        T300 $
12m0997 04653 010506004000115314167621254715560000                  D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                        T300 $
12m0997                                                          ][
12m0997 04654 011516004000115354167621254715560000                  D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                        T300 $
12m0997 04655 010506004000115354167621254715560000                  D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                        T300 $
12m0997                                                          ][
12m0997 04656 011516004000115414167621254715560000                  D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                        T300 $
12m0997 04657 010506004000115414167621254715560000                  D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                        T300 $
12m0997                                                          ][
12m0997 04660 011516004000115454167621254715560000                  D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                        T300 $
12m0997 04661 010506004000115454167621254715560000                  D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                        T300 $
12m0997                                                          ][
12m0997 04662 011516004000115514167621254715560000                  D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                        T300 $
12m0997 04663 010506004000115514167621254715560000                  D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                        T300 $
12m0997                                                          ][
12m0997 04664 011516004000115554167621254715560000                  D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                        T300 $
12m0997 04665 010506004000115554167621254715560000                  D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                        T300 $
12m0997                                                          ][
12m0997 04666 011516004000115614167621254715560000                  D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                        T300 $
12m0997 04667 010506004000115614167621254715560000                  D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                        T300 $
12m0997                                                          ][
12m0997 04670 011516004000115654167621254715560000                  D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                        T300 $
12m0997 04671 010506004000115654167621254715560000                  D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                        T300 $
12m0997                                                          ][
12m0997 04672 011516004000115714167621254715560000                  D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                        T300 $
12m0997 04673 010506004000115714167621254715560000                  D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                        T300 $
12m0997                                                          ][
```

```
12m0997 04676 011516004000116014167621254715560000          D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                    T300 $
12m0997 04677 010506004000116014167621254715560000          D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                    T300 $
12m0997                                                        ][
12m0997 04700 011516004000116054167621254715560000          D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                    T300 $
12m0997 04701 010506004000116054167621254715560000          D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                    T300 $
12m0997                                                        ][
12m0997 04702 011516004000116114167621254715560000          D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                    T300 $
12m0997 04703 010506004000116114167621254715560000          D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                    T300 $
12m0997                                                        ][
12m0997 04704 011516004000116154167621254715560000          D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                    T300 $
12m0997 04705 010506004000116154167621254715560000          D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                    T300 $
12m0997                                                        ][
12m0997 04706 011516004000116214167621254715560000          D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                    T300 $
12m0997 04707 010506004000116214167621254715560000          D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                    T300 $
12m0997                                                        ][
12m0997 04710 011516004000116254167621254715560000          D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                    T300 $
12m0997 04711 010506004000116254167621254715560000          D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                    T300 $
12m0997                                                        ][
12m0997 04712 011516004000116314167621254715560000          D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                    T300 $
12m0997 04713 010506004000116314167621254715560000          D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                    T300 $
12m0997                                                        ][
12m0997 04714 011516004000116354167621254715560000          D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                    T300 $
12m0997 04715 010506004000116354167621254715560000          D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                    T300 $
12m0997                                                        ][
12m0997 04716 011516004000116414167621254715560000          D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                    T300 $
12m0997 04717 010506004000116414167621254715560000          D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                    T300 $
12m0997                                                        ][
12m0997 04720 011516004000116454167621254715560000          D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                    T300 $
12m0997 04721 010506004000116454167621254715560000          D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                    T300 $
12m0997                                                        ][
12m0997 04722 011516004000116514167621254715560000          D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                    T300 $
12m0997 04723 010506004000116514167621254715560000          D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                    T300 $
12m0997                                                        ][
12m0997 04724 011516004000116554167621254715560000          D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                    T300 $
12m0997 04725 010506004000116554167621254715560000          D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                    T300 $
12m0997                                                        ][
12m0997 04726 011516004000116614167621254715560000          D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                    T300 $
12m0997 04727 010506004000116614167621254715560000          D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                    T300 $
12m0997                                                        ][
12m0997 04730 011516004000116654167621254715560000          D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                    T300 $
12m0997 04731 010506004000116654167621254715560000          D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                    T300 $
12m0997                                                        ][
12m0997 04732 011516004000116714167621254715560000          D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                    T300 $
12m0997 04733 010506004000116714167621254715560000          D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                    T300 $
12m0997                                                        ][
12m0997 04734 011516004000116754167621254715560000          D[MEM] ALU[AC-D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 2]
12m0997                                                    T300 $
12m0997 04735 010506004000116754167621254715560000          D[MEM] ALU[AC+D] DEST[D6] MASK[73] COND[OBUS<0] LBJUMP[. + 1]
12m0997                                                    T300 $
12 0997 04736 011516170000117014167620254715560000            ]  D[MEM] ALU[AC-D] DEST[D6] MASK[73]  JUMP[. + 2] NORM $
12 0998 04737 010506170000060554167623654715160000             D[MEM] ALU[AC+D] DEST[D6] MASK[73]  NORM $
12 0999
12 1000 04740 010435170000060554001623654314160000          ALU[SH-AC] DEST[DS] MASK[0] NORM $
12 1001 04741 010563006000060550603624255614160000        DODIV6: D[CONST 1] ROT[43] ALU[D#AC] DEST[AC] COND[-OBUS<0] C550 POPJ $
12 1001
12 1002                                                              ;ADJUST REM SIGN, CHECK IT
12 1003 04742 010503170000060554161624254715160000             D[MEM] ALU[D+AC] DEST[AC] NORM POPJ $
12 1004                                                              ;ADJUST REM.
12 1005
12 1006
12 1007
12 1008                                                    COMMENT %
12 1009                                                          .QUAD
12 1010                                                    DODIV4: D[MEM] ALU[DIVAC-D] DEST[D6] MASK[3] COND[OBUS<0] SLOOP[DODIV4]
12 1010                                                     C600 $
12 1011                                                          ALU[SH-AC] DEST[DS] MASK[0] JUMP[DODIV6] NORM $ ; RE-SHIFT
12 1011                                                     REMAINDER
12 1012                                                          D[MEM] ALU[DIVAC+D] DEST[D6] MASK[3] COND[OBUS<0] SLOOP[DODIV4]
12 1012                                                     C600 $
12 1013                                                          ALU[SH-AC] DEST[DS] MASK[0] NORM $
12 1014                                                       ;end-of-.QUAD
12 1015                                                    DODIV6: D[CONST 1] ROT[43] ALU[D#AC] DEST[AC] COND[-OBUS<0] C550 POPJ $
12 1015
12 1016                                                              ;ADJUST REM SIGN, CHECK IT
12 1017                                                          D[MEM] ALU[D+AC] DEST[AC] NORM POPJ $
12 1018                                                              ;ADJUST REM.
12 1019                                                     %
12 1020
12 1021
```

```
13 1022
13 1023                                                 ;;; Shift/rotate group
13m1024                                                   .opcode[240]   [xlist
13 1024                                                    list  ]                                                    ;ASH
13 1025 06500 010730022000000001416162125771456000             D[IR] COND[-OBUS18] DEST[Q] LBJUMP[ASH1] $
13 1026 06501 010730170000000001416044025771457000             D[IR] DEST[Q ROTR] JUMP[ASHDOP] $ ;count is small positive
13 1027
13m1028                                                    .reloc
13m1028                                                   [.USE[NORMAL]
13m1028                                                   [ xlist
13 1029                                                    list ]                                  ]
13m1030                                                    .pair
13m1030                                                   [:. \ 2 + .
13 1031 04744 010630170000000002045620247314560000         ]ASH1:       D[MASK 22] ROT[10] ALU[DORQ] DEST[Q] JUMP[ASHNEG] $ ; neg
13 1031                                                   arg
13 1032 04745 010730170000060554020443657714170000             D[IR] MASK[10] DEST[Q ROTR] $   ;Pos. count, mask
13 1033 04746 011611020000000041411620255614560000             D[CONST 44] ALU[Q-D] C550 COND[OBUS18] JUMP[ASHDOP] $
13 1034 04747 011610170000000041411044025571457000             D[CONST 44] ALU[Q-D] DEST[Q ROTR] JUMP[BIGASH] $
13 1035
13 1036 04750 010600022000000041411044025571457000       ASHNEG: D[CONST 44] ALU[D+Q] DEST[Q ROTR] COND[-OBUS18] JUMP[ASHDO] $
13 1037 04751 010731170000000041416162025431456000             JUMP[BIGASZ] $
13 1038
13 1039 04752 010331002000022711416162025431456000       BIGASH: ALU[AC] COND[OBUS=0] JUMP[MAIN] $
13 1040 04753 010543170000060550603623655714160000             D[CONST 1] ROT[43] ALU[D&AC] DEST[AC] $
13 1041 04754 010730170000060554161623646314160000             D[PC-FLAGS] DEST[Q] $
13 1042                                                 SETOV1: D[CONST 1] ROT[43] ALU[DORQ] DEST[PC-FLAGS]
13 1043                                                 .repeat waitsfix [
13 1044                                                               C550 PUSHJ[SETOVX] $
13m1045                                                 ];[
13m1045 04755 010631170000000001060242225561457000             C550 PUSHJ[SETOVX] $
13 1045                                                 ].repeat waitsfix
13 1046                                                           ;Set PC flag bit 0, and go cause trap if enabled.
13 1047 04756 540731170000060554161623254204160000             NEOI $
13 1048
13 1049 04757 010331206000000041416162025431456000       ASHDO:  ALU[AC] DEST[AR] COND[-OBUS<0] JUMP[ASHDO1] $
13 1050 04760 011621170000060554110503655714170000             D[CONST 44] ALU[D-Q] DEST[MASKR] $
13 1051 04761 010730170000060552121623647314160000             D[MASKR] ROT[R] DEST[Q] $
13 1052 04762 540633170000060552161623254204160000             D[AR] ROT[R] ALU[DORQ] DEST[AC] NEOI $
13 1053
13 1054 04763 010231170000060554160503654314170000       ASHDO1: ALU[Q] DEST[MASKR] $
13 1055 04764 540733170000060552121623254204160000             D[AR] MASK[R] ROT[R] DEST[AC] NEOI $
13 1056
13 1057 04765 011621170000060554106503655714170000       ASHDOP: D[CONST 43] ALU[D-Q] DEST[MASKR] $
13 1058 04766 610231170000060554161443654314160000             ALU[Q] DEST[IR-ADR] $: SAVE SH AMT
13 1059 04767 010540170000060550603623655714160000             D[CONST 1] ROT[43] ALU[D&AC] DEST[Q] $ ;GET SIGN BIT
13 1060 04770 010541370000060554121623647314160000             D[MASKR] ALU[D&AC] DEST[AR] $
13 1061 04771 010632370000060552161623654314160000             D[AR] ROT[R] ALU[DORQ] DEST[0-AC AR] $
13 1062 04772 011701170000060554160503657714170000             D[IR] ALU[D+1] DEST[MASKR] $
13 1063 04773 010730170000060552161623654314160000             D[AR] ROT[R] DEST[Q] $
13 1064 04774 010640002000227110721620247314560000             D[MASKR] ROT[43] ALU[D&Q] DEST[Q] COND[OBUS=0] JUMP[MAIN] $
13 1065 04775 010661002000227110721620247314560000             D[MASKR] ROT[43] ALU[D#Q] COND[OBUS=0] JUMP[MAIN] $
13 1066 04776 010730170000117334161620246314560000             D[PC-FLAGS] DEST[Q] JUMP[SETOV1] NORM $
13 1067
13 1068 04777 010331006000000041416162025431456000       BIGASZ: ALU[AC] COND[-OBUS<0] JUMP[LSH0] $
13 1069 05000 540723170000060554001623255604160000             ALU[-1] DEST[AC] NEOI $
13 1070
13m1071                                                    .opcode[241]   [xlist
13 1071                                                    list  ]                                                    ;ROT
13 1072 06502 010730022000000001416162125771456000             D[IR] COND[-OBUS18] DEST[Q] LBJUMP[ROT1] $
13 1073 06503 010731170000000001416044025771457000             D[IR] DEST[ROTR] JUMP[ROTDO] $  ;count is small positive
13 1074
13m1075                                                    .reloc
13m1075                                                   [.USE[NORMAL]
13m1075                                                   [ xlist
13m1076                                                    list ]                                  ]  .pair
13m1076                                                   [:. \ 2 + .
13 1077 05002 010630170000000000202562024731456000         ]ROT1:       D[MASK 12] ROT[10] ALU[DORQ] DEST[Q] JUMP[ROTNEG] $
13 1078                                                                 ;Negative count, extend sign
13 1079 05003 010730170000060554020443657714170000             D[IR] MASK[10] DEST[Q ROTR] $   ;Pos. count, mask
13 1080 05004 011611020000000041411620255714560000             D[CONST 44] ALU[Q-D] COND[OBUS18] JUMP[ROTDO] $ ;Mod 36.
13 1081 05005 011610170000120114110440255714570000             D[CONST 44] ALU[Q-D] DEST[Q ROTR] JUMP[. - 1] $
13 1082
13 1083 05006 010600020000120154110440255714570000       ROTNEG: D[CONST 44] ALU[Q+D] DEST[Q ROTR] COND[OBUS18] JUMP[.] $
13 1084 05007 010331370000060554161623654314160000       ROTDO:  ALU[AC] DEST[AR] $
13 1085 05010 540733170000060552161623254204160000             D[AR] ROT[R] DEST[AC] NEOI $
13 1086
13m1087                                                    .opcode[242]   [xlist
13 1087                                                    list  ]                                                    ;LSH
13 1088 06504 010730022000000001416162125771456000             D[IR] COND[-OBUS18] DEST[Q] LBJUMP[LSH1] $
13 1089 06505 010730170000000001416044025771457000             D[IR] DEST[Q ROTR] JUMP[LSHDO] $ ;count is small positive
13 1090
13m1091                                                    .reloc
13m1091                                                   [.USE[NORMAL]
13m1091                                                   [ xlist
13m1092                                                    list ]                                  ]  .pair
13m1092                                                   [:. \ 2 + .
13 1093 05012 010630170000000000202562024731456000         ]LSH1:       D[MASK 12] ROT[10] ALU[DORQ] DEST[Q] JUMP[LSHNEG] $
13 1094                                                                 ;Negative count, extend sign
13 1095 05013 010730170000060554020443657714170000             D[IR] MASK[10] DEST[Q ROTR] $   ;Pos. count, mask
13 1096 05014 011611012000000041411162025571456000       LSHX:   D[CONST 44] ALU[Q-D] COND[OBUS>0] JUMP[LSH0] $ ;ASH joins here
13 1097 05015 011621170000060554110503655714170000       LSHDO:  D[CONST 44] ALU[D-Q] DEST[MASKR] $
13 1098 05016 010541370000060554121623647314160000             D[MASKR] ALU[D&AC] DEST[AR] $
13 1099 05017 540733170000060552161623254204160000             D[AR] ROT[R] DEST[AC] NEOI $
13 1100
13 1101 05020 540243170000060554161623254204160000       LSH0:   ALU[0] DEST[AC] NEOI $
13 1102
13 1103 05021 010600020000120414110440255714570000       LSHNEG: D[CONST 44] ALU[D+Q] DEST[Q ROTR] COND[OBUS18] JUMP[LSH0] $
13 1104 05022 010231170000060554160503654314170000             ALU[Q] DEST[MASKR] $
13 1105 05023 010331370000060554161623654314160000             ALU[AC] DEST[AR] $
13 1106 05024 540733170000060552121623254204160000             D[AR] ROT[R] MASK[R] DEST[AC] NEOI $
13 1107
```

```
14 1108
14m1109                                                  .opcode[243]   [xlist
14 1109                                                   list ]                                                    ;JFFO
14 1110 06506 010331200000000001416162125431456000              ALU[AC] DEST[AR] COND[-OBUS=0] LBJUMP[JFFO1] $
14 1111 06507 010331200000000001416162125431456000              ALU[AC] DEST[AR] COND[-OBUS=0] LBJUMP[JFFO1] $
14 1112
14m1113                                                  .reloc
14m1113                                                 [.USE[NORMAL]
14m1113                                                 [ xlist
14m1114                                                   list ]                                  ] .pair
14m1114                                                 [:. \ Z + .
14 1115 05026 540243170000060554161623154204160000       ]JFFO1:      ACSEL[AC+1] ALU[0] DEST[AC] NEOI $
14 1116 05027 010243170000060554161623554314160000              ACSEL[AC+1] ALU[0] DEST[AC] $
14 1117 05030 010731000000000001416562025431456000       JFFO2: D[AR] ROT[6] MASK[6] COND[-OBUS=0] JUMP[JFFO3] $
14 1118 05031 010731370000060541561623654314160000              D[AR] ROT[6] DEST[AR] NORM $
14 1119 05032 010503170000120614015620155571456000              D[CONST 6] ACSEL[AC+1] ALU[AC+D] DEST[AC] JUMP[JFFO2] NORM $
14 1120 05033 010731004000000001416162025431456000       JFFO3: D[AR] COND[OBUS<0] JUMP[JFFO4] $
14 1121 05034 011303170000060554161623554314160000       JFFO5: ACSEL[AC+1] ALU[AC+1] DEST[AC] NORM $
14 1122 05035 010731206000120700361620254314560000              D[AR] ROT[1] DEST[AR] COND[-OBUS<0] JUMP[JFFO5] $
14 1123 05036 550731170000277554161623254212160000       JFFO4: JDSP $
14 1124
14 1125                                                  .DEFINE SHLOOP [SHTYP DIR]
14 1126                                                 [       D[IR] MASK[8.] ALU[D-1] DEST[AR] $
14 1127                                                         D[AR] ROT[LLOAD-ROT] DEST[LLOAD] $
14 1128                                                         ALU[SH-AC] DEST[DIR] ENDCONN[SHTYP] NORM LOOP[.] $
14 1129                                                 ]
14 1130
14m1131                                                  .opcode[244]   [xlist
14 1131                                                   list ]                                                    ;ASHC
14 1132 06510 010541370000000001410762014731456000              ACSEL[AC+1] D[MASK 43] ALU[D&AC] DEST[AR] JUMP[ASHC1] $
14 1133 06511 010541370000000001410762014731456000              ACSEL[AC+1] D[MASK 43] ALU[D&AC] DEST[AR] JUMP[ASHC1] $
14 1134
14m1135                                                  .reloc
14m1135                                                 [.USE[NORMAL]
14m1135                                                 [ xlist
14 1136 05037 010730170000060540361623654314160000       list ]                                  ]ASHC1:      D[AR] ROT[1]
14 1136                                                 DEST[Q] $
14 1137 05040 010541170000060550603503655571416000              D[CONST 1] ROT[43] ALU[D&AC] DEST[HOLD] NORM $
14 1138                                                                 ; Save sign bit
14 1139 05041 010731020000000014045620257714560000              D[IR] MASK[18.] COND[OBUS18] JUMP[ASHC2] $
14 1140 05042 010731002000227114021620257714560000              D[IR] MASK[10] COND[OBUS=0] JUMP[MAIN]  $
14 1141 05043 010243170000060554161623554314160000              ALU[0] ACSEL[AC+1] DEST[AC] $
14 1142 05044 010721370000060554021623657714160000              D[IR] MASK[10] ALU[D-1] DEST[AR] $
14 1143 05045 010731170010060540160003654314170000              D[AR] ROT[LLOAD-ROT] DEST[LLOAD] $
14 1144 05046 010436170000060554003623654314160000       ASHC8: ALU[SH-AC] DEST[D6] MASK[1] NORM $
14 1145 05047 010561004000000001416162025471556000              D[MEM] ALU[D#AC] COND[OBUS<0] JUMP[ASHC9] $
14 1146 05050 010731070000121154161620254314560000       ASHC6: LOOP[ASHC8] NORM $
14 1147                                                  ;;;    D[PC] ACSEL[AC+1] ALU[DORAC] DEST[CRYOV] JUMP[ASHC7] NORM $
14 1148 05051 010731170000000001416162025431456000              JUMP[ASHC7] $           ; +++ have to set PC-FLAGS somehow
14 1149 05052 010733170000121210603620155571456000       ASHC9: D[CONST 1] ROT[43] ACSEL[AC+1] DEST[AC] JUMP[ASHC6] NORM $
14 1150
14 1151 05053 010730170000060554161623654775160000       ASHC4: D[MEM] DEST[Q] SHORT $
14 1152 05054 010633170000060550707623554314160000              D[AR] ROT[43] MASK[43] ALU[DORQ] ACSEL[AC+1] DEST[AC] NORM $
14 1153 05055 010543170000060554107623647374160000              D[MASK 43] ALU[D&AC] DEST[AC] SHORT $
14 1154 05056 540033170000060554161623254204160000              ALU[QORAC] DEST[AC] NEOI $
14 1155 05057 611711170000060554161443657714160000       ASHC2: D[IR] ALU[0-D] DEST[IR-ADR] NORM $
14m1156                                                         SHLOOP [ ASH D4 ]
14m1156
14m1156 05060 010721370000060554021623657714160000       [      D[IR] MASK[8.] ALU[D-1] DEST[AR] $
14m1156 05061 010731170010060540160003654314170000              D[AR] ROT[LLOAD-ROT] DEST[LLOAD] $
14m1156 05062 010434070000121454163620254314560000              ALU[SH-AC] DEST[D4] ENDCONN[ASH] NORM LOOP[.] $
14 1157 05063 010231370000121274161620254314560000       ]ASHC7: ALU[Q] DEST[AR] JUMP[ASHC4] NORM $
14 1158
14m1159                                                  .opcode[245]   [xlist
14 1159                                                   list ]                                                    ;ROTC
14 1160 06512 010731022000000001416162125771456000              D[IR] COND[-OBUS18] LBJUMP[ROTC1] $
14 1161 06513 010330170000000001416162015431456000              ACSEL[AC+1] ALU[AC] DEST[Q] JUMP[ROTCA1] $
14 1162
14m1163                                                  .reloc
14m1163                                                 [.USE[NORMAL]
14m1163                                                 [ xlist
14m1164                                                   list ]                                  ] .pair
14m1164                                                 [:. \ Z + .
14 1165 05064 611711170000000001416144025771456000       ]ROTC1:      D[IR] ALU[0-D] DEST[IR-ADR] JUMP[ROTCZ] $ ; neg shift
14 1165                                                 count
14 1166 05065 010330170000060554161623554314160000              ACSEL[AC+1] ALU[AC] DEST[Q]   NORM $
14 1167 05066 010731002000227114021620257714560000       ROTCA1: D[IR] MASK[10] COND[OBUS=0] JUMP[MAIN] $
14m1168                                                         SHLOOP[ ROT D6 ]
14m1168
14m1168 05067 010721370000060554021623657714160000       [      D[IR] MASK[8.] ALU[D-1] DEST[AR] $
14m1168 05070 010731170010060540160003654314170000              D[AR] ROT[LLOAD-ROT] DEST[LLOAD] $
14m1168 05071 010436070000121634161620254314560000              ALU[SH-AC] DEST[D6] ENDCONN[ROT] NORM LOOP[.] $
14 1169 05072 540233170000060554161623154204160000       ]      ALU[Q] DEST[AC] ACSEL[AC+1] NEOI $
14 1170
14 1171 05073 010330170000060554161623554314160000       ROTCZ: ACSEL[AC+1] ALU[AC] DEST[Q]   NORM $
14m1172                                                         SHLOOP[ ROT D4 ]
14m1172
14m1172 05074 010721370000060554021623657714160000       [      D[IR] MASK[8.] ALU[D-1] DEST[AR] $
14m1172 05075 010731170010060540160003654314170000              D[AR] ROT[LLOAD-ROT] DEST[LLOAD] $
14m1172 05076 010434070000121754161620254314560000              ALU[SH-AC] DEST[D4] ENDCONN[ROT] NORM LOOP[.] $
14 1173 05077 540233170000060554161623154204160000       ]      ALU[Q] DEST[AC] ACSEL[AC+1] NEOI $
14 1174
14m1175                                                  .opcode[246]   [xlist
14 1175                                                   list ]                                                    ;LSHC
14 1176 06514 010731022000000001416162125771456000              D[IR] COND[-OBUS18] LBJUMP[LSHC1] $
14 1177 06515 010330170000000001416162015431456000              ACSEL[AC+1] ALU[AC] DEST[Q] JUMP[LSHCA1] $
14 1178
14m1179                                                  .reloc
14m1179                                                 [.USE[NORMAL]
14m1179                                                 [ xlist
14m1180                                                   list ]                                  ] .pair
14m1180                                                 [:. \ Z + .
14 1181 05100 611711170000000001404544025771456000       ]LSHC1:      D[IR] MASK[18.] ALU[0-D] DEST[IR-ADR] JUMP[LSHCZ] $
14 1182 05101 010330170000060554161623554314160000              ACSEL[AC+1] ALU[AC] DEST[Q] $
14 1183 05102 010731002000227114021620257714560000       LSHCA1: D[IR] MASK[10] COND[OBUS=0] JUMP[MAIN] $
14m1184                                                         SHLOOP [ LSH D6 ]
14m1184
14m1184 05103 010721370000060554021623657714160000       [      D[IR] MASK[8.] ALU[D-1] DEST[AR] $
14m1184 05104 010731170010060540160003654314170000              D[AR] ROT[LLOAD-ROT] DEST[LLOAD] $
14m1184 05105 010436070000122134165620254314560000              ALU[SH-AC] DEST[D6] ENDCONN[LSH] NORM LOOP[.] $
```

```
SLOE  March 23, 1984 21:05:39  file STRING:  --  of -- F41NNF

14m1188 05110 010721370000060554021623657714160000       [       D[IR] MASK[8.] ALU[D-1] DEST[AR] $
14m1188 05111 010731170010060540160003654314170000               D[AR] ROT[LLOAD-ROT] DEST[LLOAD] $
14m1188 05112 010434070000122254165620254314560000               ALU[SH-AC] DEST[D4] ENDCONN[LSH] NORM LOOP[.] $
14 1189 05113 540233170000060554161623154204160000       ]       ALU[Q] DEST[AC] ACSEL[AC+1] NEOI $
14 1190
```

```
1S 1191
1Sm1192                                                 .opcode[ZS0]   [xlist
1S 1192                                                  list  ]                                                    ;EXCH
1S 1193 06520 010331170000000014161500254312560000           DFRQ ALU[AC] DEST[HOLD] JUMP[EXCH1] $
1S 1194 06521 010430170000000014161620254314560000           ALU[A] DEST[Q] JUMP[EXCHA] $
1S 1195
1Sm1196                                                 .reloc
1Sm1196                                                 [.USE[NORMAL]
1Sm1196                                                 [ xlist
1S 1197 05114 010733171000226654161640254705560000      list ]                                  ]EXCH1:      IFRQ STRT-WRT
1S 1197                                                 R-M-W D[MEM] DEST[AC] JUMP[SEQI] $
1S 1198 05115 010433170000060554161623414314160000     EXCHA:  ACSEL[MA,AC] ALU[A] DEST[B] $
1S 1199 05116 540233170000060554161623054204360000           ALU[Q] DEST[MEMAC] NEQI $
1S 1200
1S 1201
```

```
16 1202
16 1203                                 .REPEAT OLDBLT [
16 1204                                 ;;; "Tasteful BLT"
16 1205                                 ;+++ Mem-to-mem loop can be cut to two cycles, probably
16 1206
16 1207                                   .opcode[251]
16 1207                                  ;BLT
16 1208                                         D[MASK 14.] ROT[4] ALU[D&AC] COND[-OBUS=0] LBJUMP[BLT0] $
16 1208
16 1209                                         D[MASK 14.] ROT[4] ALU[D&AC] COND[-OBUS=0] LBJUMP[BLT0] $
16 1210                                                 ; Test for dest<20
16 1211
16 1212                                   .reloc
16 1213                                   .pair
16 1214                                 BLT0:   JUMP[BLTA1] $           ;dest<20
16 1215                                         D[MASK 14.] ROT[18. + 4] ALU[D&AC] COND[OBUS=0] JUMP[BLTA2] $
16 1216                                                 ; Jump if src<20
16 1217
16 1218                                         D[PC] DEST[O-AC AR] $   ;AC to AR, PC to AC
16 1219                                         D[AR] MASK[18.] DEST[Q PC] $ ;dest to Q and PC
16 1220                                 ;+++    D[AR] ROT[18.] MASK[18.] ALU[Q-D-1] COND[OBUS=0] JUMP[BLTZ] $
16 1220
16 1221                                         D[IR] MASK[18.] ALU[D-Q] DEST[Q HOLD] COND[OBUS<0] JUMP[BLTB] $
16 1222                                         D[MEM] ROT[LLOAD-ROT] DEST[LLOAD] $     ;Loop counter is shifted
16 1222                                  over
16 1223
16 1224                                 BLT7:   D[AR] MASK[18.] ROT[18.] DEST[Q MA AR] $ ; Fetch first word
16 1225                                 BLT5:   DFRQ D[PC] ALU[D-1] DEST[PC] JUMP[BLTZ] $
16 1226                                                 ;Back up dest. addrs and enter loop.
16 1227                                 ; Main loop
16 1228                                 BLT3:   DFRQ $
16 1229                                          ;Fetch next word.
16 1230                                 BLTZ:   PC+1 MA-FROM-PC D[MEM] DEST[MEMSTO] COND[INTRPT] JUMP[BLT6] $
16 1231                                                 ;Inc. the dest. addr, start the store, check for intrpts.
16 1231
16 1232                                         ALU[Q+1] DEST[Q AR MA] LOOP[BLT3] $
16 1233                                                 ;Kick source ptr,loop
16 1234                                 BLT9:   D[PC] MASK[18.] DEST[O-AC PC MA] $
16 1235                                                 ;Loop count exhausted.  Prepare to exit.
16 1236                                         D[IR] MASK[18.] ALU[AC-D] COND[-OBUS<0] JUMP[BLT98] $
16 1237                                                 ; Jump if done
16 1238                                         ALU[AC+1] DEST[AC] $
16 1239                                                 ;Not done.  Prepare to xfer some more words.
16 1240                                         D[IR] ALU[D-AC] DEST[AR] $      ; Get count
16 1241                                         D[AR] ROT[LLOAD-ROT] DEST[LLOAD] $
16 1242                                         D[PC] DEST[O-AC PC]  $
16 1243                                         ALU[Q] DEST[MA] JUMP[BLT5] $
16 1244                                 BLT98:  MA-NEOI $
16 1245                                 ; Interrupt occured
16 1246                                 BLT6:   ALU[Q+1] DEST[Q AR] LOOP[BLT8] $
16 1247                                         JUMP[BLT9] $
16 1248                                 BLT8:   ALU[AC-1] DEST[AC] SPEC[PC+1] NORM $
16 1249                                         D[PC] MASK[18.] DEST[O-AC PC MA] SHORT $
16 1250                                         MA-NEOI D[AR] ROT[18.] MASK[LEFT] ALU[DORAC] DEST[AC] $
16 1251
16 1252                                 .REPEAT 1 - WAITS [     ;You'd better put in map fault code before
16 1252                                  enabling
16 1253                                                         ;this for WAITS, which must have RE-XCT on page
16 1253                                  traps.
16 1254                                 ; Ptr has form X,,X+1
16 1255                                 BLTZ:   D[IR] MASK[18.] ALU[D-Q] DEST[HOLD] $
16 1256                                         D[MEM] ROT[LLOAD-ROT] DEST[LLOAD] $     ;Loop counter is shifted
16 1256                                  over
16 1257                                         D[AR] ROT[18.] MASK[18.] DEST[MA] $     ;Read 1st word
16 1258                                         DFRQ $
16 1259                                         D[MEM] DEST[Q] $
16 1260                                         MA+MA+1 ALU[Q] DEST[MEMSTO] LOOP[.] $
16 1261                                         JUMP[BLT9] $
16 1262                                                 ];REPEAT 1 - WAITS
16 1263
16 1264                                 ; Bassackwards BLT
16 1265                                 BLTB:   ALU[0] ROT[LLOAD-ROT] DEST[LLOAD] JUMP[BLT7] $  ;End-precedes
16 1265                                  dest
16 1266
16 1267                                 ];.REPEAT OLDBLT
16 1268
```

```
17 1269                                 ;BLT0 BLT0A BLT1 BLT1A BLT1B BLT1C BLT1LP BLT1CK BLT1CZ BLT1IN BLT1I2
17 1269                                  BLT1Z BLTZLP BLTZ MA
17 1270
17 1271                                 .REPEAT 1 - OLDBLT [
17 1272
17 1273                                 .DEFINE +++[] []                ;For SRCCOM
17 1274
17 1275                                 +++;Revised BLT, still 3 cycles but doesn't use PC and doesn't require
17 1276                                 +++;special map trap code (for memory-to-memory cases).
17 1277                                 +++;
17 1278                                 +++;Mem-to-mem loop could probably be cut to two cycles, but this code
                                         was
17 1279                                 +++;written to be minimize bugs rather that execute time.
17 1280                                 +++;
17 1281                                 +++;Old AC codes is unchanged, and still has special map code.
17 1281                                  TVR/Sep82
17 1282
17 1283                                   .opcode[251]
17 1283                                  ;BLT
17 1284                                 BLT0:   D[MASK 14.] ROT[4] ALU[D&AC] COND[-OBUS=0] LBJUMP[BLT0A] $
17 1284
17 1285                                         D[MASK 14.] ROT[4] ALU[D&AC] COND[-OBUS=0] LBJUMP[BLT0A] $
17 1286                                                 ; Test for dest<20
17 1287
17 1288                                   .reloc
17 1289                                   .pair
17 1290                                 BLT0A:  JUMP[BLTA1] $           ;dest<20
17 1291                                         D[MASK 14.] ROT[18. + 4] ALU[D&AC] COND[OBUS=0] JUMP[BLTA2] $
17 1292                                                 ; Jump if src<20
17 1293                                 ;       \ /
17 1294                                 +++;Revised BLT code - TVR/Sep82
17 1295                                 +++;Many comments as this is a place where many bugs could lurk!
17 1296                                 BLT1:   ALU[AC] DEST[AR] SHORT $
17 1297                                                 ;Copy BLT point to where we can use masker/shifter
17 1298                                         D[IR] MASK[18.] DEST[Q] NORM $
17 1299                                                 ;Address of termination of BLT
17 1300                                         D[AR] MASK[18.] ALU[D-Q-1] DEST[Q HOLD] COND[OBUS<0] JUMP[BLT1B]
                                         $
17 1300                                                 ;Calculate number of words to transfer
17 1301                                                 ;Check for wraparound
17 1302                                 +++;;;  D[AR] DEST[Q] NORM $
17 1303                                 +++;;;          ;Destination of BLT will pass into ACs.
17 1304                                         ALU[-1] DEST[Q HOLD] SHORT $
17 1305                                                 ;"If AC(R) >= E, the BLT moves one word to location
17 1306                                  AR(L)"[sic]
17 1306                                 ;       \ /
17 1307                                 +++;AR: source,,dest
17 1308                                 +++;Q,MEM: -size of BLT
17 1309                                 BLT1B:  D[MASK 16.] ALU[D+Q] COND[-OBUS<0] JUMP[BLT1C] $
17 1310                                                 ;Check size of BLT against loop register
17 1311                                         D[MASK 16.] ALU[NOTD] DEST[Q HOLD] NORM $
17 1312                                                 ;Maximum loop count
17 1313                                 ;       \ /
17 1314                                 BLT1C:  D[AR] ROT[18.] MASK[18.] DEST[Q] NORM $
17 1315                                                 ;Setup source pointer
17 1316                                         D[MEM] MASK[18.] ALU[Q-D-1] COND[OBUS<0] JUMP[BLT1D] NORM $
17 1317                                                 ;Jump if source pointer won't run into ACs
17 1318                                         D[AR] ROT[18.] DEST[Q] NORM $
17 1319                                                 ;Source pointer runs into ACs.
17 1320                                 +++;    \ /
17 1321                                 +++;BLT will pass thru ACs on fetch store.  Q contains address
17 1322                                 +++;of half of transfer which will do this.  We use -1,,N
17 1323                                 +++;as a negative word count so that we will transfer 777777-N+1 words.
17 1324                                         D[MASK 18.] ROT[18.] ALU[DORQ] DEST[Q HOLD] JUMP[BLT1C] NORM $
17 1325                                                 ;Sign extended
17 1326                                                 ;Go back and setup Q again.
17 1327                                 +++;    ---
17 1328
17 1329                                 +++;AR: source,,dest
17 1330                                 +++;MEM: -size of BLT
17 1331                                 +++;Q: source
17 1332                                 BLT1D:
17 1333                                 ;+++    D[AR] MASK[18.] ALU[D-Q-1] COND[OBUS=0] JUMP[BLT1Z] $
17 1334                                                 ;Check for core-zeroing behavior.
17 1335                                         D[MEM] ROT[LLOAD-ROT] ALU[NOTD] DEST[LLOAD] $
17 1336                                                 ;Load loop counter.
17 1337                                         ALU[Q] DEST[AR MA] NORM $
17 1338                                 +++             ;Setup for first fetch
17 1339                                 +++             ;Perhaps this instruction could go away we can avoid
17 1340                                 +++             ;  DF-FROM-AR at .+2
17 1341                                 ;       \ /
17 1342                                 +++;AC: source,,dest
17 1343                                 +++;AR,Q: source
17 1344                                 +++;Loop counter:  Size of BLT.
17 1345                                 +++;We duplicate the first iteration of the loop rather than subtracting
17 1346
17 1346                                 +++;1,,1 from AC before entering it.
17 1347                                         DFRQ ALU[AC] DEST[AC MA] $
17 1348                                                 ;Do first fetch and setup destination
17 1349                                 ;Added RE-XCT when DF-FROM-XX conversion was made.  TVR/Apr83
17 1350                                         D[DF-FROM-AR MEM] RE-XCT DEST[MEMSTO] COND[INTRPT] JUMP[BLT1IN]
17 1351                                  NORM $
17 1351                                 +++             ;Finish fetch, start store.
17 1352                                 +++             ;We could leave out the interrupt test and guarantee that
17 1353
17 1353                                 +++             ;  at least would be moved between interrupts.  However,
17 1354                                  this
17 1354                                 +++             ;  would increase interrupt latency.
17 1355                                         TRP-CTL[RE-XCT] ALU[Q+1] DEST[Q MA AR] LOOP[BLT1LP] $
17 1356                                                 ;Advance WCMA.  Setup AR to be used as MA.
17 1357                                                 ;We normally expect to jump
17 1358                                         JUMP[BLT1CK] NORM $
17 1359                                                 ;Gee, a one word BLT!
17 1360                                 +++;    ---
17 1361
17 1362                                 +++;This is the main loop of the main BLT
17 1363                                 +++;AC:   source-1,,destination-1
17 1364                                 +++;Q,AR: source
17 1365                                 +++;Loop counter: Number of words to move.
17 1366                                 BLT1LP: DFRQ D[1,,1] ALU[AC+D] DEST[AC MA] $
17 1367
```

```
17 1372                                                 +++             ;NOTE:  The only other instruction that uses DF-FROM-AR
17 1372                                                  is
17 1373                                                 +++             ;       the POP instruction (unless there's one in F4JSYS
17 1373
17 1374                                                 +++             ;       or F4AUG).      TVR/Sep82
17 1375                                                         TRP-CTL[RE-XCT] ALU[Q+1] DEST[Q MA AR] LOOP[BLT1LP] $
17 1376                                                                 ;Advance WCMA.  Setup AR to be used as MA.
17 1377                                                 ;       \ /
17 1378                                                 BLT1CK: D[MASK 18.] ALU[D&AC] DEST[Q] COND[-OBUS=0] JUMP[BLT1C2] $
17 1379                                                                 ;Now, check for termination, watching for overflow
17 1380                                                         D[CONST 1] ROT[18.] ALU[AC-D] DEST[AC] NORM $
17 1381                                                                 ;Fixup after BLT pointer overflow
17 1382                                                 ;       \ /
17 1383                                                 BLT1C2: D[IR] MASK[18.] ALU[Q-D] COND[OBUS<0] JUMP[BLT0] $
17 1384                                                                 ;"If AC(R) >= E", we're done
17 1385                                                 ;       \ /
17 1386                                                         jump[main] $    ;for now...
17 1387                                                 +++;;;  MA-NEOI $
17 1388                                                                 ;BLT finished
17 1389                                                 ;       ---
17 1390
17 1391                                                 +++;This BLT got interrupted.
17 1392                                                 BLT1IN: TRP-CTL[RE-XCT] D[MASK 18.] ALU[D&AC] DEST[Q]
17 1393                                                                         COND[-MA<20] JUMP[BLT1I2] NORM $
17 1394                                                                 ;Now, check for termination, watching for otherflow
17 1395                                                         D[CONST 1] ROT[18.] ALU[AC-D] DEST[AC] NORM $
17 1396                                                                 ;Fixup after BLT pointer overflow
17 1397                                                 ;       \ /
17 1398                                                 BLT1I2: D[IR] MASK[18.] ALU[D#Q] COND[OBUS=0] JUMP[MAIN] $
17 1399                                                                 ;If we matched, the BLT finished at the same time
17 1400                                                                 ;as we got an interrupt!
17 1401                                                         D[PC] ALU[D-1] DEST[PC] JUMP[MAIN] $
17 1402                                                                 ;Backup PC and try again.
17 1403
17 1404                                                 ;We get to duplicate some stuff when zeroing
17 1405                                                 BLT1Z:  D[IR] MASK[18.] DEST[AR] NORM $
17 1406                                                                 ;Save IR in AR for end-check
17 1407                                                         D[MEM] ROT[LLOAD-ROT] DEST[LLOAD] $
17 1408                                                                 ;Load loop counter.
17 1409                                                         DFRQ ALU[AC] DEST[AC MA] $
17 1410                                                                 ;Fetch the first word, that's all we'll ever use!
17 1411                                                         TRP-CTL[RE-XCT] D[MEM] DEST[Q] $
17 1412                                                                 ;I wonder if this is really necessary.
17 1413                                                 ;       \ /
17 1414                                                 +++;This loop could be reduced to one instruction, but then we'll need
17 1415                                                 +++;fixup code.  Let's make this work first!
17 1416                                                 BLTZLP: ALU[Q] DEST[MEMSTO] COND[INTRPT] JUMP[BLT1IN] NORM $
17 1417                                                                 ;Finish fetch, start store.
17 1418                                                         TRP-CTL[RE-XCT] D[1,,1] ALU[D+AC] DEST[AC MA] LOOP[BLTZLP] $
17 1419                                                                 ;Advance BLT pointer, and check for count exhausted
17 1420                                                         JUMP[BLT1CK] NORM $
17 1421                                                                 ;Now, check for termination, watching for otherflow
17 1422
17 1423                                                 .REPEAT 0 [     ;You'd better put in map fault code before enabling
17 1424                                                                 ;this for WAITS, which must have RE-XCT on page traps.
17 1425                                                 ; Ptr has form X,,X+1
17 1426                                                 BLTZ:   D[IR] MASK[18.] ALU[D-Q] DEST[HOLD] $
17 1427                                                         D[MEM] ROT[LLOAD-ROT] DEST[LLOAD] $     ;Loop counter is shifted
 17 1427                                                 over
17 1428                                                         D[AR] ROT[18.] MASK[18.] DEST[MA] $     ;Read 1st word
17 1429                                                         DFRQ $
17 1430                                                         D[MEM] DEST[Q] $
17 1431                                                         MA←MA+1 ALU[Q] DEST[MEMSTO] LOOP[.] $
17 1432                                                         JUMP[BLT9] $
17 1433                                                                 ];REPEAT 0
17m1434                                                 ];[
17m1434
17m1434                                                 .DEFINE +++[] []                ;For SRCCOM
17m1434
17m1434                                                 +++; []Revised BLT, still 3 cycles but doesn't use PC and doesn't require
17m1434
17m1434                                                 +++; []special map trap code (for memory-to-memory cases).
17m1434                                                 +++; []
17m1434                                                 +++; []Mem-to-mem loop could probably be cut to two cycles, but this code
 17m1434                                                 was
17m1434                                                 +++; []written to be minimize bugs rather that execute time.
17m1434                                                 +++; []
17m1434                                                 +++; []Old AC codes is unchanged, and still has special map code.
 17m1434                                                 TVR/Sep82
17m1434
17m1434                                                   .opcode[251]   [xlist
17m1434                                                    list  ]                                                     ;BLT
17m1434 06522 010541000000000001035621247314560000     BLT0:   D[MASK 14.] ROT[4] ALU[D&AC] COND[-OBUS=0] LBJUMP[BLT0A] $
17m1434
17m1434 06523 010541000000000001035621247314560000             D[MASK 14.] ROT[4] ALU[D&AC] COND[-OBUS=0] LBJUMP[BLT0A] $
17m1434                                                                 ; Test for dest<Z0
17m1434
17m1434                                                   .reloc
17m1434                                                  [.USE[NORMAL]
17m1434                                                  [ xlist
17m1435                                                    list ]                                  ]  .pair
17m1435                                                  [:. \ Z + .
17m1436 05120 010731170000000014161620Z5431456000        ]BLT0A:       JUMP[BLTA1] $            ;dest<Z0
17m1436 05121 010541002000000005435620247314560000             D[MASK 14.] ROT[18. + 4] ALU[D&AC] COND[OBUS=0] JUMP[BLTAZ] $
17m1436                                                                 ; Jump if src<Z0
17m1436                                                 ;       \ /
17m1436                                                 +++; []Revised BLT code - TVR/Sep82
17m1436                                                 +++; []Many comments as this is a place where many bugs could lurk!
17m1436 05122 010331370000060554161623654374160000     BLT1:   ALU[AC] DEST[AR] SHORT $
17m1436                                                                 ;Copy BLT point to where we can use masker/shifter
17m1436 05123 010730170000060554045623657714160000             D[IR] MASK[18.] DEST[Q] NORM $
17m1436                                                                 ;Address of termination of BLT
17m1436 05124 010620004000000014045500254314560000             D[AR] MASK[18.] ALU[D-Q-1] DEST[Q HOLD] COND[OBUS<0] JUMP[BLT1B]
 17m1436                                                 $
17m1436                                                                 ;Calculate number of words to transfer
17m1436                                                                 ;Check for wraparound
17m1436                                                 +++; [];;       D[AR] DEST[Q] NORM $
17m1436                                                 +++; [];;               ;Destination of BLT will pass into ACs.
17m1436 05125 010720170000060554001503655774160000             ALU[-1] DEST[Q HOLD] SHORT $
17m1436                                                                 ;"If AC(R) >= E, the BLT moves one word to location
```

```
17m1436
17m1436 05127 010770170000060554041503647314160000             ;Check size of BLT against loop register
                                                        D[MASK 16.] ALU[NOTD] DEST[Q HOLD] NORM $
17m1436                                                         ;Maximum loop count
17m1436                                                 ;       \ /
17m1436 05130 010730170000060544445623654314160000     BLT1C:  D[AR] ROT[18.] MASK[18.] DEST[Q] NORM $
17m1436                                                         ;Setup source pointer
17m1436 05131 010611004000000014045620254715560000             D[MEM] MASK[18.] ALU[Q-D-1] COND[OBUS<0] JUMP[BLT1D] NORM $
17m1436                                                         ;Jump if source pointer won't run into ACs
17m1436 05132 010730170000060544561623654314160000             D[AR] ROT[18.] DEST[Q] NORM $
17m1436                                                         ;Source pointer runs into ACs.
17m1436                                                 +++; [] \ /
17m1436                                                 +++; []BLT will pass thru ACs on fetch store.  Q contains address
17m1436                                                 +++; []of half of transfer which will do this.  We use -1,,N
17m1436                                                 +++; []as a negative word count so that we will transfer 777777-N+1
17m1436                                                  words.
17m1436 05133 010630170000012Z60444550024731456000             D[MASK 18.] ROT[18.] ALU[DORQ] DEST[Q HOLD] JUMP[BLT1C] NORM $
17m1436                                                         ;Sign extended
17m1436                                                         ;Go back and setup Q again.
17m1436                                                 +++; [] ---
17m1436
17m1436                                                 +++; []AR: source,,dest
17m1436                                                 +++; []MEM: -size of BLT
17m1436                                                 +++; []Q: source
17m1436                                                 BLT1D:
17m1436                                                 ;+++    D[AR] MASK[18.] ALU[D-Q-1] COND[OBUS=0] JUMP[BLT1Z] $
17m1436                                                         ;Check for core-zeroing behavior.
17m1436 05134 010771170010060540160003654715170000             D[MEM] ROT[LLOAD-ROT] ALU[NOTD] DEST[LLOAD] $
17m1436                                                         ;Load loop counter.
17m1436 05135 710231370000060554161623654314160000             ALU[Q] DEST[AR MA] NORM $
17m1436                                                 +++      []     ;Setup for first fetch
17m1436                                                 +++      []     ;Perhaps this instruction could go away we can avoid
17m1436                                                 +++      []     ;  DF-FROM-AR at .+2
17m1436                                                 ;       \ /
17m1436                                                 +++; []AC: source,,dest
17m1436                                                 +++; []AR,Q: source
17m1436                                                 +++; []Loop counter:  Size of BLT.
17m1436                                                 +++; []We duplicate the first iteration of the loop rather than
17m1436                                                  subtracting
17m1436                                                 +++; []1,,1 from AC before entering it.
17m1436 05136 710333170000060554161623654312160000             DFRQ ALU[AC] DEST[AC MA] $
17m1436                                                         ;Do first fetch and setup destination
17m1436                                                 ;Added RE-XCT when DF-FROM-XX conversion was made.  TVR/Apr83
17m1436 05137 010731104700000001416156025471556000             D[DF-FROM-AR MEM] RE-XCT DEST[MEMSTO] COND[INTRPT] JUMP[BLT1IN]
17m1436                                                          NORM $
17m1436                                                 +++      []     ;Finish fetch, start store.
17m1436                                                 +++      []     ;We could leave out the interrupt test and guarantee that
17m1436
17m1436                                                 +++      []     ;  at least would be moved between interrupts.  However,
17m1436                                                  this
17m1436                                                 +++      []     ;  would increase interrupt latency.
17m1436 05140 711200270400000001416162025431556000             TRP-CTL[RE-XCT] ALU[Q+1] DEST[Q MA AR] LOOP[BLT1LP] $
17m1436                                                         ;Advance WCMA.  Setup AR to be used as MA.
17m1436                                                         ;We normally expect to jump
17m1436 05141 010731170006000001416162025431456000             JUMP[BLT1CK] NORM $
17m1436                                                         ;Gee, a one word BLT!
17m1436                                                 +++; [] ---
17m1436
17m1436                                                 +++; []This is the main loop of the main BLT
17m1436                                                 +++; []AC:   source-1,,destination-1
17m1436                                                 +++; []Q,AR: source
17m1436                                                 +++; []Loop counter: Number of words to move.
17m1436 05142 710503170037060554003623655571216000     BLT1LP: DFRQ D[1,,1] ALU[AC+D] DEST[AC MA] $
17m1436                                                 +++      []     ;Advance both source and destination
17m1436                                                 ;Added RE-XCT when DF-FROM-XX conversion was made.  TVR/Apr83
17m1436 05143 010731104700000001416156025471556000             D[DF-FROM-AR MEM] RE-XCT DEST[MEMSTO] COND[INTRPT] JUMP[BLT1IN]
17m1436                                                          NORM $
17m1436                                                         ;Finish fetch, start store and check for interrupt.
17m1436                                                 +++      []     ;NOTE:  The only other instruction that uses DF-FROM-AR
17m1436                                                  is
17m1436                                                 +++      []     ;       the POP instruction (unless there's one in F4JSYS
17m1436
17m1436                                                 +++      []     ;       or F4AUG).      TVR/Sep82
17m1436 05144 711200270400123054161620254315560000             TRP-CTL[RE-XCT] ALU[Q+1] DEST[Q MA AR] LOOP[BLT1LP] $
17m1436                                                         ;Advance WCMA.  Setup AR to be used as MA.
17m1436                                                 ;       \ /
17m1436 05145 010540000000000014045620247314560000     BLT1CK: D[MASK 18.] ALU[D&AC] DEST[Q] COND[-OBUS=0] JUMP[BLT1CZ] $
17m1436                                                         ;Now, check for termination, watching for overflow
17m1436 05146 011513170000060544403623655571416000             D[CONST 1] ROT[18.] ALU[AC-D] DEST[AC] NORM $
17m1436                                                         ;Fixup after BLT pointer overflow
17m1436                                                 ;       \ /
17m1436 05147 011611004000152454045620257714560000     BLT1C2: D[IR] MASK[18.] ALU[Q-D] COND[OBUS<0] JUMP[BLT0] $
17m1436                                                         ;"If AC(R) >= E", we're done
17m1436                                                 ;       \ /
17m1436 05150 010731170000227114161620254314560000             jump[main] $    ;for now...
17m1436                                                 +++; [];;       MA-NEOI $
17m1436                                                         ;BLT finished
17m1436                                                 ;       ---
17m1436
17m1436                                                 +++; []This BLT got interrupted.
17m1436                                                 BLT1IN: TRP-CTL[RE-XCT] D[MASK 18.] ALU[D&AC] DEST[Q]
17m1436                                                                         COND[-MA<20] JUMP[BLT1IZ] NORM $
17m1436 05151 010540112400000014045620247315560000             ;Now, check for termination, watching for otherflow
17m1436 05152 011513170000060544403623655571416000     D[CONST 1] ROT[18.] ALU[AC-D] DEST[AC] NORM $
17m1436                                                         ;Fixup after BLT pointer overflow
17m1436                                                 ;       \ /
17m1436 05153 010661002000227114045620257714560000     BLT1IZ: D[IR] MASK[18.] ALU[D#Q] COND[OBUS=0] JUMP[MAIN] $
17m1436                                                         ;If we matched, the BLT finished at the same time
17m1436                                                         ;as we got an interrupt!
17m1436 05154 610721170000227114161460256314560000     D[PC] ALU[D-1] DEST[PC] JUMP[MAIN] $
17m1436                                                         ;Backup PC and try again.
17m1436
17m1436                                                 ;We get to duplicate some stuff when zeroing
17m1436 05155 010731370000060554045623657714160000     BLT1Z:  D[IR] MASK[18.] DEST[AR] NORM $
17m1436                                                         ;Save IR in AR for end-check
17m1436 05156 010731170010060540160003654715170000     D[MEM] ROT[LLOAD-ROT] DEST[LLOAD] $
17m1436                                                         ;Load loop counter.
17m1436 05157 710333170000060554161623654312160000     DFRQ ALU[AC] DEST[AC MA] $
17m1436                                                         ;Fetch the first word, that's all we'll ever use!
17m1436 05160 010730170400060554161623654715160000     TRP-CTL[RE-XCT] D[MEM] DEST[Q] $
17m1436                                                         ;I wonder if this is really necessary.
```

```
17m1436                                                         ;Finish fetch, start store.
17m1436 05162 710503070437123434003620255715560000             TRP-CTL[RE-XCT] D[1,,1] ALU[D+AC] DEST[AC MA] LOOP[BLTZLP] $
17m1436                                                         ;Advance BLT pointer, and check for count exhausted
17m1436 05163 010731170000123134161620254314560000             JUMP[BLT1CK] NORM $
17m1436                                                         ;Now, check for termination, watching for otherflow
17m1436
17m1436                                                 .REPEAT 0 [     ;You'd better put in map fault code before enabling
17m1436                                                         ;this for WAITS, which must have RE-XCT on page traps.
17m1436                                                 ; Ptr has form X,,X+1
17m1436                                                 BLTZ:   D[IR] MASK[18.] ALU[D-Q] DEST[HOLD] $
17m1436                                                         D[MEM] ROT[LLOAD-ROT] DEST[LLOAD] $     ;Loop counter is shifted
17m1436                                                  over
17m1436                                                         D[AR] ROT[18.] MASK[18.] DEST[MA] $     ;Read 1st word
17m1436                                                         DFRQ $
17m1436                                                         D[MEM] DEST[Q] $
17m1436                                                         MA+MA+1 ALU[Q] DEST[MEMSTO] LOOP[.] $
17m1436                                                         JUMP[BLT9] $
17m1436                                                         ];REPEAT 0
17 1434                                                 ].REPEAT 1 - OLDBLT
17 1435
```

```
SLOE  March 23, 1984 21:06:15  file DSK:F4INST.SLO  --  of -- F41NNF

18 1436                                                 ;BLTA1 BLTAL3 BLTFX2 BLTAL2 BLTXIT BLTA2 BLTA2A BLTA2B BLTL1 BLTA4 BLTA10
18 1436                                                  BLTL2 BLTL2A BLTS9 BLTCHK
18 1437                                                 .repeat 0[]     ;Help SRCCOM not match too soon.
18 1438                                                 ; AC cases
18 1439                                                 ; Dest is AC
18 1440 05164 010541002000000005435620247314560000     BLTA1:  D[MASK 16] ROT[26] ALU[D&AC] COND[OBUS=0] JUMP[BLTA10] $
18 1441                                                                 ; Jump if src is AC
18 1442 05165 010731000000000010035620257714560000             D[IR] ROT[40] MASK[16] COND[-OBUS=0] JUMP[BLTA10] $
18 1443                                                                 ;Jump if final dest is not AC
18 1444 05166 010331370000060554161623654314160000             ALU[AC] DEST[AR] $      ; Get dest
18 1445 05167 010731170000060540160123654314170000             D[AR] ROT[AMEM-P-ROT] DEST[AMEM-P] $
18 1446 05170 010730170000060554045623657774160000             D[IR] MASK[18.] DEST[Q] SHORT $
18 1447 05171 011610004000000001404550225431456000             D[AR] MASK[18.] ALU[Q-D] DEST[Q HOLD] COND[OBUS<0] PUSHJ[BLTA4] $
18 1447                                                                 ;Adjust if cnt neg.
18 1448                                                         D[MEM] ROT[LLOAD-ROT] DEST[LLOAD] $     ;Loop counter is shifted
18 1449 05172 010731170010060540160003654715170000      over
18 1449                                                 .repeat oldblt [
18 1450                                                         D[AR] ROT[18.] MASK[18.] DEST[MA] NORM JUMP[BLTAL3] $
18 1451                                                          ;Get src
18 1452                                                 ];.repeat oldblt
18 1453                                                 .repeat 1 - oldblt [    ;GRUNT avoidance
18 1454                                                         D[AR] ROT[18.] MASK[18.] DEST[MA Q] NORM JUMP[BLTAL3] $
18 1455                                                          ;Get src, load MA for fetching and Q for remembering in case of
18 1456                                                  traps.
18 1456                                                 ];[     ;GRUNT avoidance
18m1457                                                         D[AR] ROT[18.] MASK[18.] DEST[MA Q] NORM JUMP[BLTAL3] $
18m1457 05173 710730170000000000444562025431456000              ;Get src, load MA for fetching and Q for remembering in case of
18m1457                                                  traps.
18m1457                                                 ].repeat 1 - oldblt
18 1457                                                   .odd
18m1458                                                  [:. + 1 - (. \ 2)
18m1458                                                    ]BLTAL3:     DFRQ MA+MA+1 PUSHJ[BLTAL2] $
18 1459 05175 110731170000000001416162225431256000             D[MA] ALU[D-1] DEST[MA] NORM JUMP[ECC-PROCEED] $
18 1460 05176 710721170000022353416162025671456000               ;ECC Trap. Restore failing MA.
18 1461                                                 .repeat oldblt [
18 1462                                                         D[MA] ALU[D-1] DEST[MA]  NORM PUSHJ[MAP-RET] $
18 1463                                                         NORM JUMP[BLTAL3] $
18 1464                                                 BLTAL2: DFRQ TRP-CTL[TRAP-FIX] ACSEL[D,AMEM-P] D[MEM] DEST[B]
18 1465                                                                 AMEM-P-INCR MA+MA+1 LOOP[BLTAL2] NORM $
18 1466                                                 ];.repeat oldblt
18 1467                                                 .repeat 1 - oldblt [
18 1468                                                         D[MA] ALU[D-1] DEST[MA] $
18 1469                                                           ;Map Trap. Restore failing MA for map code.
18 1470                                                         D[MA] ALU[D-Q] DEST[AR] JUMP[BLTFX2] $
18 1471                                                           ;Subtract initial MA value (preserved in Q) from failing
18 1472                                                  value...
18 1472                                                           ; this gives number of words moved. Note: LH is garbage.
18 1473                                                 ;       \ /
18 1474                                                 ;Update BLT point in AC by offset in AR, then finish map trap.  Note: MA
18 1475
18 1475                                                 ;should already be valid when we get here.
18 1476                                                 BLTFX2:  D[AR] ROT[18.] MASK[LEFT] ALU[D+AC] DEST[AC] $
18 1477                                                                 ;Advance source.
18 1478                                                         D[AR] MASK[18.] ALU[D+AC] DEST[D-AC AR] $
18 1479                                                                 ;Advance destination.  Watch out for overflow to left
18 1480                                                  half
18 1480                                                         D[AR] MASK[LEFT] ALU[AC-D] DEST[AR] $
18 1481                                                                 ;LH=1 if overflow happened.
18 1482                                                         D[AR] MASK[LEFT] ALU[AC-D] DEST[AR] PUSHJ[MAP-RET] $
18 1483                                                                 ;Finally, fixup overflow.  Sigh... the price of honesty.
18 1484
18 1484                                                          jump[.] $
18 1485                                                                 ;Shouldn't get here if RE-XCT is set!
18 1486
18 1487                                                 BLTAL2: DFRQ TRP-CTL[TRAP-FIX RE-XCT] ACSEL[D,AMEM-P] D[MEM] DEST[B]
18 1488                                                                AMEM-P-INCR MA+MA+1 LOOP[BLTAL2] NORM $
18 1489                                                 ];[
18m1490                                                         D[MA] ALU[D-1] DEST[MA] $
18m1490 05177 710721170000060554161623656714160000               ;Map Trap. Restore failing MA for map code.
18m1490                                                         D[MA] ALU[D-Q] DEST[AR] JUMP[BLTFX2] $
18m1490 05200 011621370000000001416162025671456000               ;Subtract initial MA value (preserved in Q) from failing
18m1490                                                  value...
18m1490                                                          ; this gives number of words moved. Note: LH is garbage.
18m1490                                                 ;       \ /
18m1490                                                 ;Update BLT point in AC by offset in AR, then finish map trap.  Note: MA
18m1490
18m1490                                                 ;should already be valid when we get here.
18m1490 05201 010503170000060544541623654314160000     BLTFX2:  D[AR] ROT[18.] MASK[LEFT] ALU[D+AC] DEST[AC] $
18m1490                                                                 ;Advance source.
18m1490 05202 010502370000060554045623654314160000             D[AR] MASK[18.] ALU[D+AC] DEST[D-AC AR] $
18m1490                                                                 ;Advance destination.  Watch out for overflow to left
18m1490                                                  half
18m1490 05203 011511370000060554141623654314160000             D[AR] MASK[LEFT] ALU[AC-D] DEST[AR] $
18m1490                                                                 ;LH=1 if overflow happened.
18m1490 05204 011511370000212374141622254314560000             D[AR] MASK[LEFT] ALU[AC-D] DEST[AR] PUSHJ[MAP-RET] $
18m1490                                                                 ;Finally, fixup overflow.  Sigh... the price of honesty.
18m1490
18m1490                                                          jump[.] $
18m1490 05205 010731170000124134161620254314560000                     ;Shouldn't get here if RE-XCT is set!
18m1490
18m1490                                                 BLTAL2: DFRQ TRP-CTL[TRAP-FIX RE-XCT] ACSEL[D,AMEM-P] D[MEM] DEST[B]
18m1490                                                                AMEM-P-INCR MA+MA+1 LOOP[BLTAL2] NORM $
18m1490 05206 110733070507124154161620314713560000     ].repeat 1 - oldblt
18 1490                                                 BLTXIT: NEOI $
18 1491 05207 540731170000060554161623254204160000
18 1492
18 1493                                                 ; Src is AC, dest is mem
18 1494 05210 710311370000060554161623654314160000     BLTA2:  ALU[AC-1] DEST[AR MA] $
18 1495                                                                 ;Get DEST-1 in MA, SRC in AR00:17
18 1496 05211 010731370000060544561623654314160000             D[AR] ROT[18.] DEST[AR] $
18 1497 05212 010731170000060540160123654314170000             D[AR] ROT[AMEM-P-ROT] DEST[AMEM-P] $
18 1498                                                                 ;Get SRC adr into AMEM PDL for use in fetching ACs.
18 1499 05213 010730170000060554045623657714160000             D[IR] MASK[18.] DEST[Q] $
18 1500 05214 010610204000000014045622256714560000             D[MA] MASK[18.] ALU[Q-D-1] DEST[Q AR] COND[OBUS<0] PUSHJ[BLTA4] $
18 1500
18 1501                                                                 ;Compute length of transfer, jump if neg
18 1502                                                 .repeat oldblt [
18 1503                                                         D[AR] ROT[LLOAD-ROT] DEST[LLOAD] $
18 1504                                                           ;Loop counter is shifted over
18 1505                                                         D[AR] ROT[18.] MASK[18.] ALU[D+Q] DEST[Q] $
```

```
18 1511                                                 ];.repeat oldblt
18 1512                                                 .repeat 1 - oldblt [
18 1513                                                         D[AR] ROT[LLOAD-ROT] DEST[LLOAD] JUMP[BLTA2A] $
18 1514                                                                 ;Loop counter is shifted over
18 1515                                                         .ODD
18 1516                                                 BLTA2A: D[AR] ROT[18.] MASK[18.] ALU[D+Q] DEST[Q] PUSHJ[BLTA2B] $
18 1517                                                                 ;Calc. final source addrs.
18 1518                                                                 ;Setup stack for map trap
18 1519                                                          JUMP[.] $
18 1520                                                                 ;Shouldn't get ECC traps on writes!
18 1521                                                          D[MA] ALU[D-AC] DEST[AR] JUMP[BLTFX2] $
18 1522                                                                 ;Calculate number of words moved.  Note: LH is garbage
18 1523                                                                 ;Now, proceed with regular AC flavor of BLT.
18 1524
18 1525                                                 BLTA2B: D[MASK 30] ROT[4] ALU[D&Q] COND[-OBUS=0] JUMP[BLTA10] $
18 1526                                                           ;Jump if src leaves ACs and flush map trap address
18 1527                                                 BLTL1:  ACSEL[D,AMEM-P] TRP-CTL[TRAP-FIX RE-XCT] ALU[B] DEST[MEMSTO]
18 1528                                                                 AMEM-P-INCR MA+MA+1 LOOP[BLTL1] $
18 1529                                                 ];[
18m1529 05Z15 0107311700100000001600002543145700000            D[AR] ROT[LLOAD-ROT] DEST[LLOAD] JUMP[BLTA2A] $
18m1529                                                                 ;Loop counter is shifted over
18m1529                                                         .ODD
18m1529                                                  [:. + 1 - (. \ 2)
18m1530 05Z17 0106001700000000004445622543145600000              ]BLTA2A:      D[AR] ROT[18.] MASK[18.] ALU[D+Q] DEST[Q] PUSHJ[BLTA2B] $
18m1530
18m1530                                                                 ;Calc. final source addrs.
18m1530                                                                 ;Setup stack for map trap
18m1530 05Z20 0107311700001244141616Z02543145600000              JUMP[.] $
18m1530                                                                 ;Shouldn't get ECC traps on writes!
18m1530 05Z21 0115Z13700001240341616Z02567145600000              D[MA] ALU[D-AC] DEST[AR] JUMP[BLTFX2] $
18m1530                                                                 ;Calculate number of words moved.  Note: LH is garbage
18m1530                                                                 ;Now, proceed with regular AC flavor of BLT.
18m1530
18m1530 05Z22 0106410000000000010616Z02473145600000    BLTA2B: D[MASK 30] ROT[4] ALU[D&Q] COND[-OBUS=0] JUMP[BLTA10] $
18m1530                                                           ;Jump if src leaves ACs and flush map trap address
18m1530 05Z23 1103310705071244741615603143155600000    BLTL1:  ACSEL[D,AMEM-P] TRP-CTL[TRAP-FIX RE-XCT] ALU[B] DEST[MEMSTO]
18 1529                                                                 AMEM-P-INCR MA+MA+1 LOOP[BLTL1] $
18 1530 05Z24 0107311700001241741616Z02543145600000    ].repeat 1 - oldblt
18 1531                                                         NORM JUMP[BLTXIT] $
18 1532                                                                         ;
18 1533 05Z25 0102401700100605401600042543141700000    ; called on attempted back-BLT to or from ACs
18 1534                                                 BLTA4:  ALU[0] ROT[LLOAD-ROT] DEST[Q LLOAD] POPJ $
18 1535
18 1536                                                 ; Weird cases: both src and dest in ACs, or either crosses boundary
18 1537 05Z26 0103313700000605541616Z365431416000       ;++ No attempt has been made to make this efficient
18 1538 05Z27 7107311700000605444456Z365431416000       BLTA10: ALU[AC] DEST[AR] $
18 1539 05Z30 0103311700000605541615034543121600       BLTL2:  D[AR] ROT[18.] MASK[18.] DEST[MA] $     ;Read from AC or mem
18 1540                                                         DFRQ ALU[MEMAC] DEST[HOLD] $
18 1541                                                           ;Load MEM from either AC or real mem.
18 1542                                                 .repeat oldblt [
18 1543                                                         D[MEM] DEST[HOLD Q] $
18 1544                                                         D[AR] MASK[18.] DEST[MA] STRT-WRT $
18 1545                                                         COND[-MA<Z0] JUMP[. + 2] $
18 1546                                                         ALU[Q] DEST[MEMAC] $
18 1547                                                         D[AR] MASK[18.] DEST[Q] $
18 1548                                                         D[IR] MASK[18.] ALU[Q-D] COND[-OBUS<0] JUMP[BLT99] $
18 1549                                                         D[1,,1] DEST[Q] $
18 1549                                                         D[AR] ALU[D+Q] DEST[AR] JUMP[BLTL2] NORM $;*** TEST FOR INTERRUPT
18 1549                                                  HERE?
18 1550                                                 BLT99:  NEOI $
18 1551                                                 ];.repeat oldblt
18 1552                                                 .repeat 1 - oldblt [
18 1553                                                         TRP-CTL[RE-XCT] D[MEM] DEST[HOLD Q] $
18 1554                                                           ;Re-start inst. if we page fault here.
18 1555                                                         D[AR] MASK[18.] DEST[MA] STRT-WRT C500 COND[-EA<Z0]
18 1555                                                  LBJUMP[BLTL2A] $
18 1556                                                  .even
18 1557                                                 BLTL2A: ALU[Q] DEST[MEMAC] $
18 1558                                                           ;Here if storing an AC (no trap is possible).
18 1559                                                         TRP-CTL[RE-XCT] D[AR] MASK[18.] DEST[Q] $
18 1560                                                           ;Re-start inst. if we page fault here.
18 1561                                                         D[IR] MASK[18.] ALU[Q-D] COND[-OBUS<0] JUMP[BLT99] $
18 1562                                                         D[1,,1] ALU[D+AC] DEST[AC AR] COND[-INTRPT] JUMP[BLTL2] NORM $
18 1563                                                 +++             ;If no interrupt, move more words.  We should be glad if
18 1563
18 1564                                                 +++             ;we get an interrupt, as when we re-execute the BLT, we
18 1565                                                 +++             ;will probably continue execution in a faster loop.
18 1566                                                         D[PC] ALU[D-1] DEST[PC] JUMP[MAIN] $
18 1567
18 1568                                                 BLT99:  NEOI $
18m1569                                                 ];[
18m1569 05Z31 0107301704000605541615036547151600000            TRP-CTL[RE-XCT] D[MEM] DEST[HOLD Q] $
18m1569                                                           ;Re-start inst. if we page fault here.
18m1569 05Z32 7107311220000000014045641254214560000            D[AR] MASK[18.] DEST[MA] STRT-WRT C500 COND[-EA<Z0]
18m1569                                                  LBJUMP[BLTL2A] $
18m1569                                                  .even
18m1569                                                  [:. \ 2 + .
18m1570 05Z34 0102331700000605541616Z345431436000          ]BLTL2A:      ALU[Q] DEST[MEMAC] $
18m1570                                                           ;Here if storing an AC (no trap is possible).
18m1570 05Z35 0107301704000605540456Z365431516000               TRP-CTL[RE-XCT] D[AR] MASK[18.] DEST[Q] $
18m1570                                                           ;Re-start inst. if we page fault here.
18m1570 05Z36 0116110060000000014045620Z577145600000            D[IR] MASK[18.] ALU[Q-D] COND[-OBUS<0] JUMP[BLT99] $
18m1570 05Z37 0105033060371245740036Z025571456000               D[1,,1] ALU[D+AC] DEST[AC AR] COND[-INTRPT] JUMP[BLTL2] NORM $
18m1570                                                 +++      []     ;If no interrupt, move more words.  We should be glad if
18m1570
18m1570                                                 +++      []     ;we get an interrupt, as when we re-execute the BLT, we
18m1570                                                 +++      []     ;will probably continue execution in a faster loop.
18m1570 05Z40 6107211700002Z711416146025631456000               D[PC] ALU[D-1] DEST[PC] JUMP[MAIN] $
18m1570
18m1570 05Z41 5407311700000605541616Z32542041600000    BLT99:  NEOI $
18 1569                                                 ].repeat 1 - oldblt
18 1570
18 1571                                                 .repeat 0 [
18 1572                                                 ;Look for BLTST bug detected
18 1573                                                 BLTCHK: D[MASK 18.] ALU[D&AC] DEST[Q] NORM $
18 1574                                                                 ;Extract destination
18 1575                                                         D[IR] ALU[D-Q] DEST[Q] NORM $
18 1576                                                                 ;Calculate number of words to move
18 1577                                                         ALU[AC] DEST[AR] NORM $
18 1578                                                                 ;Move BLT pointer so we can get left half
```

```
19 1584
19 1585                                                 ;;; Note:  AOBJP and AOBJN behave KA-style (RH overflow adds to LH)
19 1586
19m1587                                                   .opcode[252]   [xlist
19 1587                                                    list  ]                                                    ;AOBJP
19 1588 06524 550503006037277554003623255512160000             D[1,,1] ALU[D+AC] DEST[AC] COND[-OBUS<0] JDSP C600 $
19 1589 06525 010731170000224334161620254314560000             JUMP [JMPAC] $
19 1590
19m1591                                                   .opcode[253]   [xlist
19 1591                                                    list  ]                                                    ;AOBJN
19 1592 06526 550503004037277554003623255512160000             D[1,,1] ALU[D+AC] DEST[AC] COND[OBUS<0] JDSP C600 $
19 1593 06527 010731170000224334161620254314560000             JUMP [JMPAC] $
19 1594                                                 .repeat 0 [
19 1595                                                   .opcode[254]
19 1595                                                  ;JRST
19 1596                                                         D[IR] DEST[Q] COND-ABORT COND[AC=0] ABORT-ADR-12
19 1596                                                  JDSP-SPECIAL-ABORT $
19 1597                                                          ;If AC not =0, abort (and come to JRST0).
19 1598                                                         JUMP[JMPAC] $
19 1599
19 1600                                                   .reloc
19 1601                                                 JRST0:  D[AR] DEST[MA] SHORT $
19 1602                                                          ;Here from trap location 12 if JRST had non-zero AC field.
19 1603                                                         D[IR] ROT[11.] MASK[2] COND[ZERO] LBJUMP[JRST1] C550 $
19 1604                                                          ;Separate according to the BITS 9 AND 10 of the AC field.
19 1605                                                 ]
19m1606                                                   .opcode[254]   [xlist
19 1606                                                    list  ]                                                    ;JRST
19 1607 06530 010731116000000001416162125431456000             COND[-AC=0] LBJUMP[JRST0] $
19 1608 06531 010731170000224334161620254314560000             JUMP[JMPAC] $
19 1609
19m1610                                                   .reloc
19m1610                                                  [.USE[NORMAL]
19m1610                                                  [ xlist
19m1611                                                    list ]                                  ] .pair
19m1611                                                  [:. \ 2 + .
19 1612 05242 550731170000027755416162325421216000       ]JRST0:      TRUE JDSP $              ;JRST 0,
19 1613 05243 010731002000000002605621257614560000             D[IR] ROT[11.] MASK[2] COND[ZERO] LBJUMP[JRST1] C550 $
19 1614                                                                 ;Separate according to the BITS 9 AND 10 of the AC field.
19 1614
19m1615                                                   .pair
19m1615                                                  [:. \ 2 + .
19 1616 05244 610731124000102114161460256714560000       ]JRST1:      D[MA] DEST[PC] COND[USER] JUMP[MUUO] NORM $
19 1617                                                                 ;HERE FOR PI PI DISMIS OR HALT
19 1618                                                                 ;LOAD PC WITH EFFADR
19 1619 05245 010731004000000000276162025761456000             D[IR] ROT[11.] COND[SIGNON] JUMP[JRST2] C550 $  ;Bit 11.
19 1620 05246 010731004000000000316162025761456000       JRST4:  D[IR] ROT[12.] COND[SIGNON] JUMP[JRST3] C550 $  ;Bit 12.
19 1621 05247 010731004000000000256162025761456000       JRST5:  D[IR] ROT[10.] COND[SIGNON] JUMP[JRST9] C550 $  ;Bit 10.
19 1622 05250 010731004000204062361620257614560000             D[IR] ROT[9.]  COND[SIGNON] JUMP[PI-DISMISS] C550 $  ;Bit 9.
19 1623 05251 550731170000027755416162325421216000             JDSP $
19 1624
19 1625                                                 ; Bit 11.--Flag restore
19 1626 05252 010730170043060554161623646374160000       JRSTZ:  CLR-HALF DIPC-FLAGS] DEST[Q] SHORT $
19 1627 05253 010640170000060547403623655774160000             D[CONST 1] ROT[36] ALU[D&Q] DEST[Q] SHORT $
19 1628                                                          ;Don't let USER be cleared.
19 1629 05254 010731000000000004411620257714560000             D[IR] ROT[18.] MASK[4] COND[-OBUS=0] JUMP[JRST21] $
19 1630                                                          ;Indexed?
19 1631 05255 010630170000060554161623656714160000             D[MA] ALU[DORQ] DEST[Q] NORM $
19 1632                                                          ;No.  Put in the new bits (but OR in the old USER)
19 1633 05256 010651170000125146402420255714570000       JRSTZ2: D[CONST 1] ROT[35. - 9.] ALU[-D&Q] DEST[PC-FLAGS] JUMP[JRST4]
19 1633                                                  NORM $
19 1634                                                          ;Don't let bit 9 be set (it is a sort of trap flag for PDL OV)
19 1635
19 1636                                                 ; Indexing used on restore flags.  Get them in wierd manner.
19 1637 05257 010030170000125354161620114314560000       JRSTZ1: ACSEL[IDX] ALU[AORQ] DEST[Q] JUMP[JRSTZZ] $
19 1638
19 1639                                                 ; Bit 12.--Enter user mode
19 1640 05260 010730170000060554161623646314160000       JRST3:  D[PC-FLAGS] DEST[Q] $
19 1641 05261 010631170000125167402420255714570000             D[CONST 1] ROT[36] ALU[DORQ] DEST[PC-FLAGS] JUMP[JRST5] NORM $
19 1642                                                 ; Bit 10.--Halt processor
19 1643 05262 010731370000224234023620255714560000       JRST9:  D[CONST 11] DEST[AR] JUMP[HLTLOC] $     ;MHOFF
19 1644
19m1645                                                   .opcode[255]   [xlist
19 1645                                                    list  ]                                                    ;JFCL
19 1646 06532 010731316000000000321162125771456000             D[IR] ROT[15] MASK[4] DEST[AR] COND[-AC=0] LBJUMP[JFCL1] $
19 1647 06533 010731316000000000321162125771456000             D[IR] ROT[15] MASK[4] DEST[AR] COND[-AC=0] LBJUMP[JFCL2] $
19 1648
19m1649                                                   .reloc
19m1649                                                  [.USE[NORMAL]
19m1649                                                  [ xlist
19m1650                                                    list ]                                  ] .pair
19m1650                                                  [:. \ 2 + .
19 1651 05264 540731170000060554161623254204160000       ]JFCL1:      NEOI $                   ;AC=0, a NOP
19 1652 05265 010730170000060554161623646314160000             D[PC-FLAGS] DEST[Q] $   ;AC is nonzero
19 1653 05266 010641002000125510161620254314560000             D[AR] ROT[40] ALU[D&Q] COND[OBUS=0] JUMP[JFCL1] $
19 1654                                                                 ;TEST SELECTED FLAGS
19 1655 05267 010651170000060550160423654314170000             D[AR] ROT[40] ALU[-D&Q] DEST[PC-FLAGS] $ ;CLEAR FLAGS
19 1656 05270 550731170000027755416162325421216000             JDSP $
19 1657 05271 540731170000060554161623254204160000       JFCL2:  NEOI $
19 1658 05272 010731170000224334161620254314560000             JUMP[JMPAC] $
19 1659
19 1660
```

```
20 1661
20m1662                                                  .opcode[256]   [xlist
20 1662                                                   list  ]                                                    ;XCT
20 1663 06534 010731114000000001416162125431256000              DFRQ COND[AC=0] LBJUMP[XCT1] $
20 1664 06535 010331114000000001416150105431456000              ALU[MEMAC] DEST[HOLD] COND[AC=0] LBJUMP[XCT1] $
20 1665
20 1666
20m1667                                                  .reloc
20m1667                                                 [.USE[NORMAL]
20m1667                                                 [ xlist
20 1668                                                   list ]                                  ]
20 1669
20m1670                                                  .use[HILOC]
20m1670                                                 [ xlist
20m1671                                                   list ]  .pair
20m1671                                                 [:. \ 2 + .
20 1672 11362 010731170000000001416162025475556000              ]XCT1:       D[MEM] TNRM JUMP[XCTP] $
20 1673 11363 540731170000006047416162325465516000      XCT11:  D[MEM] XCT-IDISP $
20 1674                                                          ;The D[MEM] causes us to trap here for map or ECC.  This is
20 1675                                                          ; necessary because the PC doesn't point to the instr we are
20 1675                                                  xct'ing.
20 1676                                                          ;Dispatch without incrementing the PC.
20 1677
20 1678                                                 ;HERE WHEN AC NOT= 0
20 1679                                                 ;MEM/ CONTENTS PTED TO BY XCT INSTRUCTION (INSTR TO XCT)
20 1680                                                 ;INSTR NOT YET TRAPPED FOR INDEXING OR INDIRECTION SO CAN DO EFF ADR CALC
20 1680
20 1681                                                 ;IN APPROPRIATE SPACE
20 1682                                                 ;PC/ XCT INSTR + 1
20 1683
20 1684 11364 010731124000227474161620254354560000      XCTP:   COND[USER] JUMP[XCT11] TNRM $
20 1685                                                                  ;LIKE A NORMAL XCT (AC=0) IF USER MODE
20 1686 11365 010730170000000000321162025775456000              D[IR] ROT[13.] MASK[4] DEST[Q] JUMP[XCTMRG] TNRM $
20 1687                                                                  ; SAVE XCT AC FLD AND MERGE
20 1688
20 1689
20 1690                                                 ;MERGE POINT FOR UMOVEX AND SPECIAL XCT
20 1691
20 1692                                                 .REPEAT 1 - WAITS [
20 1693                                                 VSTAADR = 775000         ;VIRTUAL ADR OF STATE/AC PAGE
20 1694                                                                          ;THIS PAGE IS KNOWN TO BE LOCKED IN CORE!
20 1695                                                 ];.REPEAT 1 - WAITS
20 1696
20 1697                                                 ;MERGE POINT FOR UMOVEX.
20 1698                                                 ;MEM/ INSTR TO XCT (MIGHT BE UMOVX CONVERTED TO MOVX)
20 1699                                                 ;       EFF ADR CALC NOT YET TRAPPED UP FOR INSTR TO XCT
20 1700                                                 ;Q/ AC BITS OF XCT INSTR (MIGHT BE BITS FAKED BY UMOVE GROUP)
20 1701                                                 ;PC/ UMOVE INSTR + 1 OR XCT INSTR + 1
20 1702
20 1703 11366 710731370000060554161403654755160000      XCTMRG: D[MEM] DEST[AR IR-ALL MA] TNRM $
20 1704                                                                  ;Load AR,IR,MA from MEM.
20 1705 11367 010731002000000004413620254615560000              D[MEM] ROT[18.] MASK[5] COND[ZERO] JUMP[XCTEFF] CS50 $
20 1706                                                                  ;Jump if no indexing or indirection necessary.
20 1707 11370 010731002000000004411620257614560000      XCTNDX: D[IR] ROT[18.] MASK[4] COND[ZERO] JUMP[XCTIND] CS50 $
20 1708                                                                  ;Jump if no index fld.
20 1709 11371 710501370000060554045443517714160000              D[IR] MASK[18.] ALU[D+XR] DEST[MA IR-ADR AR] $
20 1710                                                                  ;Do index (load AR,IR,MA)
20 1711 11372 010731006000000003361620257614560000      XCTIND: D[IR] ROT[13.] COND[SIGNOFF] JUMP[XCTEFF] CS50 $
20 1712                                                                  ;Jump if no indr
20 1713 11373 010641000000000001402162225561456000              D[CONST 10] ALU[D&Q] COND[-ZERO] PUSHJ[XFIXMA] CS50 $
20 1714                                                                  ;Pushj if addr calc user, fixup MA and adr space select.
20 1714
20 1715 11374 010331170000060554161503454352160000              DFRQ ALU[MEMAC] DEST[HOLD] TNRM $
20 1716                                                                  ;Load MEM BUS with either contents of AC (if MA<Z0) or
20 1717                                                                  ; MEM via fetch (hdwr hack makes right thing happen).
20 1718                                                                  ;All we want here is xct access but map ucode
20 1719                                                                  ;allows xct or read if either on in map ptr
20 1720                                                                  ;so can take normal read trap and win.
20 1721 11375 710731370000060554161423654755160000              D[MEM] DEST[AR IR-23 MA] TNRM $
20 1722                                                                  ;LOAD AR,MA, IR BUT DON'T CLOBBER OPCODE OR AC
20 1723 11376 010731170000060554000403655754170000              SET-TEMP-EXEC  [ D[CONST 00] DEST[MAP-USER-SR] ]TNRM $
20 1724                                                                  ;BACK TO EXEC MODE INCASE XFIXMA SELECTED USER SPACE
20 1725 11377 010731000000227604413620254615560000              D[MEM] ROT[18.] MASK[5] COND[-ZERO] JUMP[XCTNDX] CS50 $
20 1726                                                                  ;JUMP IF INDEXING OR INDIRECTION NECESSARY
20 1727 11400 610231170000060554161443654354160000      XCTEFF: ALU[Q] DEST[IR-ADR] TNRM $
20 1728                                                                  ;SAVE XCT AC FLD IN IR RH NOW
20 1729 11401 010730170000060541617623657754160000              D[IR] ROT[7] MASK[7] DEST[Q] TNRM $
20 1730                                                                  ;GET HO 7 BITS OF OPCODE
20 1731 11402 010661002000000014131620255614560000              D[CONST 54] ALU[D#Q] COND[ZERO] JUMP[XCTSTK] CS50 $
20 1732                                                                  ;JUMP IF STACK GROUP
20 1733 11403 010661002000000001405762025561456000              D[CONST 27] ALU[D#Q] COND[ZERO] JUMP[XCTBYT] CS50 $
20 1734                                                                  ;JUMP IF BYTE GROUP
20 1735 11404 010661002000000001412562025561456000              D[CONST 52] ALU[D#Q] COND[ZERO] JUMP[XCTBLG] CS50 $
20 1736                                                                  ;JUMP IF GROUP WITH BLT IN IT
20 1737 11405 010730170000060554011623657754160000      XCTEZE: D[IR] MASK[4] DEST[Q] TNRM $
20 1738                                                                  ;GET XCT AC FLD
20 1739 11406 010641002000000001401362025561456000              D[CONST 5] ALU[D&Q] COND[ZERO] JUMP[XCTEE] CS50 $
20 1740                                                                  ;JUMP IF NEITHER RELEVANT BIT
20 1741 11407 010731110000000001416162025435456000              COND[MA<Z0] JUMP[XCTAC] TNRM $
20 1742                                                                  ;JUMP IF EF ADR IS AC
20 1743 11410 010731004000000000176162024621456000              D[PC-FLAGS] ROT[7.] COND[SIGNON] JUMP[XCTEE] CS50 $
20 1744                                                                  ;JUMP IF CALL FM MON
20 1745 11411 010641002000000001400362025561456000              D[CONST 1] ALU[D&Q] COND[ZERO] JUMP[XCTEU] CS50 $
20 1746                                                                  ;JUMP IF BIT OFF
20 1747 11412 010641002000000001401162025561456000              D[CONST 4] ALU[D&Q] COND[ZERO] JUMP[XCTUE] CS50 $
20 1748                                                                  ;JUMP IF OTHER BIT OFF
20 1749 11413 010731170000000001403640025575457000      XCTUU:  D[CONST 17] DEST[MAP-USER-SR] JUMP[XDISP] TNRM $
20 1750                                                                  ;SET SRC & DEST TO USER
20 1751
20 1752 11414 010731170000000001403440025575457000      XCTUE:  D[CONST 16] DEST[MAP-USER-SR] JUMP[XDISP] TNRM $
20 1753                                                                  ;SET DEST TO USER, SRC TO EXEC
20 1754
20 1755 11415 010731170000000001400240025575457000      XCTEU:  D[CONST 01] DEST[MAP-USER-SR] JUMP[XDISP] TNRM $
20 1756                                                                  ;SET SRC TO USER, DEST TO EXEC
20 1757
20 1758
20 1759                                                 XCTEE:
20 1760 11416 710731170000000001416144025675456000      XDISP:  D[MA] DEST[MA IR-ADR] JUMP[XCTEOI] TNRM $
20 1761                                                                  ;FINISH UP INSTR MODIFICATION
20 1762
```

```
20 1768
20 1769
20 1770                                                 ;DISPATCH TO EXECUTE INSTRUCTION
20 1771
20 1772 11421 010730170000060554161623655775416000     XCTEOI: D[IR] DEST[Q] TNRM $
20 1773 11422 610651170000060544413403647355416000             D[MASK 5] ROT[18.] ALU[-D&Q] DEST[IR-ALL] TNRM $
20 1774                                                                 ;Remove the XR and IND fields... eff. addr. already
20 1774                                                  calculated.
20 1775 11423 010731170000060554161503655775416000             D[IR] DEST[HOLD] TNRM $
20 1776                                                                 ;FAKE INSTR FETCH
20 1777 11424 440731170000060474161623254254160000             PXCT-IDISP $
20 1778                                                                 ;DISPATCH WITHOUT INSTR FETCH #
20 1779                                                                 ;PC DOES NOT GET INCREMENTED
20 1780                                                                 ;MA DOES NOT GET CHANGED
20 1781                                                                 ;MAP-USER-SR DOES NOT GET CHANGED
20 1782
20 1783
20 1784
```

```
21 1785
21 1786
21 1787                                                 ;Make address in MA refer to previous context (including stored AC's).
21 1788                                                 ;MUST NOT CLOBBER Q or HOLD !!!
21 1789
21 1790 11425 010731110000000014000400255754570000     XFIXMA: SET-TEMP-EXEC  [ D[CONST 00] DEST[MAP-USER-SR] ]COND[MA<Z0]
21 1790                                                  JUMP[XFIXAC1] TNRM $
21 1791                                                         ;Select EXEC to start, jump if addrs is AC.
21 1792 11426 010731004000060541761624246114160000             D[PC-FLAGS] ROT[7.] LONG COND[SIGNON] POPJ  $
21 1793                                                         ;Leave if CALL FM MON
21 1794 11427 010731170000060554036403655754170000             SET-TEMP-USER  [ D[CONST 17] DEST[MAP-USER-SR] ]TNRM $
21 1795 11430 010731170000060554161624254354160000             POPJ TNRM $
21 1796                                                         ;2 INSTR SO MAP SETTLES INCASE DFRQ ON NEXT INSTR
21 1797
21 1798
21 1799                                                 ;Load MA with adr of user saved AC. MUST NOT CLOBBER Q!!!
21 1800
21 1801 11431 010731170000060554000403655754170000     XFIXAC: SET-TEMP-EXEC  [ D[CONST 00] DEST[MAP-USER-SR] ]TNRM $
21 1802                                                         ;Insure EXEC space selected.
21 1803                                                 XFIXAC1:
21 1804                                                   .repeat 1 - WAITS [
21 1805                                                          D[MA] DEST[ALU1 Q] TNRM $
21 1806                                                                 ;Get AC addrs.
21 1807                                                          D[AMEM-ABS MAP-AGE/AC8] MASK[9.] ALU[ALU1 D+Q] DEST[Q] TNRM $
21 1808                                                                 ;Add in AC BASE REG.
21 1809                                                          D[LIT VSTAADR] ALU[ALU1 DORQ] DEST[MA] TNRM POPJ $
21 1810                                                                 ;Include virtual page # of STATE/AC BLOCK in adr.
21 1811                                                   ];.repeat 1 - WAITS
21 1812
21 1813                                                   .repeat WAITS [
21 1814
21 1815                                                         D[PC-FLAGS] ROT[8.] MASK[1] COND[-ZERO] POPJ CS50 MAPF[2] $
21 1816                                                                 ;LEAVE IF CALL FM MON
21 1817                                                         D[MA] DEST[ALU1 Q] $
21 1818                                                                 ;Get address of AC, we need to mung it
21 1819                                                 ;*** 770000 ought to be defined symbolicly
21 1820                                                         D[LIT 770000] ALU[ALU1 DORQ] DEST[MA] TNRM $
21 1821                                                                 ;Add page number of window
21 1822                                                                 ;Use 770xxx for window into user ACs
21 1823                                                         D[AMEM-ABS MAP-AGE/AC8] ROT[9.] MASK[36. - 13.] DEST[Q] $
21 1824                                                                 ;Get phys. page no. of user page 0.
21 1825                                                                 ;If for some perverse reason, there is no user page 0,
21 1826                                                                 ;we will reference exec AC's shadow, which is presumably
21 1826
21 1827                                                                 ;harmless
21 1828                                                         D[CONST 3] ROT[24.] ALU[DORQ] DEST[MAP-MA] LONG $
21 1829                                                                 ;Include access permission bits and load into Map Ram.
21 1830                                                         ALU[-1] DEST[MAP-TAG] LONG POPJ  $
21 1831                                                                 ;Set MAP-TAG to be always valid (no compare !)
21m1832                                                   ];[
21m1832
21m1832 11432 010731000200060542003624246214160000             D[PC-FLAGS] ROT[8.] MASK[1] COND[-ZERO] POPJ CS50 MAPF[2] $
21m1832                                                                 ;LEAVE IF CALL FM MON
21m1832 11433 010730570000060554161623656714160000             D[MA] DEST[ALU1 Q] $
21m1832                                                                 ;Get address of AC, we need to mung it
21m1832                                                 ;*** 770000 ought to be defined symbolicly
21m1832 11434 710631570000001760001623655354160000             D[LIT 770000] ALU[ALU1 DORQ] DEST[MA] TNRM $
21m1832                                                                 ;Add page number of window
21m1832                                                                 ;Use 770xxx for window into user ACs
21m1832 11435 010730170000000422257623670314160000             D[AMEM-ABS MAP-AGE/AC8] ROT[9.] MASK[36. - 13.] DEST[Q] $
21m1832                                                                 ;Get phys. page no. of user page 0.
21m1832                                                                 ;If for some perverse reason, there is no user page 0,
21m1832                                                                 ;we will reference exec AC's shadow, which is presumably
21m1832
21m1832                                                                 ;harmless
21m1832 11436 010631170000060546006543655514170000             D[CONST 3] ROT[24.] ALU[DORQ] DEST[MAP-MA] LONG $
21m1832                                                                 ;Include access permission bits and load into Map Ram.
21m1832 11437 010721170000060554000564255514170000             ALU[-1] DEST[MAP-TAG] LONG POPJ  $
21m1832                                                                 ;Set MAP-TAG to be always valid (no compare !)
21 1832                                                   ].repeat WAITS
21 1833
21 1834
```

```
22 1835
22 1836
22 1837                                                 ;STACK GROUP
22 1838
22 1839 11440 010730002000230122205620257614560000     XCTSTK: D[IR] ROT[9.] MASK[2] DEST[Q] COND[ZERO] JUMP[XCTEZE] C550 $
22 1840                                                         ;GET 2 LOW ORDER BITS OF OP CODE. JUMP IF PUSHJ
22 1841 11441 010661002000230134007620255614560000             D[CONST 3] ALU[D=Q] COND[ZERO] JUMP[XCTEZE] C550 $
22 1842                                                         ;JUMP IF POPJ
22 1843 11442 010661002000000014005620255614560000             D[CONST 2] ALU[D=Q] COND[ZERO] JUMP[XPOP1] C550 $
22 1844                                                         ;JUMP IF POP
22 1845
22 1846                                                 ;Special case for PUSH.  Check for AC reference from stack pointer
22 1847
22 1848 11443 010731006000230130761620257614560000     XPUSH1: D[IR] ROT[35.] COND[SIGNOFF] JUMP[XCTEZE] C550 $
22 1849                                                         ;If not mapping stack references, it's simple
22 1850 11444 010731004000230530361622257614560000             D[IR] ROT[33.] COND[SIGNON] PUSHJ[XFIXMA]  C550$
22 1851                                                         ;Make sure we're referring to the right place here.
22 1852 11445 010331170000060554161503454352160000             DFRQ ALU[MEMAC] DEST[HOLD] TNRM $
22 1853                                                         ;LOAD MEM BUS WITH EITHER CONTENTS OF AC IF MA<20
22 1854                                                         ;OR MEM VIA FETCH (HOWR HACK MAKES RIGHT THING HAPPEN)
22 1855                                                         ;Check page faults for effective address part.
22 1856 11446 010731370400060554161623654755160000             RE-XCT D[MEM] DEST[AR] TNRM $
22 1857                                                         ;Copy away the thing that we want to push
22 1858 11447 710503030037000014003622255614560000             D[1,,1] ALU[D+AC] DEST[AC MA] COND[CRY0] PUSHJ[PDLO] C550 $
22 1859                                                         ;advance frame pointer and set pc flag if necessary
22 1860 11450 010731170000230534161622254354560000             PUSHJ[XFIXMA] TNRM $
22 1861                                                         ;Map destination.
22 1862 11451 010731310000226654161561254304560000             D[AR] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<20] LBJUMP[SEOI] ]$
22 1863                                                         ;Finish instruction. We're done
22 1864
22 1865
22 1866                                                 ;Special case for POP.  Check for AC reference from stack pointer
22 1867
22 1868 11452 010730170000060554161623657754160000     XPOP1:  D[IR] DEST[Q] TNRM $
22 1869                                                         ;Save bit meaning 'source from user' IN Q
22 1870 11453 610731170000060554161443656754160000             D[MA] DEST[IR-AOR] TNRM $
22 1871                                                         ;Restore normal effective address
22 1872 11454 711512170037060554003623655754160000             ACSEL[AC] D[1,,1] ALU[AC-D] DEST[MA 0-AC] TNRM $
22 1873                                                         ;Update AC and MA for stack reference
22 1874 11455 010641000000230534011622255614560000             D[CONST 4] ALU[D&Q] COND[-ZERO] PUSHJ[XFIXMA] C550 $
22 1875                                                         ;Fixup stack addressing if user AC
22 1876 11456 010331170000060554161503454352160000             DFRQ ALU[MEMAC] DEST[HOLD] TNRM $
22 1877                                                         ;LOAD MEM BUS WITH EITHER CONTENTS OF AC IF MA<20
22 1878                                                         ;OR MEM VIA FETCH (HOWR HACK MAKES RIGHT THING HAPPEN)
22 1879 11457 010731170000060554161623654755160000             D[MEM] TNRM $
22 1880                                                         ;Cause Map Trap to happen here, before we change MA...
22 1881 11460 710701170041606055404562365775416000             CLR-MAP-SR D[IR] MASK[18.] DEST[MA] TNRM $
22 1882                                                         ;Reset previous special user mode, recover effective
22 1882                                                  address.
22 1883 11461 010641000000230534003622255614560000             D[CONST 1] ALU[D&Q]  C550 COND[-ZERO] PUSHJ[XFIXMA] $
22 1884                                                         ;Fix store address if the XCTP specified it...
22 1885 11462 010501030037000014003622255614560000             D[1,,1] ALU[D+AC] COND[CRY0] PUSHJ[PDLO] C550 $
22 1886                                                         ;Set pc flag if pointer underflowed.
22 1887 11463 010731310000226654161561254705560000             D[MEM] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<20] LBJUMP[SEOI] ]$
22 1888                                                         ;Write out thing we POP'ped
22 1889
22 1890
22 1891
```

```
23 1892
23 1893
23 1894                                                 ;BYTE GROUP
23 1895
23 1896 11464 010731004000230530361622257614560000     XCTBYT: D[IR] ROT[33.] COND[SIGNON] PUSHJ[XFIXMA] C550 $
23 1897                                                         ;Pushj if B.P. fetch is from previous context.
23 1898 11465 010331170000060554161503454312160000             DFRQ ALU[MEMAC] DEST[HOLD] $
23 1899                                                         ;Fetch BP, allowing for possiblity it is in an AC.
23 1900 11466 010731004000000002161620257614560000             D[IR] ROT[8.] COND[SIGNON] JUMP[XBYNOI] C550 $
23 1901                                                         ;Jump if instr. does not incr. BP.
23 1902 11467 010730235000000001416162025475556000             R-M-W D[MEM] DEST[AR Q] COND[HALF] JUMP[XBYZND] TNRM $
23 1903                                                         ;RMW trap possible.
23 1904                                                         ;GET BP, JUMP IF INCED IT ALREADY
23 1905 11470 010731310000000000301562025421456000             D[AR] ROT[12.] MASK[6] DEST[AR] COND[MA<Z0] JUMP[XBYUDA] C550 $
23 1906                                                         ;Jump if dealing with AC, put S in AR right justified.
23 1907 11471 010731170000105774161622254354560000             PUSHJ[IBPX] TNRM $
23 1908                                                         ;INC BP NOT IN AC
23 1909 11472 010731170000000001416162225435456000             PUSHJ[SET-HALF] TNRM $
23 1910                                                         ;SET HALF SO DON'T BUMP BP AGAIN IF TRAP LATER
23 1911 11473 710730370000000001416162025475556000             D[MEM] DEST[AR Q MA] JUMP[XBYZND] TNRM $
23 1912                                                         ;MEM,AR,MA / BP
23 1913                                                 ;Here to inc BP in AC.
23 1914                                                 XBYUDA:
23 1915 11474 010731170000106134161502256754560000             D[MA] DEST[HOLD] PUSHJ[IBPXA] TNRM $
23 1916                                                         ;INC BP IN AC (IBPXA needs eff. addr. in MEM)
23 1917 11475 010731170000000001416162225435456000             PUSHJ[SET-HALF] TNRM $
23 1918                                                         ;SET HALF SO DON'T BUMP BP AGAIN IF TRAP LATER
23 1919 11476 010731170000000001416162025435456000             JUMP[XBYZND] TNRM $
23 1920                                                         ;AC,MEM,AR,MA / BP
23 1921
23 1922                                                 ;HERE IF NO NEED TO INC BP
23 1923                                                 ;MEM/ BP
23 1924
23 1925 11477 010730370000060554161623654755160000     XBYNOI: D[MEM] DEST[AR Q] TNRM $
23 1926                                                         ;READ TRAP POSSIBLE
23 1927                                                         ;GET BP
23 1928 11500 010731170000060554000403655754170000     XBYZND: SET-TEMP-EXEC  [ D[CONST 00] DEST[MAP-USER-SR] ]TNRM $
23 1929                                                         ;FORCE EXEC SPACE
23 1930 11501 010731170000060543014503654314170000             D[AR] ROT[12.] MASK[6] DEST[MASKR] $
23 1931 11502 710731170000060554045623654354160000             D[AR] MASK[18.] DEST[MA] TNRM $
23 1932                                                         ;GET Y OF BP IN MA
23 1933 11503 010731002000000000441362025461556000             D[MEM] ROT[18.] MASK[5] COND[ZERO] JUMP[XBYEFF] C550 $
23 1934                                                         ;JUMP IF BP HAS NO IDX OR INDR
23 1935 11504 010731002000000000441162025421456000     XBYNDX: D[AR] ROT[18.] MASK[4] COND[ZERO] JUMP[XBYIND] C550 $
23 1936                                                         ;JUMP IF NO IX FLD
23 1937                                                         ;LOAD AC-CNT WITH INDEX REG #
23 1938 11505 710731170000060544411623654354160000             D[AR] ROT[18.] MASK[4] DEST[MA] TNRM $
23 1939                                                         ;LOAD MA WITH AC ADR FOR AC SELECT FOR D+AC
23 1940                                                         ;OK CAUSE MA GETS UPDATED LATER
23 1941 11506 010501370000060554045623414354160000             D[AR] MASK[18.] ACSEL[MA,AC] ALU[D+AC] DEST[AR] TNRM $
23 1942                                                         ;SELECT AC A INPUT VIA MA
23 1943                                                         ;ADD INDEX REG CONTENTS TO Y IN AR
23 1944 11507 010641070000060544445623647354160000             D[MASK 18.] ROT[18.] ALU[D&Q] DEST[Q] TNRM $
23 1945                                                         ;KEEP LEFT HALF OF BP IN Q
23 1946 11510 710630370000060554045623654354160000             D[AR] MASK[18.] ALU[DORQ] DEST[Q AR MA] TNRM $
23 1947                                                         ;FORM NEW BP WITH INDEXING DONE IN Y (IGNORE INDEX FIELD
23 1947                                                  NOW)
23 1948 11511 010731006000000000336162025421456000     XBYIND: D[AR] ROT[13.] COND[SIGNOFF] JUMP[XBYEFF] C550 $
23 1949                                                         ;JUMP IF NO INDR
23 1950 11512 010731004000230530561622257614560000             D[IR] ROT[34.] COND[SIGNON] PUSHJ[XFIXMA] C550 $
23 1951                                                         ;PUSHJ IF BP ADR CALC IS USER
23 1952 11513 030731170000060554000403655752170000             DF/WT SET-TEMP-EXEC  [ D[CONST 00] DEST[MAP-USER-SR] ]TNRM $
23 1953                                                         ;FETCH INDIRECT WORD AND UNDO SET-TEMP-USER IN FIXMA
23 1954 11514 010731370000060543031623654354160000             D[AR] ROT[12.] MASK[12.] DEST[AR] TNRM $
23 1955                                                         ;GET P AND S FILEDS IN RH AR
23 1956 11515 010730170000060546161623654354160000             D[AR] ROT[24.] DEST[Q] TNRM $
23 1957                                                         ;POSITION THEM IN Q
23 1958 11516 710630370000060554057623654754160000             D[MEM] MASK[23.] ALU[DORQ] DEST[Q AR MA] TNRM $
23 1959                                                         ;FORM NEW BP USING OLD S AND P AND NEW IND,NDX AND Y
23 1960 11517 010731000000232104413620254615560000             D[MEM] ROT[18.] MASK[5] COND[-ZERO] JUMP[XBYNDX] C550 $
23 1961                                                         ;JUMP IF MORE INDEX OR INDIRECTION NECESSARY
23 1962 11520 010731004000230530761622257714560000     XBYEFF: D[IR] ROT[35.] COND[SIGNON] PUSHJ[XFIXMA] $
23 1963                                                         ;MAKE MA OK
23 1964 11521 010731006000105121761620257614560000             D[IR] ROT[7.] COND[SIGNOFF] JUMP[ILDBS] C550 $
23 1965                                                         ;DISTINGUISH LOADS FROM DEPOSITS, JUMP IF LOAD
23 1966 11522 010731170000105634161620254354560000             JUMP[IDPBS] TNRM $
23 1967                                                         ;JUMP IF DEPOSIT
23 1968
23 1969
```

```
24 1970
24 1971
24 1972                                                 ;GROUP WITH BLT IN IT
24 1973
24 1974                                                 .repeat waitsfix [      ;Part of GRUNT avoidance
24 1975                                                         .ODD
24m1976                                                 ];[     ;Part of GRUNT avoidance
24m1976                                                         .ODD
24m1976                                                  [:. + 1 - (. \ 2)
24 1976                                                    ]].repeat waitsfix
24 1977 11523 010730002000230122205620257614560000     XCTBLG: D[IR] ROT[9.] MASK[2] DEST[Q] COND[ZERO] JUMP[XCTEZE] C550 $
24 1978                                                                 ;GET 2 LO BITS OF OPCODE, JUMP IF EXCH
24 1979 11524 010661000000230134003620255614560000             D[CONST 1] ALU[D#Q] COND[-ZERO] JUMP[XCTEZE] C550 $
24 1980                                                                 ;JUMP IF NOT BLT
24 1981                                                 XCTBLT:
24 1982                                                 .repeat 1 - waitsfix [
24 1983                                                         D[MA] MASK[18.] DEST[O-AC AR] TNRM $
24 1984                                                                 ;END-ADR IN AC, AC IN AR
24 1985                                                 ];.repeat 1 - waitsfix
24 1986                                                 .repeat waitsfix [      ;GRUNT avoidance
24 1987                                                         D[MA] MASK[18.] DEST[O-AC AR] TNRM PUSHJ[XBLTLP] $
24 1988                                                                 ;END-ADR IN AC, AC IN AR
24 1989                                                          TNRM JUMP[ECC-PROCEED] $
24 1990                                                                 ;Nothing special for ECC errors
24 1991                                                          TNRM D[AR] DEST[O-AC AR] PUSHJ[MAP-RET] $
24 1992                                                                  ;Unscramble registers
24 1993                                                          JUMP[.] $
24m1994                                                 ];[     ;GRUNT avoidance
24m1994 11525 010732370000000001404562225675456000             D[MA] MASK[18.] DEST[O-AC AR] TNRM PUSHJ[XBLTLP] $
24m1994                                                                 ;END-ADR IN AC, AC IN AR
24m1994 11526 010731170000223534161620254354560000              TNRM JUMP[ECC-PROCEED] $
24m1994                                                                 ;Nothing special for ECC errors
24m1994 11527 010732370000212374161622254354560000              TNRM D[AR] DEST[O-AC AR] PUSHJ[MAP-RET] $
24m1994                                                                 ;Unscramble registers
24m1994 11530 010731170000232614161620254314560000              JUMP[.] $
24 1994                                                 ].repeat waitsfix
24 1995
24 1996 11531 010731170000060554000403655754170000     XBLTLP: SET-TEMP-EXEC  [ D[CONST 00] DEST[MAP-USER-SR] ]TNRM $
24 1997                                                                 ;USE EXEC FOR FETCH TIL TOLD BETTER
24 1998 11532 710731170000060544445623654354160000             D[AR] ROT[18.] MASK[18.] DEST[MA] TNRM $
24 1999                                                                 ;GET READY FOR DATA WORD
24 2000 11533 010731004000230530361622257614560000             D[IR] ROT[33.] COND[SIGNON] PUSHJ[XFIXMA] C550 $
24 2001                                                                 ;FIX IF USER ADDRS
24 2002 11534 010331170000060554161503454354160000             DFRQ ALU[MEMAC] DEST[HOLD] TNRM $
24 2003                                                                 ;LOAD MEM BUS WITH EITHER CONTENTS OF AC IF MA<Z0
24 2004                                                                 ;OR MEM VIA FETCH (HDWR HACK MAKES RIGHT THING HAPPEN)
24 2005                                                 .repeat 1 - waitsfix [
24 2006                                                         D[MEM] DEST[HOLD] TNRM $
24 2007                                                 ];.repeat 1 - waitsfix
24 2008                                                 .repeat waitsfix [
24 2009                                                         TRP-CTL[TRAP-FIX RE-XCT] D[MEM] DEST[HOLD] TNRM $
24m2010                                                 ];[
24m2010 11535 010731170500060554161503654754516000             TRP-CTL[TRAP-FIX RE-XCT] D[MEM] DEST[HOLD] TNRM $
24 2010                                                 ].repeat waitsfix
24 2011                                                                 ;GET DATA WORD. POSSIBLE FETCH TRAP HERE.
24 2012 11536 710731170000060554045623654354160000             D[AR] MASK[18.] DEST[MA] TNRM $
24 2013                                                                 ;GET DEST ADR IN MA
24 2014 11537 010731170000060554000403655754170000             SET-TEMP-EXEC  [ D[CONST 00] DEST[MAP-USER-SR] ]TNRM $
24 2015                                                                 ;USE EXEC FOR STORE TIL TOLD BETTER
24 2016 11540 010731004000230530761622257614560000             D[IR] ROT[35.] COND[SIGNON] PUSHJ[XFIXMA] C550 $
24 2017                                                                 ;FIX MA IF NECESSARY
24 2018                                                 .repeat 1 - waitsfix [
24 2019                                                         D[1,,1] DEST[Q] COND[MA<Z0] JUMP[XBLTSA] C550 $
24 2020                                                                 ;PUT 1,,1 IN Q, JUMP IF STORE INTO AC
24 2021                                                 ];.repeat 1 - waitsfix
24 2022                                                 .repeat waitsfix [
24 2023                                                         ALU[AC] DEST[Q] COND[MA<Z0] JUMP[XBLTSA] C550 $
24 2024                                                                 ;Setup address for end-comparison
24m2025                                                 ];[
24m2025 11541 010330110000000001416162025421456000             ALU[AC] DEST[Q] COND[MA<Z0] JUMP[XBLTSA] C550 $
24m2025                                                                 ;Setup address for end-comparison
24 2025                                                 ].repeat waitsfix
24 2026 11542 010731170000000001416156025475556000             D[MEM] DEST[MEMSTO] JUMP[XBLTL1] TNRM $
24 2027                                                                 ;WRITE DATA WORD VIA (FIXED) MA TO MEM DESTINATION
24 2028
24 2029 11543 010733170000060554161623454754516000     XBLTSA: D[MEM] ACSEL[MA] DEST[AC] TNRM $
24 2030                                                                 ;Store into real AC
24 2031                                                 .repeat 1 - waitsfix [
24 2032                                                 XBLTL1: D[AR] MASK[18.] ALU[D-AC] COND[SIGNOFF] JUMP[XBLTDN] C600 $
24 2033                                                                 ;JUMP IF DONE
24 2034                                                                 ;POSSIBLE WRITE TRAP HERE
24 2035                                                         D[AR] ALU[D+Q] DEST[AR] JUMP[XBLTLP] TNRM $
24 2036                                                                 ;UPDATE FROM,,TO IN AR AND LOOP TIL DONE
24 2037                                                                 ;NOTE:  NON-INTERRUPTABLE IF NO TEST FOR INT HERE!!
24 2038                                                 ];repeat 1 - waitsfix
24 2039                                                 .repeat waitsfix [
24 2040                                                 XBLTL1: TRP-CTL[TRAP-FIX RE-XCT] D[AR] MASK[18.] ALU[D-Q]
24 2041                                                                         COND[SIGNOFF] JUMP[XBLTDN] C600 $
24 2042                                                                 ;Jump if done
24 2043                                                                 ;possible Write Trap here
24 2044                                                         D[1,,1] DEST[Q] TNRM $
24 2045                                                         D[AR] ALU[D+Q] DEST[AR] C500 COND[-INTRPT] JUMP[XBLTLP] $
24 2046                                                                 ;UPDATE FROM,,TO IN AR AND LOOP TIL DONE
24 2047                                                                 ;NOTE:  NON-INTERRUPTABLE IF NO TEST FOR INT HERE!!
24 2048                                                         D[AR] DEST[AC] TNRM $
24 2049                                                                 ;Restore AC, which is ready for next fetch
24 2050                                                         D[PC] ALU[D-1] DEST[PC] JUMP[MAIN] $
24 2051                                                                 ;Go process interrupt
24m2052                                                 ];[
24m2052                                                 XBLTL1: TRP-CTL[TRAP-FIX RE-XCT] D[AR] MASK[18.] ALU[D-Q]
24m2052 11544 011621006500000001404562025411556000                     COND[SIGNOFF] JUMP[XBLTDN] C600 $
24m2052                                                                 ;Jump if done
24m2052                                                                 ;possible Write Trap here
24m2052 11545 010730170037050554003623655754160000             D[1,,1] DEST[Q] TNRM $
24m2052 11546 010601306000232634161620254214560000             D[AR] ALU[D+Q] DEST[AR] C500 COND[-INTRPT] JUMP[XBLTLP] $
24m2052                                                                 ;UPDATE FROM,,TO IN AR AND LOOP TIL DONE
24m2052                                                                 ;NOTE:  NON-INTERRUPTABLE IF NO TEST FOR INT HERE!!
24m2052 11547 010733170000060554161623654354160000             D[AR] DEST[AC] TNRM $
24m2052                                                                 ;Restore AC, which is ready for next fetch
24m2052 11550 610721170000227114161460256314560000             D[PC] ALU[D-1] DEST[PC] JUMP[MAIN] $
24m2052                                                                 ;Go process interrupt
```

```
24 2056
24m2057                                                  .use[NORMAL]
24m2057                                                 [ xlist
24 2058                                                   list ]
24 2059
24 2060
24 2061
```

```
25 2062
25 2063                                                 ;;; Stack instructions
25 2064
25m2065                                                   .opcode[261]   [xlist
25 2065                                                    list  ]                                                    ;PUSH
25 2066
25 2067 06542 710503032037000001400362125571256000             DFRQ D[1,,1] ALU[AC+D] DEST[AC MA]   -CRY0 LBJUMP[PUSH1] $
25 2068                                                               ; Read data, kick ptr, test for overflow
25 2069 06543 010331370000000001416150005431456000             ALU[MEMAC] DEST[AR HOLD] NORM JUMP[PUSH2] $
25 2070                                                               ; Move data to AR and HOLD so it can be stored.
25m2071                                                   .reloc
25m2071                                                  [.USE[NORMAL]
25m2071                                                  [ xlist
25m2072                                                    list ]                                    ]  .pair
25m2072                                                  [:. \ 2 + .
25 2073 05274 010731170000000001416162225431456000       ]PUSH1:       NORM PUSHJ[PDLO] $
25 2074                                                         ; PDL overflow - set bit in PC-FLAGS
25 2075 05275 010731310200226654161561254705560000       D[DF-FROM-IR MEM] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<20]
25 2075                                                  LBJUMP[SEOI] ]$
25 2076                                                         ; Store data on stack and exit
25 2077
25 2078 05276 710503120037000014003641255704560000     PUSH2:  IFRQ D[1,,1] ALU[AC+D] DEST[AC MA] STRT-WRT EA<20 LBJUMP[PUSH3] $
25 2078
25 2079                                                         ;Push of an AC.  Kick ptr, test for pdl in ACs
25m2080                                                   .pair
25m2080                                                  [:. \ 2 + .
25 2081 05300 540501030000071554141623247254160000       ]PUSH3:       D[MASK 60]  ALU[D+AC]  COND-ABORT COND[CRY0] ABORT-ADR-10
25 2081                                                  EOI $
25 2082                                                         ;Abort to trap location 10 if A00:17 = 0; this gets us to
25 2082                                                  PDL-TRP.
25 2083 05301 010733170000126014161620054314760000             D[AR] DEST[MEMAC] NORM JUMP[PUSH3] $
25 2084                                                           ;Here if the PDL is in the AC's; store data on PDL.
25 2085
25m2086                                                   .opcode[262]   [xlist
25 2086                                                    list  ]                                                    ;POP
25 2087 06544 711512230037000014003621255714560000             D[1,,1] ALU[AC-D] DEST[D-AC MA AR] COND[CRY0] LBJUMP[POP1] $
25 2088 06545 711512230037000014003621255714560000             D[1,,1] ALU[AC-D] DEST[D-AC MA AR] COND[CRY0] LBJUMP[POP2] $
25 2089                                                               ; Decrement ptr, put OLD value into MA
25m2090                                                   .reloc
25m2090                                                  [.USE[NORMAL]
25m2090                                                  [ xlist
25m2091                                                    list ]                                    ]  .pair
25m2091                                                  [:. \ 2 + .
25 2092 05302 010731170000000001416162225431456000       ]POP1:        PUSHJ[PDLO] $           ; Set PDL ovflow
25 2093 05303 710731110000000001404562225771256000             DFRQ D[IR] MASK[18.] DEST[MA] MA<20 PUSHJ[POP3] $
25 2094                                                               ; Get back original MA from IR, check for pdl in ACs.
25 2095                                                               ; NOTE:  The only other instruction that currently
25 2096                                                               ;        uses DF-FROM-AR is the revised BLT (see
25 2097 05304 010731170300226654161560254705560000             IFRQ  D[DF-FROM-AR MEM] DEST[MEMSTO] JUMP[SEOI] $
25 2098 05305 710731170000060554161623654374160000     POP3:   D[AR] DEST[MA] SHORT $
25 2099                                                          ;Stack ptr points into ACs.
25 2100 05306 010331170000060554161503454314160000             ALU[MEMAC] DEST[HOLD] NORM $
25 2101 05307 710731170000060554045624257714160000             D[IR] MASK[18.] DEST[MA] NORM POPJ $
25m2102                                                   .pair
25m2102                                                  [:. \ 2 + .
25 2103 05310 010731170000000001416162225431456000       ]POP2:        PUSHJ[PDLO] $
25 2104 05311 710731110000126134045622257714560000             DFRQ D[IR] MASK[18.] DEST[MA] MA<20 PUSHJ[POP3] $
25 2105                                                               ; Get back original MA from IR, check for pdl in ACs.
25 2106 05312 540733170300060554161623054605360000             D[DF-FROM-AR MEM] DEST[MEMAC] NEOI $
25 2107
25 2108
25m2109                                                   .opcode[260]   [xlist
25 2109                                                    list  ]                                                    ;PUSHJ
25 2110 06540 010730170000000001404546025631456000             D[PC] MASK[18.] DEST[Q] LD-PC JUMP[PUSHJ1] $
25 2111                                                           ;Put old PC in Q, load PC from MA.
25 2112 06541 010731170000224334161620254314560000             JUMP[JMPAC] $
25m2113                                                   .reloc
25m2113                                                  [.USE[NORMAL]
25m2113                                                  [ xlist
25 2114 05313 010631370000060554141503646312160000       ]PUSHJ1:      DFRQ D[PC-FLAGS]
25 2114                                                  MASK[LEFT] ALU[DORQ] DEST[HOLD AR] $
25 2115                                                         ;Fetch target instr., put PC and flags into HOLD and AR, for
25 2115                                                  storing.
25 2116 05314 710503120037126014003641255714560000             D[1,,1] ALU[AC+D] DEST[AC MA] STRT-WRT COND[EA<20] LBJUMP[PUSH3]
25 2116                                                  $
25 2117                                                         ;Increment pointer, start store, test for PDL in the AC's.
25 2118                                                         ;The EOI at PUSH3 dispatches to the target instr; we loaded PC
25 2118                                                  above.
25 2119
25 2120
25m2121                                                   .opcode[263]   [xlist
25 2121                                                    list  ]                                                    ;POPJ
25 2122 06546 710330122000000001416162125431456000             ALU[AC] DEST[Q MA] T300 -EA<20 LBJUMP[POPJ1] $
25 2123 06547 710330122000000001416162125431456000             ALU[AC] DEST[Q MA] T300 -EA<20 LBJUMP[POPJ1] $
25 2124                                                           ;Get loc. of top of stack, check for stack in AC's.
25m2125                                                   .reloc
25m2125                                                  [.USE[NORMAL]
25m2125                                                  [ xlist
25m2126                                                    list ]                                    ]  .pair
25m2126                                                  [:. \ 2 + .
25 2127 05316 710331170000000001416162005431456000       ]POPJ1:       ALU[MEMAC] DEST[MA]  JUMP[POPJ3] $
25 2128                                                          ;Stack is in ACs... get return addr. off stack into MA.
25m2129                                                   .odd
25m2129                                                  [:. + 1 - (. \ 2)
25 2130                                                    ]
25 2131 05317 110601030006000001414162224721256000     POPJ0:  DFRQ MA-FROM-MEM D[MASK 60] ALU[D+Q]  C550 COND[CRY0]
25 2131                                                  PUSHJ[POPJ2] $
25 2132                                                         ;Get return adr into MA, test whether pdl ptr will underflow.
25 2132                                                  If
25 2133                                                         ; not, do PUSHJ to provide trap fix-up adr for MAPTRP & ECCTRP.
25 2133
25 2134 05320 010730170020000014161621646314560000     POPJTR: SUPPRESS-FETCH-TRAPS D[PC-FLAGS] DEST[Q] LBJUMPLC[POPJOV] $
25 2135                                                         ;We get here either from previous uinst (if PDLOV), in which
25 2135                                                  case
25 2136                                                         ; Last Cond is FALSE, or from an ECC Trap, in which case L.C. is
25 2136                                                  TRUE.
25 2137                                                         ;In case of both PDLOV and ECC error, suppress the ECC trap for
25 2137                                                  now.
25 2138 05321 710331170000021237416162225431456000             ALU[AC] DEST[MA] NORM PUSHJ[MAP-RET] $
25 2139                                                         ;MAPTRP will POPJLB to here. Give it a good MA & an adr to
```

```
25 2144                                                         ;Here to check for underflow when stack is in AC's.
25 2145                                                 POPJZ:  TRP-CTL[TRAP-FIX] D[1,,1] ALU[AC-D] DEST[AC LD-PC]
25 2146 05324 551513112137276554003463255613160000              COND[-MA<20]  ABORT-ADR-06 JDSP-SPECIAL-ABORT $
25 2147                                                                 ;Trap to 06 if jumping into AC's, else dispatch to
25 2147                                                          target.
25m2148                                                           .pair
25m2148                                                          [:. \ 2 + .
25 2149 05326 010630170000012636640242025571457000               ]POPJOV:     D[CONST 1] ROT[35. - 9.] ALU[DORQ] DEST[PC-FLAGS Q]
25 2149                                                          JUMP[POPJ0] $
25 2150                                                                 ;The ptr is going to underflow, so set the PDLOV flag.
25 2151 05327 710331170000223554161622254314560000             POPJEC: ALU[AC] DEST[MA] NORM PUSHJ[ECC-RET] $
25 2152                                                                 ;ECC traps wind up here. Give them a good MA.
25 2153 05330 710731170000126514161620254715560000                     D[MEM] DEST[MA] JUMP[POPJZ] $
25 2154                                                                 ;We return here after the ECC Trap is finished getting good
25 2154                                                          data.
25 2155
25 2156
25 2157                                                         ; Subr to set overflow bit
25 2158 05331 010730170000060554161623646114160000             PDLO:   D[PC-FLAGS] DEST[Q] C600 $
25 2159                                                                 ;Set bit 9 in flags. This will cause a PC+TRAP after next
25 2159                                                          dispatch,
25 2160                                                                 ; which will get us to POVTRP (below).
25 2161 05332 010631170000060546402424255514170000                     D[CONST 1] ROT[35. - 9.] ALU[DORQ] DEST[PC-FLAGS] C600 POPJ $
25 2162
25 2163
25 2164 05333 610721170041060554161463656314160000             PDL-TRAP:CLR-PC-TRAP-FLAGS D[PC] ALU[D-1] DEST[PC] NORM  $
25 2165                                                                  ;Here from PCTRAP  when bit 9 of PC flags is on at EOI, or
25 2166                                                                  ;  from an IDISP aborted due to PDLOV (ABORT-ADR-10).
25 2167                                                                  ;Make PC point to instr. following the one that set bit 9.
25 2168 05334 010730170000000005416162367031416000                     D[AMEM-ABS APR-STATUS] DEST[Q] NORM $
25 2169 05335 010631170000000004400354365571416000                     D[CONST 1] ROT[35. - 19.] ALU[DorQ] DEST[AMEM-ABS APR-STATUS]
25 2169                                                          NORM $
25 2170                                                                  ;Set the PDL OV flag in the CONI APR, status.
25 2171 05336 010641370000203434007620247314560000                     D[MASK 3] ALU[D&Q] DEST[AR] NORM JUMP[PIGEN] $
25 2172                                                                  ;Go request a PI on the APR channel.
25 2173
```

```
26 2174
26m2175                                          .opcode[264]   [xlist
26 2175                                           list  ]                                                    ;JSR
26 2176 06550 010730170000000001414162024631456000          D[PC-FLAGS] MASK[LEFT] DEST[Q] NORM JUMP[JSR1] $
26 2177                                               ;Save the PC and reload it with (MA)+1.
26 2178 06551 010731170000224334161620254314560000          JUMP[JMPAC] $
26 2179
26m2180                                          .reloc
26m2180                                         [.USE[NORMAL]
26m2180                                         [ xlist
26 2181 05337 010631170000060554045563656314160000           ]JSR1:       D[PC] MASK[18.]
26 2181                                         ALU[DORQ] DEST[MEMSTO]  NORM $
26 2182                                               ;Store PC with flags
26 2183 05340 711701170043060554161463656714160000          CLR-HALF D[MA] ALU[D+1] DEST[MA PC] NORM $
26 2184 05341 550731170000027755416162325421216000          JDSP $
26 2185
26m2186                                          .opcode[265]   [xlist
26 2186                                           list  ]                                                    ;JSP
26 2187 06552 010730170000000001404562025631456000          D[PC] MASK[18.] DEST[Q] NORM JUMP[JSP1] $
26 2188 06553 010731170000224334161620254314560000          JUMP[JMPAC] $
26m2189                                           .reloc
26m2189                                         [.USE[NORMAL]
26m2189                                         [ xlist
26 2190 05342 550633170043277554141623246212160000           ]JSP1:       D[PC-FLAGS]
26 2190                                         MASK[LEFT] ALU[DORQ] DEST[AC] CLR-HALF TRUE JDSP $
26 2191
26 2192
26 2193
26m2194                                          .opcode[266]   [xlist
26 2194                                           list  ]                                                    ;JSA
26 2195 06554 010732170000000001404556025631456000          D[PC] MASK[18.] DEST[0-AC MEMSTO] NORM JUMP[JSA1] $
26 2196                                               ;Begin store of AC, place PC in AC
26 2197 06555 010731170000224334161620254314560000          JUMP[JMPAC] $
26 2198
26m2199                                          .reloc
26m2199                                         [.USE[NORMAL]
26m2199                                         [ xlist
26 2200 05343 010533170000060544541623656714160000           ]JSA1:       D[MA] ROT[18.]
26 2200                                         MASK[LEFT] ALU[DORAC] DEST[AC] $
26 2201                                               ;Place pointer in left half of AC, complete store of old AC.
26 2202 05344 711701170000060554161623656774160000          D[MA] ALU[D+1] DEST[MA] SHORT $
26 2203                                               ;Now branch to E+1.
26 2204 05345 550731170000027755416162325421216000          JDSP $
26 2205
26m2206                                          .opcode[267]   [xlist
26 2206                                           list  ]                                                    ;JRA
26 2207 06556 010331370000000001416162025431456000          ALU[AC] DEST[AR] NORM JUMP[JRA1] $
26 2208 06557 010731170000224334161620254314560000          JUMP[JMPAC] $
26 2209
26m2210                                          .reloc
26m2210                                         [.USE[NORMAL]
26m2210                                         [ xlist
26 2211 05346 710731120000000000444562125431456000           ]JRA1:       D[AR] ROT[18.]
26 2211                                         MASK[18.] DEST[MA] NORM EA<20 LBJUMP[JRA2] $
26m2212                                          .even
26m2212                                         [:. \ 2 + .
26 2213 05350 030733170000012725416162025471356000           ]JRA2:       DF/WT D[MEM] DEST[AC] JUMP[. + 2] $
26 2214 05351 010433170000060554161623414314160000          ACSEL[MA,AC] ALU[A] DEST[B] NORM $
26 2215 05352 710731170000060554045623657714160000          D[IR] MASK[18.] DEST[MA] NORM $
26 2216 05353 550731170000027755416162325421216000          JDSP $
26 2217
```

```
27 2218
27m2219                                                  .opcode[270]
27m2219                                                 [xlist
27 2220 06560 550503170044060554161623254603160000       list  ]      DF/IF D[MEM] ALU[D+AC] DEST[AC] SET-PC-FLAGS NEOI $
27 2220                                                 ;ADD
27 2221 06561 540103170044060554161623014204160000             ACSEL[MA,AC] ALU[A+B]  DEST[B] SET-PC-FLAGS NEOI $
27 2222
27 2223 06562 540503170044060554045623257604160000             D[IR] MASK[18.] ALU[D+AC] DEST[AC] SET-PC-FLAGS NEOI $  ;ADDI
27 2224 06563 540503170044060554045623257604160000             D[IR] MASK[18.] ALU[D+AC] DEST[AC] SET-PC-FLAGS NEOI $
27 2225
27 2226                                                         DF/IF R-M-W IFRQ D[MEM] ALU[D+AC] DEST[MEMSTO] SET-PC-FLAGS
27 2227 06564 030501171044226654161560254703560000                                                      JUMP[SEOI] $
27 2227                                                 ;ADDM
27 2228 06565 540103170044060554161623054204160000             ACSEL[AC,MA] ALU[A+B]  DEST[B] SET-PC-FLAGS NEOI $
27 2229
27 2230                                                         DF/IF R-M-W IFRQ D[MEM] ALU[D+AC] DEST[MEMSTO AR] SET-PC-FLAGS
27 2231 06566 030501371044227014161560254703560000                                                      JUMP[ASEOI] $
27 2231                                                 ;ADDB
27 2232 06567 010103370044227014161620054304560000             IFRQ ACSEL[AC,MA] ALU[A+B] DEST[B AR] SET-PC-FLAGS JUMP[ASEOI] $
27 2232
27 2233
27m2234                                                  .opcode[274]
27m2234                                                 [xlist
27 2235 06570 561513170044060554161623254603160000       list  ]      DF/IF D[MEM] ALU[AC-D] DEST[AC] SET-PC-FLAGS NEOI $
27 2235                                                 ;SUB
27 2236 06571 541113170044060554161623014204160000             ACSEL[MA,AC] ALU[B-A]  DEST[B] SET-PC-FLAGS NEOI $
27 2237
27 2238 06572 541513170044060554045623257604160000             D[IR] MASK[18.] ALU[AC-D] DEST[AC] SET-PC-FLAGS NEOI $  ;SUBI
27 2239 06573 541513170044060554045623257604160000             D[IR] MASK[18.] ALU[AC-D] DEST[AC] SET-PC-FLAGS NEOI $
27 2240
27 2241                                                         DF/IF R-M-W IFRQ D[MEM] ALU[AC-D] DEST[MEMSTO] SET-PC-FLAGS
27 2242 06574 031511171044226654161560254703560000                                                      JUMP[SEOI] $
27 2242                                                 ;SUBM
27 2243 06575 541123170044060554161623054204160000             ACSEL[AC,MA] ALU[A-B]  DEST[B] SET-PC-FLAGS NEOI $
27 2244
27 2245                                                         DF/IF R-M-W IFRQ D[MEM] ALU[AC-D] DEST[MEMSTO AR] SET-PC-FLAGS
27 2246 06576 031511371044227014161560254703560000                                                      JUMP[ASEOI] $
27 2246                                                 ;SUBB
27 2247 06577 011123370044227014161620054304560000             IFRQ ACSEL[AC,MA] ALU[A-B] DEST[B AR] SET-PC-FLAGS JUMP[ASEOI] $
27 2247
27 2248
```

```
28 2249
28 2250                                                 ;;; JUMPx group
28 2251
28m2252                                                   .opcode[320]  [xlist
28 2252                                                    list  ]                                                    ;JUMP
28 2253 06640 540731170000060554161623254204160000            NEOI $
28 2254 06641 540731170000060554161623254204160000            NEOI $
28 2255
28 2256 06642 550331004000027755416162325421216000            ALU[AC] COND[OBUS<0] JDSP $
28 2257 06643 010731170000224334161620254314560000            JUMP [JMPAC] $
28 2258
28 2259 06644 550331002000027755416162325421216000            ALU[AC] COND[OBUS=0] JDSP $
28 2260 06645 010731170000224334161620254314560000            JUMP [JMPAC] $
28 2261
28 2262 06646 550301010000027755416162325421216000            ALU[AC+0] COND[-OBUS>0] JDSP $
28 2263 06647 010731170000224334161620254314560000            JUMP [JMPAC] $
28 2264
28 2265 06650 550731170000027755416162325421216000            JDSP $
28 2266 06651 010731170000224334161620254314560000            JUMP [JMPAC] $
28 2267
28 2268 06652 550331006000027755416162325421216000            ALU[AC] COND[-OBUS<0] JDSP $
28 2269 06653 010731170000224334161620254314560000            JUMP [JMPAC] $
28 2270
28 2271 06654 550331000000027755416162325421216000            ALU[AC] COND[-OBUS=0] JDSP $
28 2272 06655 010731170000224334161620254314560000            JUMP [JMPAC] $
28 2273
28 2274 06656 550301012000027755416162325421216000            ALU[AC+0] COND[OBUS>0] JDSP $
28 2275 06657 010731170000224334161620254314560000            JUMP [JMPAC] $
28 2276
28 2277
28m2278                                                   .opcode[330]  [xlist
28 2278                                                    list  ]                                                    ;SKIP
28 2279 06660 560731170000060554160163254603160000            DF/IF D[MEM]            COND-AC-STO NEOI $
28 2280 06661 540431170000060554160163014204160000            ACSEL[MA,AC] ALU[A]     COND-AC-STO NEOI $
28 2281
28 2282 06662 560731006000073554160163254603160000            DF/IF D[MEM]            COND-AC-STO CONDSKP[ALU<0] $
28 2282                                                 ;SKIPL
28 2283 06663 540431006000073554160163014204160000            ACSEL[MA,AC] ALU[A]     COND-AC-STO CONDSKP[ALU<0] $
28 2284
28 2285 06664 560731000000073554160163254603160000            DF/IF D[MEM]            COND-AC-STO CONDSKP[ALU=0] $
28 2285                                                 ;SKIPE
28 2286 06665 540431000000073554160163014204160000            ACSEL[MA,AC] ALU[A]     COND-AC-STO CONDSKP[ALU=0] $
28 2287
28 2288 06666 560701012000073554160163254603160000            DF/IF D[MEM] ALU[D+0]   COND-AC-STO CONDSKP[-ALU>0] $
28 2288                                                 ;SKIPLE
28 2289 06667 540401012000073554160163014204160000            ACSEL[MA,AC] ALU[A+0]   COND-AC-STO CONDSKP[-ALU>0] $
28 2290
28m2291                                                   .opcode[334]  [xlist
28 2291                                                    list  ]                                                    ;SKIPA
28 2292 06670 560731172000073554160163254603160000            DF/IF D[MEM]            COND-AC-STO CONDSKP[ALWAYS] $
28 2293 06671 540431172000073554160163014204160000            ACSEL[MA,AC] ALU[A]     COND-AC-STO CONDSKP[ALWAYS] $
28 2294
28 2294                                                 ;SKIPGE
28 2295 06672 560731004000073554160163254603160000            DF/IF D[MEM]            COND-AC-STO CONDSKP[-ALU<0] $
28 2296 06673 540431004000073554160163014204160000            ACSEL[MA,AC] ALU[A]     COND-AC-STO CONDSKP[-ALU<0] $
28 2297
28 2297                                                 ;SKIPN
28 2298 06674 560731002000073554160163254603160000            DF/IF D[MEM]            COND-AC-STO CONDSKP[-ALU=0] $
28 2299 06675 540431002000073554160163014204160000            ACSEL[MA,AC] ALU[A]     COND-AC-STO CONDSKP[-ALU=0] $
28 2300
28 2301 06676 560701010000073554160163254603160000            DF/IF D[MEM] ALU[D+0]   COND-AC-STO CONDSKP[ALU>0] $
28 2301                                                 ;SKIPG
28 2302 06677 540401010000073554160163014204160000            ACSEL[MA,AC] ALU[A+0]   COND-AC-STO CONDSKP[ALU>0] $
28 2303
28m2304                                                   .reloc
28m2304                                                 [.USE[NORMAL]
28m2304                                                 [ xlist
28 2305                                                    list ]                                  ]
28m2306                                                   .pair
28m2306                                                 [:. \ 2 + .
28 2307 05354 540731170000060554160163254254160000              ]CSSKP1:      D[AR] COND-AC-STO EOI $
28 2308                                                                       ;Here to finish AOS/SOS type if NOT SKIPPING.
28 2309 05355 540731172000073554160163254254160000                    D[AR] COND-AC-STO EOI ID-COND[1] ABORT-ADR-00 FALSE $
28 2310                                                                       ;AOS/SOS type comes here if SKIPPING. Do an
28 2311                                                                       ; aborted skip-type dispatch.
28 2312
```

```
29 2313
29 2314                                                 ;;; AOJ group
29 2315
29m2316                                                  .opcode[340]   [xlist
29 2316                                                  list  ]                                                 ;AOJ
29 2317 06700 541303170044406055416162325420416000           ALU[AC+1] DEST[AC] SET-PC-FLAGS NEOI $
29 2318 06701 541303170044406055416162325420416000           ALU[AC+1] DEST[AC] SET-PC-FLAGS NEOI $
29 2319
29 2320 06702 551303004044277554161623254212160000           ALU[AC+1] DEST[AC] SET-PC-FLAGS  COND[OBUS<0] JDSP $
29 2321 06703 010731170000224334161620254314560000           JUMP [JMPAC] $
29 2322
29 2323 06704 551303002044277554161623254212160000           ALU[AC+1] DEST[AC] SET-PC-FLAGS COND[OBUS=0] JDSP $
29 2324 06705 010731170000224334161620254314560000           JUMP [JMPAC] $
29 2325
29 2326 06706 551303010044277554161623254212160000           ALU[AC+1] DEST[AC] SET-PC-FLAGS COND[-OBUS>0] JDSP $
29 2327 06707 010731170000224334161620254314560000           JUMP [JMPAC] $
29 2328
29 2329 06710 551303170044277554161623254212160000           ALU[AC+1] DEST[AC] SET-PC-FLAGS JDSP $
29 2330 06711 010731170000224334161620254314560000           JUMP [JMPAC] $
29 2331
29 2332 06712 551303006044277554161623254212160000           ALU[AC+1] DEST[AC] SET-PC-FLAGS COND[-OBUS<0] JDSP $
29 2333 06713 010731170000224334161620254314560000           JUMP [JMPAC] $
29 2334
29 2335 06714 551303000044277554161623254212160000           ALU[AC+1] DEST[AC] SET-PC-FLAGS COND[-OBUS=0] JDSP $
29 2336 06715 010731170000224334161620254314560000           JUMP [JMPAC] $
29 2337
29 2338 06716 551303012044277554161623254212160000           ALU[AC+1] DEST[AC] SET-PC-FLAGS COND[OBUS>0] JDSP $
29 2339 06717 010731170000224334161620254314560000           JUMP [JMPAC] $
29 2340
29m2341                                                  .opcode[350]  [xlist
29 2341                                                  list  ]                                                 ;AOS
29 2342
29 2343                                                       DF/IF R-M-W D[MEM] ALU[D+1] DEST[MEMSTO AR]    TRUE SET-PC-FLAGS
29 2343
29 2344 06720 031701371044127314161560254703560000                                                             JUMP[CSSKP1]  $
29 2344
29 2345                                                       IFRQ ACSEL[MA] ALU[B+1] DEST[B AR]      FALSE SET-PC-FLAGS
29 2346 06721 011303372044127314161621054304560000                                                             LBJUMP[CSSKP1] $
29 2346
29 2347
29 2348                                                       DF/IF R-M-W D[MEM] ALU[D+1] DEST[MEMSTO AR]  OBUS<0 SET-PC-FLAGS
29 2348
29 2349 06722 031701205044127314161561254703560000                                                             LBJUMP[CSSKP1] $
29 2349                                                ;AOSL
29 2350                                                       IFRQ ACSEL[MA] ALU[B+1] DEST[B AR]     OBUS<0 SET-PC-FLAGS
29 2351 06723 011303204044127314161621054304560000                                                             LBJUMP[CSSKP1] $
29 2351
29 2352
29 2353                                                       DF/IF R-M-W D[MEM] ALU[D+1] DEST[MEMSTO AR]  OBUS=0 SET-PC-FLAGS
29 2353
29 2354 06724 031701203044127314161561254703560000                                                             LBJUMP[CSSKP1] $
29 2354                                                ;AOSE
29 2355                                                       IFRQ ACSEL[MA] ALU[B+1] DEST[B AR]     OBUS=0 SET-PC-FLAGS
29 2356 06725 011303202044127314161621054304560000                                                             LBJUMP[CSSKP1] $
29 2356
29 2357
29 2358                                                       DF/IF R-M-W D[MEM] ALU[D+1] DEST[MEMSTO AR] -OBUS>0 SET-PC-FLAGS
29 2358
29 2359 06726 031701211044127314161561254703560000                                                             LBJUMP[CSSKP1] $
29 2359                                                ;AOSLE
29 2360                                                       IFRQ ACSEL[MA] ALU[B+1] DEST[B AR]    -OBUS>0 SET-PC-FLAGS
29 2361 06727 011303210044127314161621054304560000                                                             LBJUMP[CSSKP1] $
29 2361
29 2362
29m2363                                                  .opcode[354]  [xlist
29 2363                                                  list  ]
29 2364                                                       DF/IF R-M-W D[MEM] ALU[D+1] DEST[MEMSTO AR]    TRUE SET-PC-FLAGS
29 2364
29 2365 06730 031701371044127314161561254703560000                                                             LBJUMP[CSSKP1] $
29 2365                                                ;AOSA
29 2366                                                       IFRQ ACSEL[MA] ALU[B+1] DEST[B AR]       TRUE SET-PC-FLAGS
29 2367 06731 011303370044127314161621054304560000                                                             LBJUMP[CSSKP1] $
29 2367
29 2368
29 2368                                                       DF/IF R-M-W D[MEM] ALU[D+1] DEST[MEMSTO AR] -OBUS<0 SET-PC-FLAGS
29 2369
29 2370 06732 031701207044127314161561254703560000                                                             LBJUMP[CSSKP1] $
29 2370                                                ;AOSGE
29 2371                                                       IFRQ ACSEL[MA] ALU[B+1] DEST[B AR]    -OBUS<0 SET-PC-FLAGS
29 2372 06733 011303206044127314161621054304560000                                                             LBJUMP[CSSKP1] $
29 2372
29 2373
29 2374                                                       DF/IF R-M-W D[MEM] ALU[D+1] DEST[MEMSTO AR] -OBUS=0 SET-PC-FLAGS
29 2374
29 2375 06734 031701201044127314161561254703560000                                                             LBJUMP[CSSKP1] $
29 2375                                                ;AOSN
29 2376                                                       IFRQ ACSEL[MA] ALU[B+1] DEST[B AR]    -OBUS=0 SET-PC-FLAGS
29 2377 06735 011303200044127314161621054304560000                                                             LBJUMP[CSSKP1] $
29 2377
29 2378
29 2379                                                       DF/IF R-M-W D[MEM] ALU[D+1] DEST[MEMSTO AR]  OBUS>0 SET-PC-FLAGS
29 2379
29 2380 06736 031701213044127314161561254703560000                                                             LBJUMP[CSSKP1] $
29 2380                                                ;AOSG
29 2381                                                       IFRQ ACSEL[MA] ALU[B+1] DEST[B AR]     OBUS>0 SET-PC-FLAGS
29 2382 06737 011303212044127314161621054304560000                                                             LBJUMP[CSSKP1] $
29 2382
29 2383
29 2384                                                 ;SOJx series
29 2385
29m2386                                                  .opcode[360]   [xlist
29 2386                                                  list  ]                                                 ;SOJ
29 2387 06740 540313170044060554161623254204160000           ALU[AC-1] DEST[AC] SET-PC-FLAGS NEOI $
29 2388 06741 540313170044060554161623254204160000           ALU[AC-1] DEST[AC] SET-PC-FLAGS NEOI $
29 2389
29 2390 06742 550313004044277554161623254212160000           ALU[AC-1] DEST[AC] SET-PC-FLAGS  COND[OBUS<0] JDSP $
29 2391 06743 010731170000224334161620254314560000           JUMP [JMPAC] $
29 2392
29 2393 06744 550313002044277554161623254212160000           ALU[AC-1] DEST[AC] SET-PC-FLAGS COND[OBUS=0] JDSP $
```

```
29 2399 06750 550313170044277554161623254212160000            ALU[AC-1] DEST[AC] SET-PC-FLAGS JDSP $
29 2400 06751 010731170000224334161620254314560000            JUMP [JMPAC] $
29 2401
29 2402 06752 550313006044277554161623254212160000            ALU[AC-1] DEST[AC] SET-PC-FLAGS COND[-OBUS<0] JDSP $
29 2403 06753 010731170000224334161620254314560000            JUMP [JMPAC] $
29 2404
29 2405 06754 550313000044277554161623254212160000            ALU[AC-1] DEST[AC] SET-PC-FLAGS COND[-OBUS=0] JDSP $
29 2406 06755 010731170000224334161620254314560000            JUMP [JMPAC] $
29 2407
29 2408 06756 550313012044277554161623254212160000            ALU[AC-1] DEST[AC] SET-PC-FLAGS COND[OBUS>0] JDSP $
29 2409 06757 010731170000224334161620254314560000            JUMP [JMPAC] $
29 2410
29m2411                                                      .opcode[370]  [xlist
29 2411                                                      list  ]                                          ;SOS
29 2412                                                           DF/IF R-M-W D[MEM] ALU[D-1] DEST[MEMSTO AR]    TRUE SET-PC-FLAGS
29 2412
29 2413 06760 030721371044127314161560254703560000                                                          JUMP[CSSKP1]  $
29 2413
29 2414                                                           IFRQ ACSEL[MA] ALU[B-1] DEST[B AR]      FALSE SET-PC-FLAGS
29 2415 06761 010313372044127314161621054304560000                                                          LBJUMP[CSSKP1] $
29 2415
29 2416
29 2417                                                           DF/IF R-M-W D[MEM] ALU[D-1] DEST[MEMSTO AR]  OBUS<0 SET-PC-FLAGS
29 2417
29 2418 06762 030721205044127314161561254703560000                                                          LBJUMP[CSSKP1] $
29 2418                                                      ;SOSL
29 2419                                                           IFRQ ACSEL[MA] ALU[B-1] DEST[B AR]     OBUS<0 SET-PC-FLAGS
29 2420 06763 010313204044127314161621054304560000                                                          LBJUMP[CSSKP1] $
29 2420
29 2421
29 2422                                                           DF/IF R-M-W D[MEM] ALU[D-1] DEST[MEMSTO AR]  OBUS=0 SET-PC-FLAGS
29 2422
29 2423 06764 030721203044127314161561254703560000                                                          LBJUMP[CSSKP1] $
29 2423                                                      ;SOSE
29 2424                                                           IFRQ ACSEL[MA] ALU[B-1] DEST[B AR]     OBUS=0 SET-PC-FLAGS
29 2425 06765 010313202044127314161621054304560000                                                          LBJUMP[CSSKP1] $
29 2425
29 2426
29 2427                                                           DF/IF R-M-W D[MEM] ALU[D-1] DEST[MEMSTO AR] -OBUS>0 SET-PC-FLAGS
29 2427
29 2428 06766 030721211044127314161561254703560000                                                          LBJUMP[CSSKP1] $
29 2428                                                      ;SOSLE
29 2429                                                           IFRQ ACSEL[MA] ALU[B-1] DEST[B AR]    -OBUS>0 SET-PC-FLAGS
29 2430 06767 010313210044127314161621054304560000                                                          LBJUMP[CSSKP1] $
29 2430
29 2431
29m2432                                                      .opcode[374]  [xlist
29 2432                                                      list  ]
29 2433                                                           DF/IF R-M-W D[MEM] ALU[D-1] DEST[MEMSTO AR]    TRUE SET-PC-FLAGS
29 2433
29 2434 06770 030721371044127314161561254703560000                                                          LBJUMP[CSSKP1] $
29 2434                                                      ;SOSA
29 2435                                                           IFRQ ACSEL[MA] ALU[B-1] DEST[B AR]       TRUE SET-PC-FLAGS
29 2436 06771 010313370044127314161621054304560000                                                          LBJUMP[CSSKP1] $
29 2436
29 2437
29 2437                                                           DF/IF R-M-W D[MEM] ALU[D-1] DEST[MEMSTO AR] -OBUS<0 SET-PC-FLAGS
29 2438
29 2439 06772 030721207044127314161561254703560000                                                          LBJUMP[CSSKP1] $
29 2439                                                      ;SOSGE
29 2440                                                           IFRQ ACSEL[MA] ALU[B-1] DEST[B AR]    -OBUS<0 SET-PC-FLAGS
29 2441 06773 010313206044127314161621054304560000                                                          LBJUMP[CSSKP1] $
29 2441
29 2442
29 2443                                                           DF/IF R-M-W D[MEM] ALU[D-1] DEST[MEMSTO AR] -OBUS=0 SET-PC-FLAGS
29 2443
29 2444 06774 030721201044127314161561254703560000                                                          LBJUMP[CSSKP1] $
29 2444                                                      ;SOSN
29 2445                                                           IFRQ ACSEL[MA] ALU[B-1] DEST[B AR]    -OBUS=0 SET-PC-FLAGS
29 2446 06775 010313200044127314161621054304560000                                                          LBJUMP[CSSKP1] $
29 2446
29 2447
29 2448                                                           DF/IF R-M-W D[MEM] ALU[D-1] DEST[MEMSTO AR]  OBUS>0 SET-PC-FLAGS
29 2448
29 2449 06776 030721213044127314161561254703560000                                                          LBJUMP[CSSKP1] $
29 2449                                                      ;SOSG
29 2450                                                           IFRQ ACSEL[MA] ALU[B-1] DEST[B AR]     OBUS>0 SET-PC-FLAGS
29 2451 06777 010313212044127314161621054304560000                                                          LBJUMP[CSSKP1] $
29 2451
29 2452
```

```
30 2453
30m2454                                                 .opcode[300] [xlist
30 2454                                                 list ]                                                  ;CAI
30 2455 06600 540731170000060554161623254204160000             NEOI $
30 2456 06601 540731170000060554161623254204160000             NEOI $
30 2457
30 2458 06602 541511136000073554045623257604160000             D[IR] MASK[18.] ALU[AC-D]        CONDSKP[ALU-SIGN] $
30 2458                                                 ;CAIL
30 2459 06603 541511136000073554045623257604160000             D[IR] MASK[18.] ALU[AC-D]        CONDSKP[ALU-SIGN] $
30 2460
30 2461 06604 541511000000073554045623257604160000             D[IR] MASK[18.] ALU[AC-D]        CONDSKP[ALU=0] $
30 2461                                                 ;CAIE
30 2462 06605 541511000000073554045623257604160000             D[IR] MASK[18.] ALU[AC-D]        CONDSKP[ALU=0] $
30 2463
30 2464 06606 541511012000073554045623257604160000             D[IR] MASK[18.] ALU[AC-D]        CONDSKP[-ALU>0] $
30 2464                                                 ;CAILE
30 2465 06607 541511012000073554045623257604160000             D[IR] MASK[18.] ALU[AC-D]        CONDSKP[-ALU>0] $
30 2466
30m2467                                                 .opcode[304] [xlist
30 2467                                                 list ]                                                  ;CAIA
30 2468 06610 540731172000073554161623254204160000             CONDSKP[ALWAYS] $
30 2469 06611 540731172000073554161623254204160000             CONDSKP[ALWAYS] $
30 2470
30 2470                                                 ;CAIGE
30 2471 06612 541511134000073554045623257604160000             D[IR] MASK[18.] ALU[AC-D]        CONDSKP[-ALU-SIGN] $
30 2472 06613 541511134000073554045623257604160000             D[IR] MASK[18.] ALU[AC-D]        CONDSKP[-ALU-SIGN] $
30 2473
30 2473                                                 ;CAIN
30 2474 06614 541511002000073554045623257604160000             D[IR] MASK[18.] ALU[AC-D]        CONDSKP[-ALU=0] $
30 2475 06615 541511002000073554045623257604160000             D[IR] MASK[18.] ALU[AC-D]        CONDSKP[-ALU=0] $
30 2476
30 2477 06616 541511010000073554045623257604160000             D[IR] MASK[18.] ALU[AC-D]        CONDSKP[ALU>0] $
30 2477                                                 ;CAIG
30 2478 06617 541511010000073554045623257604160000             D[IR] MASK[18.] ALU[AC-D]        CONDSKP[ALU>0] $
30 2479
30 2480
30 2481
30 2482
30m2483                                                 .opcode[310] [xlist
30 2483                                                 list ]                                                  ;CAM
30 2484 06620 540731170000060554161623254204160000             NEOI $
30 2485 06621 540731170000060554161623254204160000             NEOI $
30 2486
30 2487 06622 561511136000073554161623254603160000             DF/IF D[MEM] ALU[AC-D]  CONDSKP[ALU-SIGN] $
30 2487                                                 ;CAML
30 2488 06623 541121136000073554161623054204160000             ACSEL[AC,MA] ALU[A-B]   CONDSKP[ALU-SIGN] $
30 2489
30 2490 06624 561511000000073554161623254603160000             DF/IF D[MEM] ALU[AC-D]  CONDSKP[ALU=0] $
30 2490                                                 ;CAME
30 2491 06625 541121000000073554161623054204160000             ACSEL[AC,MA] ALU[A-B]   CONDSKP[ALU=0] $
30 2492
30 2493 06626 561511012000073554161623254603160000             DF/IF D[MEM] ALU[AC-D]  CONDSKP[-ALU>0] $
30 2493                                                 ;CAMLE
30 2494 06627 541121012000073554161623054204160000             ACSEL[AC,MA] ALU[A-B]   CONDSKP[-ALU>0] $
30 2495
30m2496                                                 .opcode[314] [xlist
30 2496                                                 list ]                                                  ;CAMA
30 2497 06630 540731172000073554161623254204160000             CONDSKP[ALWAYS] $
30 2498 06631 540731172000073554161623254204160000             CONDSKP[ALWAYS] $
30 2499
30 2499                                                 ;CAMGE
30 2500 06632 561511134000073554161623254603160000             DF/IF D[MEM] ALU[AC-D]  CONDSKP[-ALU-SIGN] $
30 2501 06633 541121134000073554161623054204160000             ACSEL[AC,MA] ALU[A-B]   CONDSKP[-ALU-SIGN] $
30 2502
30 2502                                                 ;CAMN
30 2503 06634 561511002000073554161623254603160000             DF/IF D[MEM] ALU[AC-D]  CONDSKP[-ALU=0] $
30 2504 06635 541121002000073554161623054204160000             ACSEL[AC,MA] ALU[A-B]   CONDSKP[-ALU=0] $
30 2505
30 2506 06636 561511010000073554161623254603160000             DF/IF D[MEM] ALU[AC-D]  CONDSKP[ALU>0] $
30 2506                                                 ;CAMG
30 2507 06637 541121010000073554161623054204160000             ACSEL[AC,MA] ALU[A-B]   CONDSKP[ALU>0] $
30 2508
30 2509
```

```
31 2510
31m2511
31m2511
31 2512 07000 540243170000060554161623254204160000          .opcode[400]
31 2512                                                     [xlist
31 2513 07001 540243170000060554161623014204160000           list ]      ALU[0] DEST[AC] NEOI $
31 2514                                                          ;SETZ
31 2515 07002 540243170000060554161623254204160000           ACSEL[MA,AC] ALU[0]  DEST[B] NEOI $
31 2515                                                     ;SETZI
31 2516 07003 540243170000060554161623014204160000           ALU[0] DEST[AC] NEOI $
31 2517                                                          ;SETZM
31 2518 07004 010241170000226554161560254304560000           ACSEL[MA,AC] ALU[0]  DEST[B] NEOI $
31 2518                                                     ;SETZB
31 2519 07005 540243170000060554161623054204160000           IFRQ ALU[0] DEST[MEMSTO] JUMP[SEOI] $
31 2520
31 2521 07006 010241370000227014161560254304560000          .opcode[404]
31 2521                                                     [xlist
31 2522 07007 010243370000227014161620054304560000           list ]      DF/IF D[MEM] ALU[D&AC] DEST[AC] NEOI $
31 2523                                                          ;AND
31 2524                                                           ACSEL[MA,AC] ALU[A&B]  DEST[B] NEOI $
31m2525
31m2525                                                           D[IR] MASK[18.] ALU[D&AC] DEST[AC] NEOI $
31 2526 07010 560543170000060554161623254603160000         ;ANDI
31 2526                                                           D[IR] MASK[18.] ALU[D&AC] DEST[AC] NEOI $
31 2527 07011 540143170000060554161623014204160000
31 2528                                                           DF/IF R-M-W IFRQ D[MEM] ALU[D&AC] DEST[MEMSTO] JUMP[SEOI] $
31 2529 07012 540543170000060554045623257604160000         ;ANDM
31 2529                                                           ACSEL[AC,MA] ALU[A&B]  DEST[B] NEOI $
31 2530 07013 540543170000060554045623257604160000
31 2531                                                           DF/IF R-M-W IFRQ D[MEM] ALU[D&AC] DEST[MEMSTO AR] JUMP[ASEOI] $
31 2532 07014 030541171000226554161560254703560000         ;ANDB
31 2532                                                           IFRQ ACSEL[AC,MA] ALU[A&B]  DEST[B AR] JUMP[ASEOI] $
31 2533 07015 540143170000060554161623054204160000
31 2534
31 2535 07016 030541371000227014161560254703560000          .opcode[410]
31 2535                                                     [xlist
31 2536 07017 010143370000227014161620054304560000           list ]      DFRQ  ALU[NOTAC] DEST[Q] JUMP[&CA1] $
31 2537                                                          ;ANDCA
31 2538                                                           ALU[NOTAC] DEST[Q] JUMP[&CA2] $
31m2539
31m2539                                                           ALU[NOTAC] DEST[Q] JUMP[&CAI] $
31 2540 07020 010370170000000001416162025431256000         ;ANDCAI
31 2540                                                           ALU[NOTAC] DEST[Q] JUMP[&CAI1] $
31 2541 07021 010370170000000001416162025431456000
31 2542                                                           DFRQ  ALU[NOTAC] DEST[Q] JUMP[&CAM1] $
31 2543 07022 010370170000000001416162025431456000         ;ANDCAM
31 2543                                                           ACSEL[AC,MA] ALU[-A&B]  DEST[B] NEOI $
31 2544 07023 010370170000000001416162025431456000
31 2545                                                           DFRQ  ALU[NOTAC] DEST[Q] JUMP[&CAB1] $
31 2546 07024 010370170000000001416162025431256000         ;ANDCAB
31 2546                                                           IFRQ ACSEL[AC,MA] ALU[-A&B]  DEST[B AR] JUMP[ASEOI] $
31 2547 07025 540153170000060554161623054204160000
31 2548
31 2549 07026 010370170000000001416162025431256000          .reloc
31 2549                                                     [.USE[NORMAL]
31 2550 07027 010153370000227014161620054304560000         [ xlist
31 2551                                                      list ]                                  ]&CA1:      D[MEM] ALU[D&Q]
31m2552                                                     DEST[AC] NEOI $
31m2552                                                     &CA2:   ACSEL[MA,AC] ALU[A&Q]  DEST[B] NEOI $
31m2552                                                     &CAI1:  D[IR] MASK[18.] ALU[D&Q] DEST[AC] NEOI $
31 2553 05356 540643170000060554161623254605160000         &CAM1:  IFRQ R-M-W D[MEM] ALU[D&Q] DEST[MEMSTO] JUMP[SEOI] $
31 2553                                                     &CAB1:  IFRQ R-M-W D[MEM] ALU[D&Q] DEST[MEMSTO AR] JUMP[ASEOI] $
31 2554 05357 540043170000060554161623014204160000
31 2555 05360 540643170000060554045623257604160000
31 2556 05361 010641171000226554161560254705560000
31 2557 05362 010641371000227014161560254705560000
31 2557
31 2558
31 2559
31m2560
31 2560                                                      .opcode[414]  [xlist
31 2561 07030 560733170000060554161623254603160000          list ]                                                  ;SETM
31 2562 07031 540433170000060554161623014204160000               DF/IF D[MEM] DEST[AC] NEOI $
31 2563                                                           ACSEL[MA,AC] ALU[A] DEST[B] NEOI $
31 2564 07032 540733170000060554045623257604160000
31 2564                                                           D[IR] MASK[18.] DEST[AC] NEOI $
31 2565 07033 540733170000060554045623257604160000         ;SETMI
31 2566                                                           D[IR] MASK[18.] DEST[AC] NEOI $
31 2567 07034 030731371000227034161560254703560000
31 2567                                                           DF/IF R-M-W D[MEM] DEST[MEMSTO AR] JUMP[CASEOI]  $
31 2568 07035 540431170000060554160163014204160000         ;setmm
31 2568                                                           ACSEL[MA,AC] ALU[A] COND-AC-STO NEOI $                  ;like
31 2569                                                     moves
31 2570 07036 560733170000060554161623254603160000
31 2570                                                           DF/IF D[MEM] DEST[AC] NEOI $
31 2571 07037 540433170000060554161623014204160000         ;SETMB
31 2572                                                           ACSEL[MA,AC] ALU[A] DEST[B] NEOI $
31 2573
31m2574
31m2574                                                      .opcode[420]
31 2575 07040 560553170000060554161623254603160000         [xlist
31 2575                                                       list ]      DF/IF D[MEM] ALU[-D&AC] DEST[AC] NEOI $
31 2576 07041 540153170000060554161623014204160000               ;ANDCM
31 2577                                                           ACSEL[MA,AC] ALU[-A&B]  DEST[B] NEOI $
31 2578 07042 540553170000060554045623257604160000
31 2578                                                           D[IR] MASK[18.] ALU[-D&AC] DEST[AC] NEOI $
31 2579 07043 540553170000060554045623257604160000         ;ANDCMI
31 2580                                                           D[IR] MASK[18.] ALU[-D&AC] DEST[AC] NEOI $
31 2581 07044 030551171000226554161560254703560000
31 2581                                                           DF/IF R-M-W IFRQ D[MEM] ALU[-D&AC] DEST[MEMSTO] JUMP[SEOI] $
31 2582 07045 010151370000000001416162001430456000         ;ANDCMM
31 2583                                                           IFRQ ACSEL[MA,AC] ALU[-A&B]  DEST[AR] JUMP[&CMM1] $
31 2584 07046 030551371000227014161560254703560000
31 2584                                                           DF/IF R-M-W IFRQ D[MEM] ALU[-D&AC] DEST[MEMSTO AR] JUMP[ASEOI] $
31 2585 07047 010153370000000001416162001430456000         ;ANDCMB
31 2586                                                           IFRQ ACSEL[MA,AC] ALU[-A&B]  DEST[B AR] JUMP[&CMM1] $
31m2587
31m2587                                                      .reloc
31m2587                                                     [.USE[NORMAL]
31 2588 05363 540733170000060554161623054254160000         [ xlist
31 2588                                                      list ]                                  ]&CMM1:     ACSEL[MA] D[AR]
```

```
SLOE  March 23, 1984 21:08:10  file DSK:F4INST.SLO  --  of -- F41NNF

31 2592 07050 540731170000060554161623254204160000          list  ]      NEOI $
31 2592                                                          ;SETA
31 2593 07051 540731170000060554161623254204160000               NEOI $
31 2594
31 2595 07052 540731170000060554161623254204160000               NEOI $
31 2595                                                     ;SETAI
31 2596 07053 540731170000060554161623254204160000               NEOI $
31 2597
31 2598 07054 010331170000226554161560254304560000               IFRQ ALU[AC] DEST[MEMSTO] NORM JUMP[SEOI] $
31 2598                                                     ;SETAM
31 2599 07055 540433170000060554161623054204160000               ACSEL[AC,MA] ALU[A] DEST[B] NEOI $
31 2600
31 2601 07056 010331170000226554161560254304560000               IFRQ ALU[AC] DEST[MEMSTO] NORM JUMP[SEOI] $
31 2601                                                     ;SETAB
31 2602 07057 540433170000060554161623054204160000               ACSEL[AC,MA] ALU[A] DEST[B] NEOI $
31 2603
31 2604
31m2605                                                     .opcode[430]
31m2605                                                     [xlist
31 2606 07060 560563170000060554161623254603160000          list  ]      DF/IF D[MEM] ALU[D#AC] DEST[AC] NEOI $
31 2606                                                          ;XOR
31 2607 07061 540163170000060554161623014204160000               ACSEL[MA,AC] ALU[A#B]  DEST[B] NEOI $
31 2608
31 2609 07062 540563170000060554045623257604160000               D[IR] MASK[18.] ALU[D#AC] DEST[AC] NEOI $
31 2609                                                     ;XORI
31 2610 07063 540563170000060554045623257604160000               D[IR] MASK[18.] ALU[D#AC] DEST[AC] NEOI $
31 2611
31 2612 07064 030561171000226554161560254703560000               DF/IF R-M-W IFRQ D[MEM] ALU[D#AC] DEST[MEMSTO] JUMP[SEOI] $
31 2612                                                          ;XORM
31 2613 07065 540163170000060554161623054204160000               ACSEL[AC,MA] ALU[A#B]  DEST[B] NEOI $
31 2614
31 2615 07066 030561371000227014161560254703560000               DF/IF R-M-W IFRQ D[MEM] ALU[D#AC] DEST[MEMSTO AR] JUMP[ASEOI] $
31 2615                                                     ;XORB
31 2616 07067 010163370000227014161620054304560000               IFRQ ACSEL[AC,MA] ALU[A#B]  DEST[B AR] JUMP[ASEOI] $
31 2617
31 2618
31m2619                                                     .opcode[434]
31m2619                                                     [xlist
31 2620 07070 560533170000060554161623254603160000          list  ]      DF/IF D[MEM] ALU[DORAC] DEST[AC] NEOI $
31 2620                                                          ;IOR
31 2621 07071 540133170000060554161623014204160000               ACSEL[MA,AC] ALU[AORB]  DEST[B] NEOI $
31 2622
31 2623 07072 540533170000060554045623257604160000               D[IR] MASK[18.] ALU[DORAC] DEST[AC] NEOI $
31 2623                                                     ;IORI
31 2624 07073 540533170000060554045623257604160000               D[IR] MASK[18.] ALU[DORAC] DEST[AC] NEOI $
31 2625
31 2626 07074 030531171000226554161560254703560000               DF/IF R-M-W IFRQ D[MEM] ALU[DORAC] DEST[MEMSTO] JUMP[SEOI] $
31 2626                                                          ;IORM
31 2627 07075 540133170000060554161623054204160000               ACSEL[AC,MA] ALU[AORB]  DEST[B] NEOI $
31 2628
31 2629 07076 030531371000227014161560254703560000               DF/IF R-M-W IFRQ D[MEM] ALU[DORAC] DEST[MEMSTO AR] JUMP[ASEOI] $
31 2629                                                          ;IORB
31 2630 07077 010133370000227014161620054304560000               IFRQ ACSEL[AC,MA] ALU[AORB]  DEST[B AR] JUMP[ASEOI] $
31 2631
31 2632
31m2633                                                     .opcode[440]
31m2633                                                     [xlist
31 2634 07100 030531370000000014161620254713560000          list  ]      DF/WT D[MEM] ALU[DORAC] DEST[AR] JUMP[STCM] $
31 2634                                                          ;ANDCB
31 2635 07101 010131370000000014161620014314560000               ACSEL[MA,AC] ALU[AORB]  DEST[AR] JUMP[STCM] $
31 2636
31 2637 07102 010531370000000014045620257714560000               D[IR] MASK[18.] ALU[DORAC] DEST[AR] JUMP[STCM] $
31 2637                                                     ;ANDCBI
31 2638 07103 010531370000000014045620257714560000               D[IR] MASK[18.] ALU[DORAC] DEST[AR] JUMP[STCM] $
31 2639
31 2640 07104 030531371000000014161620254703560000               DF/WT R-M-W IFRQ D[MEM] ALU[DORAC] DEST[AR] JUMP[STCMM] $
31 2640                                                          ;ANDCBM
31 2641 07105 010131370000000014161620054314560000               ACSEL[AC,MA] ALU[AORB]  DEST[AR] JUMP[STCMMA] $
31 2642
31 2643 07106 030531371000000014161620254703560000               DF/WT R-M-W IFRQ D[MEM] ALU[DORAC] DEST[AR] JUMP[STCMB] $
31 2643                                                          ;ANDCBB
31 2644 07107 010131370000000014161620054304560000               IFRQ ACSEL[AC,MA] ALU[AORB]  DEST[AR] JUMP[STCMBA] $
31 2645
31m2646                                                       .reloc
31m2646                                                     [.USE[NORMAL]
31m2646                                                     [ xlist
31 2647 05364 540773170000060554161623254204160000          list ]                                  ]STCM:       D[AR] ALU[NOTD]
31 2647                                                     DEST[AC] NEOI $
31 2648 05365 010771170000226554161560254304560000     STCMM:  IFRQ D[AR] ALU[NOTD] DEST[MEMSTO] JUMP[SEOI] $
31 2649 05366 540773170000060554161623054204160000     STCMMA: D[AR] ALU[NOTD] ACSEL[MA] DEST[B] NEOI $
31 2650 05367 010771370000227014161560254304560000     STCMB:  IFRQ D[AR] ALU[NOTD] DEST[MEMSTO AR] JUMP[ASEOI] $
31 2651 05370 010773370000227014161620054304560000     STCMBA: IFRQ D[AR] ALU[NOTD] ACSEL[MA] DEST[B AR] JUMP[ASEOI] $
31 2652
31m2653                                                     .opcode[444]
31m2653                                                     [xlist
31 2654 07110 560573170000060554161623254603160000          list  ]      DF/IF D[MEM] ALU[D/#AC] DEST[AC] NEOI $
31 2654                                                          ;EQV
31 2655 07111 540173170000060554161623014204160000               ACSEL[MA,AC] ALU[A/#B]  DEST[B] NEOI $
31 2656
31 2657 07112 540573170000060554045623257604160000               D[IR] MASK[18.] ALU[D/#AC] DEST[AC] NEOI $
31 2657                                                     ;EQVI
31 2658 07113 540573170000060554045623257604160000               D[IR] MASK[18.] ALU[D/#AC] DEST[AC] NEOI $
31 2659
31 2660 07114 030571171000226554161560254703560000               DF/IF R-M-W IFRQ D[MEM] ALU[D/#AC] DEST[MEMSTO] JUMP[SEOI] $
31 2660                                                          ;EQVM
31 2661 07115 540173170000060554161623054204160000               ACSEL[AC,MA] ALU[A/#B]  DEST[B] NEOI $
31 2662
31 2663 07116 030571371000227014161560254703560000               DF/IF R-M-W IFRQ D[MEM] ALU[D/#AC] DEST[MEMSTO AR] JUMP[ASEOI] $
31 2663                                                          ;EQVB
31 2664 07117 010173370000227014161620054304560000               IFRQ ACSEL[AC,MA] ALU[A/#B]  DEST[B AR] JUMP[ASEOI] $
31 2665
31 2666
31m2667                                                     .opcode[450]
31m2667                                                     [xlist
31 2668 07120 540373170000060554161623254204160000          list  ]      ALU[NOTAC] DEST[AC] NEOI $
31 2668                                                          ;SETCA
31 2669 07121 540373170000060554161623254204160000               ALU[NOTAC] DEST[AC] NEOI $
31 2670
31 2671 07122 540373170000060554161623254204160000               ALU[NOTAC] DEST[AC] NEOI $
31 2671                                                     ;SETCAI
```

```
31 2676
31 2677 07126 010471370000227014161560054304560000
31 2678 07127 010473370000227014161620054304560000
31 2679
31 2680
31m2681
31m2681
31 2682 07130 030551370000127514161620254713560000
31 2683
31 2684 07131 010151370000127514161620014314560000
31 2685
31 2686 07132 010551370000127514045620257714560000
31 2687
31 2688 07133 010551370000127514045620257714560000
31 2689
31 2690 07134 030551371000127534161620254713560000
31 2691
31 2692 07135 010151370000127554161620014314560000
31 2693
31 2694 07136 030551371000127574161620254713560000
31 2695
31m2696 07137 010151370000127614161620014314560000
31m2696
31 2697 07140 560773170000006055416162325460316000
31 2698
31 2699 07141 540473170000060554161623014204160000
31 2700
31 2701 07142 540773170000060554045623257604160000
31 2702
31 2703 07143 540773170000060554045623257604160000
31 2704
31 2705 07144 030771171000226654161560254703560000
31 2706
31 2707 07145 540373170000060554161623054204160000
31 2708
31 2709 07146 030773171000226654161560254703560000
31m2710
31m2710 07147 010373370000227014161620054304560000
31 2711
31 2712 07150 030771370000000001416162025471356000
31 2713
31 2714 07151 010371370000000001416162005431456000
31 2715
31 2716 07152 010771370000000001404562025771456000
31 2717
31 2718 07153 010771370000000001404562025771456000
31 2719
31 2720 07154 030771371000000001416162025471356000
31 2721
31 2722 07155 010371370000000001416162005431456000
31m2723
31m2723 07156 030771371000000001416162025471356000
31m2723
31 2724 05371 540533170000060554161623254204160000
31 2725
31 2726 05372 010531170000226654161560254304560000
31 2727 05373 540533170000060554161623054204160000
31 2728 05374 010531370000227014161560254304560000
31 2729 05375 010533370000227014161620054304560000
31 2730
31m2731
31m2731
31 2732 07160 030541370000127514161620254713560000
31 2733
31 2734 07161 010141370000127514161620014314560000
31 2735
31 2736 07162 010541370000127514045620257714560000
31 2737
31 2738 07163 010541370000127514045620257714560000
31 2739
31 2740 07164 030541371000127534161620254703560000
31 2741
31 2742 07165 010141370000127554161620054314560000
31 2743
31 2744 07166 030541371000127574161620254703560000
31m2745
31m2745 07167 010141370000127614161620054304560000
31 2746
31 2747 07170 540723170000060554001623255604160000
31 2748
31 2749 07171 540723170000060554001623255604160000
31 2750
31 2751 07172 540723170000060554001623255604160000
31 2752
31 2753 07173 540723170000060554001623255604160000
31 2754
31 2755 07174 010721170000226654001560255704560000
31 2756
31 2757 07175 540723170000060554001623055604360000
31 2758
31 2759 07176 010721370000227014001560255704560000
31i 2755
31 2756 07177 010723370000227014001620055704260000
```

```
        IFRQ ACSEL[AC,MA] ALU[NOTA] DEST[MEMSTO AR] NORM JUMP[ASEOI] $
 ;SETCAB
        IFRQ ACSEL[AC,MA] ALU[NOTA] DEST[B AR] NORM JUMP[ASEOI] $


 .opcode[454]
 [xlist
   list ]      DF/WT D[MEM] ALU[-D&AC] DEST[AR] JUMP[STCM] $
      ;ORCA
       ACSEL[MA,AC] ALU[-A&B]  DEST[AR] JUMP[STCM] $

       D[IR] MASK[18.] ALU[-D&AC] DEST[AR] JUMP[STCM] $
 ;ORCAI
       D[IR] MASK[18.] ALU[-D&AC] DEST[AR] JUMP[STCM] $

       DF/WT R-M-W D[MEM] ALU[-D&AC] DEST[AR] JUMP[STCMM] $
      ;ORCAM
       ACSEL[MA,AC] ALU[-A&B]  DEST[AR] JUMP[STCMMA] $

       DF/WT R-M-W D[MEM] ALU[-D&AC] DEST[AR] JUMP[STCMB] $
      ;ORCAB
       ACSEL[MA,AC] ALU[-A&B]  DEST[AR] JUMP[STCMBA] $


 .opcode[460]
 [xlist
   list ]      DF/IF D[MEM] ALU[NOTD] DEST[AC] NEOI $
      ;SETCM
       ACSEL[MA,AC] ALU[NOTA]  DEST[B] NEOI $

       D[IR] MASK[18.] ALU[NOTD] DEST[AC] NEOI $
 ;SETCMI
       D[IR] MASK[18.] ALU[NOTD] DEST[AC] NEOI $

       DF/IF R-M-W IFRQ D[MEM] ALU[NOTD] DEST[MEMSTO] JUMP[SEOI] $
      ;SETCMM
       IFRQ ACSEL[AC,MA] ALU[NOTB]  DEST[B] NEOI $

       DF/IF R-M-W IFRQ D[MEM] ALU[NOTD] DEST[MEMSTO AC] JUMP[SEOI] $
 ;SETCMB
       IFRQ ACSEL[AC,MA] ALU[NOTB]  DEST[B AR] JUMP[ASEOI] $


 .opcode[464]
 [xlist
   list ]      DF/WT D[MEM] ALU[NOTD] DEST[AR] JUMP[ORCM1] $
      ;ORCM
       ACSEL[MA] ALU[NOTAC] DEST[AR] JUMP[ORCM1] $

       D[IR] MASK[18.] ALU[NOTD] DEST[AR] JUMP[ORCM1] $
 ;ORCMI
       D[IR] MASK[18.] ALU[NOTD] DEST[AR] JUMP[ORCM1] $

       DF/WT R-M-W D[MEM] ALU[NOTD] DEST[AR] JUMP[ORCMM1] $
 ;ORCMM
       ACSEL[MA] ALU[NOTAC] DEST[AR] JUMP[ORCMM2] $

       DF/WT R-M-W D[MEM] ALU[NOTD] DEST[AR] JUMP[ORCMB1] $
 ;ORCMB
       ACSEL[MA] ALU[NOTAC] DEST[AR] JUMP[ORCMB2] $

 .reloc
 [.USE[NORMAL]
 [ xlist
   list ]                                        ]ORCM1:      D[AR] ALU[DORAC]
 DEST[AC] NEOI $
ORCMM1: IFRQ D[AR] ALU[DORAC] DEST[MEMSTO] JUMP[SEOI] $
ORCMM2: D[AR] ACSEL[AC,MA] ALU[DORAC] DEST[B] NEOI $
ORCMB1: IFRQ D[AR] ALU[DORAC] DEST[MEMSTO AR] JUMP[ASEOI] $
ORCMB2: IFRQ D[AR] ACSEL[AC,MA] ALU[DORAC] DEST[B AR] JUMP[ASEOI] $


 .opcode[470]
 [xlist
   list ]      DF/WT D[MEM] ALU[D&AC] DEST[AR] JUMP[STCM] $
      ;ORCB
       ACSEL[MA,AC] ALU[A&B]  DEST[AR] JUMP[STCM] $

       D[IR] MASK[18.] ALU[D&AC] DEST[AR] JUMP[STCM] $
 ;ORCBI
       D[IR] MASK[18.] ALU[D&AC] DEST[AR] JUMP[STCM] $

       DF/WT R-M-W IFRQ D[MEM] ALU[D&AC] DEST[AR] JUMP[STCMM] $
 ;ORCBM
       ACSEL[AC,MA] ALU[A&B]  DEST[AR] JUMP[STCMMA] $

       DF/IF R-M-W IFRQ D[MEM] ALU[D&AC] DEST[AR] JUMP[STCMB] $
 ;ORCBB
       IFRQ ACSEL[AC,MA] ALU[A&B]  DEST[AR] JUMP[STCMBA] $


 .opcode[474]
 [xlist
   list ]      ALU[-1] DEST[AC] NEOI $
      ;SETO
       ALU[-1]  DEST[B] NEOI $

       ALU[-1] DEST[AC] NEOI $
 ;SETOI
       ALU[-1]  DEST[B] NEOI $

       IFRQ ALU[-1] DEST[MEMSTO] JUMP[SEOI] $
 ;SETOM
       ALU[-1]  DEST[MEMAC] NEOI $

       IFRQ ALU[-1] DEST[MEMSTO AR] JUMP[ASEOI] $
 ;SETOB
```

```
32 2759
32m2760
32m2760
32 2761 07200 010540170000000001404562024731256000
32 2761
32 2762 07201 010540170000000001404562024731456000
32 2763
32 2764 07202 540543170000060554045623247204160000
32 2764
32 2765 07203 540543170000060554045623247204160000
32 2766
32 2767 07204 010540170000000001414162024731256000
32 2767
32 2768 07205 010540170000000001414162024731456000
32 2769
32 2770 07206 560731170000060554160163254603160000
32 2770
32 2771 07207 540431170000060554160163014204160000
32 2772
32m2773
32m2773
32m2773
32 2774 05376 540633170000060554141623254605160000
32 2774
32 2775 05377 010331170000127754161500054314560000
32 2776 05400 010531171000226654045560254705560000
32 2777 05401 010541370000000001404562000731456000
32 2778 05402 540633170000060554161623054204360000
32 2779
32m2780
32m2780
32 2781 07210 010540170000000001404562024731256000
32 2781
32 2782 07211 010540170000000001404562024731456000
32 2783
32 2784 07212 010540170000000001404562024731456000
32 2784
32 2785 07213 010540170000000001404562024731456000
32 2786
32 2787 07214 010331370000000001416162025431256000
32 2787
32 2788 07215 010331370000000001416162025431456000
32 2789
32 2790 07216 010731170000000001416162025431256000
32 2790
32 2791 07217 010331370000000001416162005431456000
32 2792
32m2793
32m2793
32m2793
32 2794 05403 540633170000060544541623254605160000
32 2794
32 2795 05404 010331170000130074161500054314560000
32 2796 05405 540633170000060544541623257604160000
32 2797 05406 010730170000130004541620254314560000
32 2798 05407 010540170000006055404562340731416000
32 2799 05410 540633170000060544541623054204360000
32 2800 05411 010730171000060544541623654715160000
32 2801 05412 010631370000227034045560254705560000
32 2801
32 2802 05413 010730114000000004541621254314560000
32m2803
32m2803
32 2804 05414 010633170000060554045623654314160000
32 2805 05415 540633170000060554045623054204360000
32 2806
32 2807
32m2808
32m2808
32 2809 07220 560733170000060554141623254603160000
32 2809
32 2810 07221 540553170000060554045623007204160000
32 2811
32 2812 07222 540243170000060554161623254204160000
32 2812
32 2813 07223 540243170000060554161623254204160000
32 2814
32 2815 07224 010551170000226654045560247304560000
32 2815
32 2816 07225 540553170000060554045623047204160000
32 2817
32 2818 07226 030731371000227034141560254703560000
32 2818
32 2819
32 2820 07227 010551370000000001404416000731456000
32m2821
32m2821
32m2821
32 2822 05416 540733170000060554161623054204360000
32 2822
32 2823
32 2824
32m2825
32m2825
32 2826 07230 560733170000060544541623254603160000
32 2826
32 2827 07231 010331370000000001416162005431456000
32 2828
32 2829 07232 540733170000060544541623257604160000
32 2829
32 2830 07233 540733170000060544541623257604160000
32 2831
32 2832 07234 010331370000000001416162025431456000
32 2832
32 2833 07235 010331370000000001416162025431456000
32 2834
32 2835
32 2836 07236 030731371000227024541560254703560000
32 2835
32 2837 07237 010331370000000001416162005431456000
```

```
.opcode[500]
[xlist
  list ]      DFRQ D[MASK 18.] ALU[D&AC] DEST[Q] JUMP[HLL1] $
      ;HLL
       D[MASK 18.] ALU[D&AC] DEST[Q] JUMP[HLLZ] $

       D[MASK 18.] ALU[D&AC] DEST[AC] NEOI $
;HLLI
       D[MASK 18.] ALU[D&AC] DEST[AC] NEOI $

       DFRQ D[MASK 60] ALU[D&AC] DEST[Q] JUMP[HLLM1] $
;HLLM
       D[MASK 60] ALU[D&AC] DEST[Q] NORM JUMP[HLLM2] $

       DF/IF D[MEM] COND-AC-STO NEOI $
;HLLS
       ACSEL[MA,AC] ALU[A] COND-AC-STO NEOI $

.reloc
[.USE[NORMAL]
[ xlist
  list ]                                    ]HLL1:      D[MEM] MASK[LEFT]
ALU[DORQ] DEST[AC] NEOI $
HLLZ:   ALU[MEMAC] DEST[HOLD] NORM JUMP[HLL1] $
HLLM1:  IFRQ R-M-W D[MEM] MASK[18.] ALU[DORQ] DEST[MEMSTO] JUMP[SEOI] $
HLLM2:  D[MASK 18.] ACSEL[MA,AC] ALU[D&AC] DEST[AR] NORM JUMP[HXA] $
HXA:    D[AR] ALU[DORQ] DEST[MEMAC] NEOI $

.opcode[504]
[xlist
  list ]      DFRQ D[MASK 18.] ALU[D&AC] DEST[Q] JUMP[HRL1] $
      ;HRL
       D[MASK 18.] ALU[D&AC] DEST[Q] JUMP[HRLZ] $

       D[MASK 18.] ALU[D&AC] DEST[Q] JUMP[HRLI1] $
;HRLI
       D[MASK 18.] ALU[D&AC] DEST[Q] JUMP[HRLI1] $

       DFRQ ALU[AC] DEST[AR] JUMP[HRLM1] $
;HRLM
       ALU[AC] DEST[AR] NORM JUMP[HRLM2] $

       DFRQ JUMP[HRLS1] $
;HRLS
       ALU[MEMAC] DEST[AR] NORM JUMP[HRLS2] $

.reloc
[.USE[NORMAL]
[ xlist
  list ]                                    ]HRL1:      D[MEM] ROT[18.]
MASK[LEFT] ALU[DORQ] DEST[AC] NEOI $
HRLZ:   ALU[MEMAC] DEST[HOLD] NORM JUMP[HRL1] $
HRLI1:  D[IR]  ROT[18.] MASK[LEFT] ALU[DORQ] DEST[AC] NEOI $
HRLM1:  D[AR] ROT[18.] MASK[LEFT] DEST[Q] NORM JUMP[HLLM1] $
HRLM2:  D[MASK 18.] ACSEL[MA,AC] ALU[D&AC] DEST[Q] NORM  $
        D[AR]  ROT[18.] MASK[LEFT] ALU[DORQ] DEST[MEMAC] NEOI $
HRLS1:  R-M-W D[MEM] ROT[18.] MASK[LEFT] DEST[Q] NORM $
        IFRQ D[MEM]  MASK[18.] ALU[DORQ] DEST[MEMSTO AR] JUMP[CASEOI]
 $
HRLS2:  D[AR] ROT[18.] MASK[LEFT] DEST[Q] NORM  AC=0 LBJUMP[HRLS3] $
 .pair
[:. \ Z + .
  ]HRLS3:     D[AR] MASK[18.] ALU[DORQ] DEST[AC] NORM $
       D[AR] MASK[18.] ALU[DORQ] DEST[MEMAC] NEOI $


.opcode[510]
[xlist
  list ]      DF/IF D[MEM] MASK[LEFT] DEST[AC] NEOI $
      ;HLLZ
       D[MASK 18.] ACSEL[MA,AC] ALU[-D&A] DEST[B] NEOI $

       ALU[0] DEST[AC] NEOI $
;HLLZI
       ALU[0] DEST[AC] NEOI $

       IFRQ D[MASK 18.] ALU[-D&AC] DEST[MEMSTO] JUMP[SEOI] $
;HLLZM
       D[MASK 18.] ACSEL[AC,MA] ALU[-D&A] DEST[B] NEOI $

       DF/IF R-M-W D[MEM] MASK[LEFT] DEST[MEMSTO AR] JUMP[CASEOI]  $
;HLLZS
       D[MASK 18.] ACSEL[MA,AC] ALU[-D&A] DEST[AR] COND-AC-STO
                                                  NORM  JUMP[HLLZS1] $
.reloc
[.USE[NORMAL]
[ xlist
  list ]                                    ]HLLZS1:    D[AR] DEST[MEMAC]
NEOI $


.opcode[514]
[xlist
  list ]      DF/IF D[MEM] ROT[18.] MASK[LEFT] DEST[AC] NEOI $
      ;HRLZ
       ALU[MEMAC] DEST[AR] NORM JUMP[HRLZ1] $

       D[IR] ROT[18.] MASK[LEFT] DEST[AC] NEOI $
;HRLZI
       D[IR] ROT[18.] MASK[LEFT] DEST[AC] NEOI $

       ALU[AC] DEST[AR] NORM JUMP[HRLZM1] $
;HRLZM
       ALU[AC] DEST[AR] NORM JUMP[HRLZM2] $

       DF/IF R-M-W IFRQ D[MEM] ROT[18.] MASK[LEFT] DEST[MEMSTO AR]
                                                        JUMP[CASEOI] $
;HRLZS
       ALU[MEMAC] DEST[AR] NORM  JUMP[HRLZS1] $
```

```
32 2841 05420 010731170000ZZ6644541560Z5430456000      HRLZM1: IFRQ D[AR] ROT[18.] MASK[LEFT] DEST[MEMSTO] JUMP[SEQI] $
32 2842 05421 540733170000060544541623054203G000      HRLZM2:      D[AR] ROT[18.] MASK[LEFT] DEST[MEMAC] NEQI $
32 2843 05422 010731700001303445401602543145G000      HRLZS1: D[AR] ROT[18.] MASK[LEFT] DEST[AR] COND-AC-STO NORM JUMP[HLLZS1]
32 2843                                                 $
32 2844
32m2845                                                 .opcode[520]
32m2845                                                [xlist
32 2846 07240 010730170000000001404562024731ZG6000       list  ]      DFRQ D[MASK 18.]  DEST[Q] JUMP[HLLO1] $
32 2846                                                      ;HLLO
32 2847 07241 540533170000060554045623007204160000           D[MASK 18.] ACSEL[MA,AC] ALU[DORA] DEST[AC] NEQI $
32 2848
32 2849 07242 540733170000060554045623247204160000           D[MASK 18.]  DEST[AC] NEQI $
32 2849                                                ;HLLOI
32 2850 07243 540733170000060554045623247204160000           D[MASK 18.]  DEST[AC] NEQI $
32 2851
32 2852 07244 010531170000ZZ665404556024730456000           IFRQ D[MASK 18.] ALU[DORAC] DEST[MEMSTO] JUMP[SEQI] $
32 2852                                                ;HLLOM
32 2853 07245 540533170000060554045623047204160000           D[MASK 18.] ACSEL[AC,MA] ALU[DORA] DEST[B] NEQI $
32 2854
32 2855 07246 010730170000000001404562024731ZG6000           DFRQ D[MASK 18.]  DEST[Q] JUMP[HLLOS1] $
32 2855                                                ;HLLOS
32 2856 07247 010531370000130354044160007314560O0           D[MASK 18.] ACSEL[MA,AC] ALU[DORA] DEST[AR] COND-AC-STO
32 2856                                                JUMP[HLLZS1] $
32 2857
32m2858                                                 .reloc
32m2858                                                [.USE[NORMAL]
32m2858                                                [ xlist
32 2859 05423 540633170000060554161623254605160000       list ]                                    ]HLLO1:      D[MEM] ALU[DORQ]
32 2859                                                DEST[AC] NEQI $
32 2860 05424 010631371000ZZ703416156025470556000      HLLOS1: IFRQ R-M-W D[MEM] ALU[DORQ] DEST[MEMSTO AR] JUMP[CASEQI]
32 2860                                                 $
32 2861
32m2862                                                 .opcode[524]
32m2862                                                [xlist
32 2863 07250 010730170000000001404562024731ZG6000       list  ]      DFRQ D[MASK 18.]  DEST[Q] JUMP[HRLO1] $
32 2863                                                      ;HRLO
32 2864 07251 010531170000000001414150000731456000           D[MASK 60] ACSEL[MA,AC] ALU[DORAC] DEST[HOLD] NORM JUMP[HRLOZ] $
32 2864
32 2865
32 2866 07252 010730170000000004561620257714560O0           D[IR] ROT[18.] DEST[Q] NORM JUMP[HRLOI1] $
32 2866                                                ;HRLOI
32 2867 07253 010730170000000004561620257714560O0           D[IR] ROT[18.] DEST[Q] NORM JUMP[HRLOI1] $
32 2868
32 2869 07254 010531370000000001414162024731456000           D[MASK 60] ALU[DORAC] DEST[AR]  NORM JUMP[HRLOM1] $
32 2869                                                ;HRLOM
32 2870 07255 010531370000000001414162024731456000           D[MASK 60] ALU[DORAC] DEST[AR]  NORM JUMP[HRLOM2] $
32 2871
32 2872 07256 010730170000000001404562024731ZG6000           DFRQ D[MASK 18.]  DEST[Q] JUMP[HRLOS1] $
32 2872                                                ;HRLOS
32 2873 07257 010531370000000001414162000731456000           D[MASK 60] ACSEL[MA,AC] ALU[DORA] DEST[AR] NORM JUMP[HRLOSZ] $
32 2874
32m2875                                                 .reloc
32m2875                                                [.USE[NORMAL]
32m2875                                                [ xlist
32 2876 05425 540633170000060544561623254605160000       list ]                                    ]HRLO1:      D[MEM] ROT[18.]
32 2876                                                ALU[DORQ] DEST[AC] NEQI $
32 2877 05426 540733170000060544561623254605160000      HRLOZ:  D[MEM] ROT[18.] DEST[AC] NEQI $
32 2878 05427 540633170000060554045623247204160000      HRLOI1: D[MASK 18.] ALU[DORQ] DEST[AC] NEQI $
32 2879 05430 010731170000ZZ664456156025430456000      HRLOM1: IFRQ D[AR] ROT[18.] DEST[MEMSTO] JUMP[SEQI] $
32 2880 05431 540733170000060544561623054203G000      HRLOM2:      D[AR] ROT[18.] DEST[MEMAC] NEQI $
32 2881 05432 010631371000ZZ702456156025470556000      HRLOS1: IFRQ R-M-W D[MEM] ROT[18.] ALU[DORQ] DEST[MEMSTO AR] JUMP[CASEQI]
32 2881                                                 $
32 2882 05433 010731370000130344560160254314560O0      HRLOS2:      D[AR] ROT[18.] DEST[AR]   COND-AC-STO JUMP[HLLZS1] $
32 2883
32 2884
32m2885                                                 .opcode[530]
32m2885                                                [xlist
32 2886 07260 010730170000000001404562024731ZG6000       list  ]      DFRQ D[MASK 18.]  DEST[Q] JUMP[HLLE1] $
32 2886                                                      ;HLLE
32 2887 07261 010331004000000001416162105431456000           ALU[MEMAC] NORM OBUS<0 LBJUMP[HLLE3] $
32 2888
32 2889 07262 540243170000060554161623254204160000           ALU[0] DEST[AC] NEQI $
32 2889                                                ;HLLEI
32 2890 07263 540243170000060554161623254204160000           ALU[0] DEST[AC] NEQI $
32 2891
32 2892 07264 010331004000000001416162125431456000           ALU[AC] NORM OBUS<0 LBJUMP[HLLEM1] $                    ;HLLEM
32 2893 07265 010331004000000001416162125431456000           ALU[AC] NORM OBUS<0 LBJUMP[HLLEM2] $
32 2894
32 2895 07266 010730170000000001404562024731ZG6000           DFRQ D[MASK 18.] DEST[Q] JUMP[HLLES1] $                 ;HLLES
32 2896 07267 010331004000000001416162105431456000           ALU[MEMAC] NORM OBUS<0 LBJUMP[HLLES3] $
32 2897
32m2898                                                 .reloc
32m2898                                                [.USE[NORMAL]
32m2898                                                [ xlist
32 2899 05434 010731004000000001416162125471556000       list ]                                    ]HLLE1:      D[MEM] OBUS<0
32 2899                                                LBJUMP[HLLEZ] $
32 2900 05435 010731005000000001416162125471556000      HLLES1: R-M-W D[MEM] OBUS<0 LBJUMP[HLLESZ] $
32m2901                                                  .pair
32m2901                                                 [:. \ Z + .
32 2902 05436 540733170000060554141623254605160000        ]HLLEZ:      D[MEM] MASK[LEFT] DEST[AC] NEQI $
32 2903 05437 540633170000060554161623254605160000           D[MEM] ALU[DORQ] DEST[AC] NEQI $
32 2904 05440 540553170000060554045623007204160000      HLLE3:  D[MASK 18.] ACSEL[MA,AC] ALU[-D&A] DEST[B] NEQI $
32 2905 05441 540533170000060554045623007204160000           D[MASK 18.] ACSEL[MA,AC] ALU[DORA] DEST[B] NEQI $
32 2906 05442 010551170000ZZ665404556004730456000      HLLEM1: IFRQ D[MASK 18.] ACSEL[AC,MA] ALU[-D&A] DEST[MEMSTO] JUMP[SEQI] $
32 2906
32 2907 05443 010531170000ZZ665404556004730456000           IFRQ D[MASK 18.] ACSEL[AC,MA] ALU[DORA] DEST[MEMSTO] JUMP[SEQI] $
32 2907
32 2908 05444 540553170000060554045623047204160000      HLLEM2:      D[MASK 18.] ACSEL[AC,MA] ALU[-D&A] DEST[B] NEQI $
32 2909 05445 540533170000060554045623047204160000                   D[MASK 18.] ACSEL[AC,MA] ALU[DORA] DEST[B] NEQI $
32 2910 05446 010731370000ZZ703414156025470556000      HLLES2: IFRQ D[MEM] MASK[LEFT] DEST[MEMSTO AR] JUMP[CASEQI]      $
32 2911 05447 010631370000ZZ703416156025470556000           IFRQ D[MEM] ALU[DORQ]  DEST[MEMSTO AR] JUMP[CASEQI]      $
32 2912 05450 010551370000130354044160007314560O0      HLLES3: D[MASK 18.] ACSEL[MA,AC] ALU[-D&A] DEST[AR] COND-AC-STO
32 2912                                                JUMP[HLLZS1] $
32 2913 05451 010531370000130354044160007314560O0           D[MASK 18.] ACSEL[MA,AC] ALU[DORA] DEST[AR] COND-AC-STO
32 2913                                                JUMP[HLLZS1] $
32 2914
32 2915
32m2916                                                 .opcode[534]
32m2916                                                [xlist
```

```
32 2922
32 2923 07274 010730170000000001404562024731456000            D[MASK 18.] DEST[Q] JUMP[HRLEM1] $                    ;HRLEM
32 2924 07275 010730170000000001404562024731456000            D[MASK 18.] DEST[Q] JUMP[HRLEM4] $
32 2925
32 2926 07276 010730170000000001404562024731256000            DFRQ D[MASK 18.] DEST[Q] JUMP[HRLES1] $               ;HRLES
32 2927 07277 010730170000000001404562024731456000            D[MASK 18.] DEST[Q] JUMP[HRLES4] $
32 2928
32m2929                                                        .reloc
32m2929                                                       [.USE[NORMAL]
32m2929                                                       [ xlist
32 2930 05452 010731020000000001416162125471556000             list ]                                ]HRLE1:      D[MEM] OBUS18
32 2930                                                       LBJUMP[HRLE2] $
32 2931 05453 010331020000000001416150105431456000           HRLE4:  ALU[MEMAC] DEST[HOLD] NORM OBUS18 LBJUMP[HRLE2] $
32 2932 05454 010731020000000001416150125771456000           HRLE1: D[IR]      DEST[HOLD] NORM OBUS18 LBJUMP[HRLE2] $
32 2933 05455 010331220000000001416162125431456000           HRLEM1: ALU[AC] DEST[AR] NORM OBUS18 LBJUMP[HRLEM2] $
32 2934 05456 010331220000000001416162125431456000           HRLEM4: ALU[AC] DEST[AR] NORM OBUS18 LBJUMP[HRLEM3] $
32 2935 05457 010731021000000001416162125471556000           HRLES1: R-M-W D[MEM] OBUS18 LBJUMP[HRLES2] $
32 2936 05460 010331220000000001416162105431456000           HRLES4: ALU[MEMAC] DEST[AR] NORM OBUS18 LBJUMP[HRLES3] $
32m2937                                                        .pair
32m2937                                                       [:. \ 2 + .
32 2938 05462 540733170000060544541623254605 16000             ]HRLE2:      D[MEM] ROT[18.] MASK[LEFT] DEST[AC] NEOI $
32 2939 05463 540633170000060544561623254605 16000                 D[MEM] ROT[18.] ALU[DORQ] DEST[AC] NEOI $
32 2940 05464 010731170000226544541560254304 56000           HRLEM2: IFRQ D[AR] ROT[18.] MASK[LEFT] DEST[MEMSTO] JUMP[SEOI] $
32 2941 05465 010631170000226544561560254304 56000                 IFRQ D[AR] ROT[18.] ALU[DORQ] DEST[MEMSTO] JUMP[SEOI] $
32 2942 05466 540733170000060544541623054204 36000           HRLEM3:      D[AR] ROT[18.] MASK[LEFT] DEST[MEMAC] NEOI $
32 2943 05467 540633170000060544561623054204 36000                 D[AR] ROT[18.] ALU[DORQ]  DEST[MEMAC] NEOI $
32 2944 05470 010731370000227024541560254705 56000           HRLES2: IFRQ D[MEM] ROT[18.] MASK[LEFT] DEST[MEMSTO AR] JUMP[CASEOI]
32 2944                                                       $
32 2945 05471 010631370000227024561560254705 56000                 IFRQ D[MEM] ROT[18.] ALU[DORQ]  DEST[MEMSTO AR] JUMP[CASEOI]
32 2945                                                       $
32 2946 05472 010731370000130344540160254314 56000           HRLES3: D[AR] ROT[18.] MASK[LEFT] DEST[AR] COND-AC-STO JUMP[HLLZS1] $
32 2947 05473 010631370000130344560160254314 56000                 D[AR] ROT[18.] ALU[DORQ]  DEST[AR] COND-AC-STO JUMP[HLLZS1] $
32 2948
32 2949
32 2950
32m2951                                                        .opcode[540]
32m2951                                                       [xlist
32 2952 07300 010540170000000001414162024731256000             list  ]      DFRQ D[MASK 60] ALU[D&AC] DEST[Q] JUMP[HRR1] $
32 2952                                                             ;HRR
32 2953 07301 010540170000000001414162024731456000                  D[MASK 60] ALU[D&AC] DEST[Q] JUMP[HRR2] $
32 2954
32 2955 07302 010540170000000001414162024731456000                 D[MASK 60] ALU[D&AC] DEST[Q] JUMP[HRRI1] $
32 2955                                                       ;HRRI
32 2956 07303 010540170000000001414162024731456000                 D[MASK 60] ALU[D&AC] DEST[Q] JUMP[HRRI1] $
32 2957
32 2958 07304 010540170000000001404562024731256000                 DFRQ D[MASK 18.] ALU[D&AC] DEST[Q] JUMP[HRRM1] $
32 2958                                                       ;HRRM
32 2959 07305 010540170000000001404562024731456000                 D[MASK 18.] ALU[D&AC] DEST[Q] NORM JUMP[HRRM2] $
32 2960
32 2961 07306 560731170000060554160163254603 16000                 DF/IF D[MEM] COND-AC-STO NEOI $
32 2961                                                       ;HRRS
32 2962 07307 540431170000060554160163014204 16000                 ACSEL[MA,AC] ALU[A] COND-AC-STO NEOI $
32 2963
32m2964                                                        .reloc
32m2964                                                       [.USE[NORMAL]
32m2964                                                       [ xlist
32 2965 05474 540633170000060554045623254605 16000             list ]                                ]HRR1:       D[MEM] MASK[18.]
32 2965                                                       ALU[DORQ] DEST[AC] NEOI $
32 2966 05475 010331170000131714161500054314 56000           HRR2:   ALU[MEMAC] DEST[HOLD] NORM JUMP[HRR1] $
32 2967 05476 540633170000060554045623257604 16000           HRRI1:  D[IR]  MASK[18.] ALU[DORQ] DEST[AC] NEOI $
32 2968 05477 010631171000226554141560254705 56000           HRRM1:  IFRQ R-M-W D[MEM] MASK[60] ALU[DORQ] DEST[MEMSTO] JUMP[SEOI] $
32 2969 05500 010541370000060554141623407314 16000           HRRM2:  D[MASK 60] ACSEL[MA,AC] ALU[D&AC] DEST[AR] NORM $
32 2970 05501 540633170000060554161623054204 36000                 D[AR] ALU[DORQ] DEST[MEMAC] NEOI $
32 2971
32 2972
32m2973                                                        .opcode[544]
32m2973                                                       [xlist
32 2974 07310 010540170000000001414162024731256000             list  ]      DFRQ D[MASK 60] ALU[D&AC] DEST[Q] JUMP[HLR1] $
32 2974                                                             ;HLR
32 2975 07311 010540170000000001414162024731456000                  D[MASK 60] ALU[D&AC] DEST[Q] JUMP[HLR2] $
32 2976
32 2977 07312 540543170000060554141623247204 16000                 D[MASK 60] ALU[D&AC] DEST[AC] NEOI $
32 2977                                                       ;HLRI
32 2978 07313 540543170000060554141623247204 16000                 D[MASK 60] ALU[D&AC] DEST[AC] NEOI $
32 2979
32 2980 07314 010331370000000001416162025431256000                 DFRQ ALU[AC] DEST[AR] JUMP[HLRM1] $
32 2980                                                       ;HLRM
32 2981 07315 010331370000000001416162025431456000                 ALU[AC] DEST[AR] NORM JUMP[HLRM2] $
32 2982
32 2983 07316 010731170000000001416162025431256000                 DFRQ JUMP[HLRS1] $
32 2983                                                       ;HLRS
32 2984 07317 010331370000000001416162005431456000                 ALU[MEMAC] DEST[AR] NORM JUMP[HLRS2] $
32 2985
32m2986                                                        .reloc
32m2986                                                       [.USE[NORMAL]
32m2986                                                       [ xlist
32 2987 05502 540633170000060544445623254605 16000             list ]                                ]HLR1:       D[MEM] ROT[18.]
32 2987                                                       MASK[18.] ALU[DORQ] DEST[AC] NEOI $
32 2988 05503 010331170000132054161500054314 56000           HLR2:   ALU[MEMAC] DEST[HOLD] NORM JUMP[HLR1] $
32 2989 05504 010730170000131764445620254314 56000           HLRM1:  D[AR] ROT[18.] MASK[18.] DEST[Q] NORM JUMP[HRRM1] $
32 2990 05505 010541370000060554141623407314 16000           HLRM2:  D[MASK 60] ACSEL[MA,AC] ALU[D&AC] DEST[Q] NORM  $
32 2991 05506 540633170000060544445623054204 36000                 D[AR]  ROT[18.] MASK[18.] ALU[DORQ] DEST[MEMAC] NEOI $
32 2992 05507 010730171000060544445623654715 16000           HLRS1:  R-M-W D[MEM] ROT[18.] MASK[18.] DEST[Q] NORM $
32 2993 05510 010631370000227034141560254705 56000                 IFRQ D[MEM] MASK[LEFT] ALU[DORQ] DEST[MEMSTO AR] JUMP[CASEOI]
32 2993                                                       $
32 2994 05511 010730114000000004444562125431456000           HLRS2:  D[AR] ROT[18.] MASK[18.] DEST[Q] NORM  AC=0 LBJUMP[HLRS3] $
32m2995                                                        .pair
32m2995                                                       [:. \ 2 + .
32 2996 05512 010633170000060554141623654314 16000             ]HLRS3:      D[AR] MASK[LEFT] ALU[DORQ] DEST[AC] NORM $
32 2997 05513 540633170000060554141623054204 36000                 D[AR] MASK[LEFT] ALU[DORQ] DEST[MEMAC] NEOI $
32 2998
32 2999
32m3000                                                        .opcode[550]
32m3000                                                       [xlist
32 3001 07320 560733170000060554045623254603 16000             list  ]      DF/IF D[MEM] MASK[18.] DEST[AC] NEOI $
32 3001                                                             ;HRRZ
32 3002 07321 540553170000060554141623007204 16000                 D[MASK 60] ACSEL[MA,AC] ALU[-D&A] DEST[B] NEOI $
32 3003
32 3004 07322 540733170000060554045623257604 16000                 D[IR] MASK[18.] DEST[AC] NEOI $
32 3004                                                       ;HRRZI
```

```
32 3008 07325 540553170000060554141623047204160000             D[MASK 60] ACSEL[AC,MA] ALU[-D&A] DEST[B] NEOI $
32 3009
32 3010 07326 030731371000227034045560254703560000             DF/IF R-M-W D[MEM] MASK[18.] DEST[MEMSTO AR] JUMP[CASEOI] $
32 3010                                                        ;HRRZS
32 3011                                                         D[MASK 60] ACSEL[MA,AC] ALU[-D&A] DEST[AR] COND-AC-STO
32 3012 07327 010551370000227074140160007304560000                                                 NORM  JUMP[MACSTO] IFRQ $
32 3012
32 3013
32 3014
32m3015                                                         .opcode[554]
32m3015                                                        [xlist
32 3016 07330 560733170000060544445623254603160000               list  ]      DF/IF D[MEM] ROT[18.] MASK[18.] DEST[AC] NEOI $
32 3016                                                              ;HLRZ
32 3017 07331 010331370000000001416162005431456000             ALU[MEMAC] DEST[AR] NORM JUMP[HLRZ1] $
32 3018
32 3019 07332 540243170000060554161623254204160000             ALU[0] DEST[AC] NEOI $
32 3019                                                        ;HLRZI
32 3020 07333 540243170000060554161623254204160000             ALU[0] DEST[AC] NEOI $
32 3021
32 3022 07334 010331370000000001416162025431456000             ALU[AC] DEST[AR] NORM JUMP[HLRZM1] $
32 3022                                                        ;HLRZM
32 3023 07335 010331370000000001416162025431456000             ALU[AC] DEST[AR] NORM JUMP[HLRZM2] $
32 3024
32 3025                                                         DF/IF R-M-W D[MEM] ROT[18.] MASK[18.] DEST[MEMSTO AR]
32 3026 07336 030731371000227024445560254703560000                                                                 JUMP[CASEOI] $
32 3026                                                        ;HLRZS
32 3027 07337 010331370000000001416162005431456000             ALU[MEMAC] DEST[AR] NORM  JUMP[HLRZS1] $
32 3028
32m3029                                                         .reloc
32m3029                                                        [.USE[NORMAL]
32m3029                                                        [ xlist
32 3030 05514 540733170000060544445623254204160000               list ]                                    ]HLRZ1:      D[AR] ROT[18.]
32 3030                                                        MASK[18.] DEST[AC] NEOI $
32 3031 05515 010731170000226644445560254304560000            HLRZM1: IFRQ D[AR] ROT[18.] MASK[18.] DEST[MEMSTO] JUMP[SEOI] $
32 3032 05516 540733170000060544445623054204360000            HLRZM2:      D[AR] ROT[18.] MASK[18.] DEST[MEMAC] NEOI $
32 3033 05517 010731370000227064444160254304560000            HLRZS1: D[AR] ROT[18.] MASK[18.] DEST[AR] COND-AC-STO NORM JUMP[MACSTO]
32 3033                                                        IFRQ $
32 3034
32m3035                                                         .opcode[550]
32m3035                                                        [xlist
32 3036 07340 010730170000000001414162024731256000               list  ]      DFRQ D[MASK 60]  DEST[Q] JUMP[HRROI] $
32 3036                                                              ;HRRQ
32 3037 07341 540533170000060554141623007204160000             D[MASK 60] ACSEL[MA,AC] ALU[DORA] DEST[AC] NEOI $
32 3038
32 3039 07342 010730170000000001414162024731456000             D[MASK 60]  DEST[Q] JUMP[HRROI1] $
32 3039                                                        ;HRROI
32 3040 07343 010730170000000001414162024731456000             D[MASK 60]  DEST[Q] JUMP[HRROI1] $
32 3041
32 3042 07344 010531170000226654141560247304560000             IFRQ D[MASK 60] ALU[DORAC] DEST[MEMSTO] JUMP[SEOI] $
32 3042                                                        ;HRROM
32 3043 07345 540533170000060554141623047204160000             D[MASK 60] ACSEL[AC,MA] ALU[DORA] DEST[B] NEOI $
32 3044
32 3045 07346 010730170000000001414162024731256000             DFRQ D[MASK 60]  DEST[Q] JUMP[HRROS1] $
32 3045                                                        ;HRROS
32 3046 07347 010531370000227074140160007304560000             D[MASK 60] ACSEL[MA,AC] ALU[DORA] DEST[AR] COND-AC-STO
32 3046                                                        JUMP[MACSTO] IFRQ $
32 3047
32m3048                                                         .reloc
32m3048                                                        [.USE[NORMAL]
32m3048                                                        [ xlist
32 3049 05520 540633170000060554161623254605160000               list ]                                    ]HRROI:      D[MEM] ALU[DORQ]
32 3049                                                        DEST[AC] NEOI $
32 3050 05521 540633170000060554161623257604160000            HRROI1: D[IR]  ALU[DORQ] DEST[AC] NEOI $
32 3051 05522 010631371000227034161560254705560000            HRROS1: IFRQ R-M-W D[MEM] ALU[DORQ] DEST[MEMSTO AR] JUMP[CASEOI]
32 3051                                                        $
32 3052
32m3053                                                         .opcode[564]
32m3053                                                        [xlist
32 3054 07350 010730170000000001414162024731256000               list  ]      DFRQ D[MASK 60]  DEST[Q] JUMP[HLRO1] $
32 3054                                                              ;HLRO
32 3055 07351 010531170000000001404550000731456000             D[MASK 18.] ACSEL[MA,AC] ALU[DORAC] DEST[HOLD] NORM JUMP[HLRO2] $
32 3055
32 3056
32 3057 07352 540733170000060554141623247204160000             D[MASK 60] DEST[AC] NEOI $
32 3057                                                        ;HLROI
32 3058 07353 540733170000060554141623247204160000             D[MASK 60] DEST[AC] NEOI $
32 3059
32 3060 07354 010531370000000001404562024731456000             D[MASK 18.] ALU[DORAC] DEST[AR]  NORM JUMP[HLROM1] $
32 3060                                                        ;HLROM
32 3061 07355 010531370000000001404562024731456000             D[MASK 18.] ALU[DORAC] DEST[AR]  NORM JUMP[HLROM2] $
32 3062
32 3063 07356 010730170000000001414162024731256000             DFRQ D[MASK 60]  DEST[Q] JUMP[HLROS1] $
32 3063                                                        ;HLROS
32 3064 07357 010531370000000001404562000731456000             D[MASK 18.] ACSEL[MA,AC] ALU[DORA] DEST[AR] NORM JUMP[HLROS2] $
32 3065
32m3066                                                         .reloc
32m3066                                                        [.USE[NORMAL]
32m3066                                                        [ xlist
32 3067 05523 540633170000060544561623254605160000               list ]                                    ]HLRO1:      D[MEM] ROT[18.]
32 3067                                                        ALU[DORQ] DEST[AC] NEOI $
32 3068 05524 540733170000060544561623254605160000            HLRO2:  D[MEM] ROT[18.] DEST[AC] NEOI $
32 3069 05525 010731170000226644561560254304560000            HLROM1: IFRQ D[AR] ROT[18.] DEST[MEMSTO] JUMP[SEOI] $
32 3070 05526 540733170000060544561623054204360000            HLROM2:      D[AR] ROT[18.] DEST[MEMAC] NEOI $
32 3071 05527 010631371000227024561560254705560000            HLROS1: IFRQ R-M-W D[MEM] ROT[18.] ALU[DORQ] DEST[MEMSTO AR] JUMP[CASEOI]
32 3071                                                        $
32 3072 05530 010731370000227064560160254304560000            HLROS2:      D[AR] ROT[18.] DEST[AR]   COND-AC-STO JUMP[MACSTO] IFRQ $
32 3073
32 3074
32m3075                                                         .opcode[570]
32m3075                                                        [xlist
32 3076 07360 010730170000000001414162024731256000               list  ]      DFRQ D[MASK 60]  DEST[Q] JUMP[HRRE1] $
32 3076                                                        ;HRRE
32 3077 07361 010331020000000001416162105431456000             ALU[MEMAC] NORM OBUS18 LBJUMP[HRRE3] $
32 3078
32 3079 07362 010730170000000001414162024731456000             D[MASK 60] DEST[Q] JUMP[HRREI1] $                       ;HRREI
32 3080 07363 010730170000000001414162024731456000             D[MASK 60] DEST[Q] JUMP[HRREI1] $
32 3081
32 3082 07364 010331020000000001416162125431456000             ALU[AC] NORM OBUS18 LBJUMP[HRREM1] $                    ;HRREM
32 3083 07365 010331020000000001416162125431456000             ALU[AC] NORM OBUS18 LBJUMP[HRREM2] $
32 3084
```

```
32m3088                                                  [ xlist
32 3089 05531 010731020000000014161621254715560000        list ]                                    ]HRRE1:      D[MEM] OBUS18
32 3089                                                  LBJUMP[HRREZ] $
32 3090 05532 010731020000000014161501257715560000      HRREI1: D[IR]      DEST[HOLD] NORM OBUS18 LBJUMP[HRREZ] $
32 3091 05533 010731021000000014161621254715560000      HRRES1: R-M-W D[MEM] OBUS18 LBJUMP[HRRES2] $
32m3092                                                   .pair
32m3092                                                  [:. \ 2 + .
32 3093 05534 540733170000060554045623254605160000        ]HRREZ:      D[MEM] MASK[18.] DEST[AC] NEOI $
32 3094 05535 540633170000060554161623254605160000             D[MEM] ALU[DORQ] DEST[AC] NEOI $
32 3095 05536 540553170000060554141623007204160000      HRRE3:  D[MASK 60] ACSEL[MA,AC] ALU[-D&A] DEST[B] NEOI $
32 3096 05537 540533170000060554141623007204160000             D[MASK 60] ACSEL[MA,AC] ALU[DORA] DEST[B] NEOI $
32 3097 05540 010551170000226654141560047304560000      HRREM1: IFRQ D[MASK 60] ACSEL[AC,MA] ALU[-D&A] DEST[MEMSTO] JUMP[SEQI] $
32 3097
32 3098 05541 010531170000226654141560047304560000             IFRQ D[MASK 60] ACSEL[AC,MA] ALU[DORA] DEST[MEMSTO] JUMP[SEQI] $
32 3098
32 3099 05542 540553170000060554141623047204160000      HRREM2:      D[MASK 60] ACSEL[AC,MA] ALU[-D&A] DEST[B] NEOI $
32 3100 05543 540533170000060554141623047204160000             D[MASK 60] ACSEL[AC,MA] ALU[DORA] DEST[B] NEOI $
32 3101 05544 010731370000227034045560254705560000      HRRES2: IFRQ D[MEM] MASK[18.] DEST[MEMSTO AR] JUMP[CASEQI]      $
32 3102 05545 010631370000227034161560254705560000             IFRQ D[MEM] ALU[DORQ] DEST[MEMSTO AR] JUMP[CASEQI]      $
32 3103 05546 010551370000227074140160007304560000      HRRES3: D[MASK 60] ACSEL[MA,AC] ALU[-D&A] DEST[AR] COND-AC-STO
32 3103                                                  JUMP[MACSTO] IFRQ $
32 3104 05547 010531370000227074140160007304560000             D[MASK 60] ACSEL[MA,AC] ALU[DORA] DEST[AR] COND-AC-STO
32 3104                                                  JUMP[MACSTO] IFRQ $
32 3105
32 3106
32m3107                                                   .opcode[574]
32m3107                                                  [xlist
32 3108 07370 010730170000000014141620247312560000        list  ]      DFRQ D[MASK 60]  DEST[Q] JUMP[HLRE1] $
32 3108                                                  ;HLRE
32 3109 07371 010730170000000014141620247314560000             D[MASK 60] DEST[Q] JUMP[HLRE4] $
32 3110
32 3111 07372 540243170000060554161623254204160000             ALU[0] DEST[AC] NEOI $                                    ;HLREI
32 3112 07373 540243170000060554161623254204160000             ALU[0] DEST[AC] NEOI $
32 3113
32 3114 07374 010730170000000014141620247314560000             D[MASK 60] DEST[Q] JUMP[HLREM1] $                         ;HLREM
32 3115 07375 010730170000000014141620247314560000             D[MASK 60] DEST[Q] JUMP[HLREM4] $
32 3116
32 3117 07376 010730170000000014141620247312560000             DFRQ D[MASK 60] DEST[Q] JUMP[HLRES1] $                    ;HLRES
32 3118 07377 010730170000000014141620247314560000             D[MASK 60] DEST[Q] JUMP[HLRES4] $
32 3119
32m3120                                                   .reloc
32m3120                                                  [.USE[NORMAL]
32m3120                                                  [ xlist
32 3121 05550 010731004000000014161621254715560000        list ]                                    ]HLRE1:      D[MEM]
32 3121                                                    NORM OBUS<0 LBJUMP[HLRE2] $
32 3122 05551 010331004000000014161501054314560000      HLRE4:  ALU[MEMAC] DEST[HOLD] NORM OBUS<0 LBJUMP[HLRE2] $
32 3123 05552 010331204000000014161621254314560000      HLREM1: ALU[AC] DEST[AR] NORM OBUS<0 LBJUMP[HLREM2] $
32 3124 05553 010331204000000014161621254314560000      HLREM4: ALU[AC] DEST[AR] NORM OBUS<0 LBJUMP[HLREM3] $
32 3125 05554 010731021000133104561501254715560000      HLRES1: R-M-W D[MEM] ROT[18.] DEST[HOLD] NORM OBUS18 LBJUMP[HRRES2] $
32 3126 05555 010331204000000014161621054314560000      HLRES4: ALU[MEMAC] DEST[AR] NORM OBUS<0 LBJUMP[HLRES3] $
32m3127                                                   .pair
32m3127                                                  [:. \ 2 + .
32 3128 05556 540733170000060544445623254605160000        ]HLREZ:      D[MEM] ROT[18.] MASK[18.] DEST[AC] NEOI $
32 3129 05557 540633170000060544556162325460516000             D[MEM] ROT[18.] ALU[DORQ] DEST[AC] NEOI $
32 3130 05560 010731170000226644445560254304560000      HLREM2: IFRQ D[AR] ROT[18.] MASK[18.] DEST[MEMSTO] JUMP[SEQI] $
32 3131 05561 010631170000226644561560254304560000             IFRQ D[AR] ROT[18.] ALU[DORQ] DEST[MEMSTO] JUMP[SEQI] $
32 3132 05562 540733170000060544445623054204160000      HLREM3:      D[AR] ROT[18.] MASK[18.] DEST[MEMAC] NEOI $
32 3133 05563 540633170000060544556162305420416000             D[AR] ROT[18.] ALU[DORQ] DEST[MEMAC] NEOI $
32 3134 05564 010731370000227064444160254304560000      HLRES3: D[AR] ROT[18.] MASK[18.] DEST[AR] COND-AC-STO JUMP[MACSTO] IFRQ $
32 3134
32 3135 05565 010631370000227064560160254304560000             D[AR] ROT[18.] ALU[DORQ] DEST[AR] COND-AC-STO JUMP[MACSTO] IFRQ $
32 3135
32 3136
```

```
33 3137
33m3138                                                  .opcode[600]
33m3138                                                 [xlist
33 3139 07400 540731170000060554161623254204160000       list ]      NEOI $
33 3139                                                       ;TRN
33 3140 07401 540731170000060554161623254204160000            NEOI $
33 3141
33 3142 07402 540731170000060554161623254204160000            NEOI $
33 3142                                                  ;TLN
33 3143 07403 540731170000060554161623254204160000            NEOI $
33 3144
33 3145 07404 540541000000007355404562325760416000            D[IR] MASK[18.] ALU[D&AC] CONDSKP[ALU=0] $
33 3145                                                  ;TRNE
33 3146 07405 540541000000007355404562325760416000            D[IR] MASK[18.] ALU[D&AC] CONDSKP[ALU=0] $
33 3146
33 3147
33 3148 07406 540541000000007354454162325760416000            D[IR] ROT[18.] MASK[LEFT] ALU[D&AC] CONDSKP[ALU=0] $
33 3148                                                  ;TLNE
33 3149 07407 540541000000007354454162325760416000            D[IR] ROT[18.] MASK[LEFT] ALU[D&AC] CONDSKP[ALU=0] $
33 3149
33 3150
33 3151 07410 540731172000073554161623254204160000            CONDSKP[ALWAYS] $
33 3151                                                       ;TRNA
33 3152 07411 540731172000073554161623254204160000            CONDSKP[ALWAYS] $
33 3152
33 3153
33 3154 07412 540731172000073554161623254204160000            CONDSKP[ALWAYS] $
33 3154                                                       ;TLNA
33 3155 07413 540731172000073554161623254204160000            CONDSKP[ALWAYS] $
33 3155
33 3156
33 3157 07414 540541002000073554045623257604160000            D[IR] MASK[18.] ALU[D&AC] CONDSKP[-ALU=0] $
33 3157                                                  ;TRNN
33 3158 07415 540541002000073554045623257604160000            D[IR] MASK[18.] ALU[D&AC] CONDSKP[-ALU=0] $
33 3158
33 3159
33 3160 07416 540541002000073544541623257604160000            D[IR] ROT[18.] MASK[LEFT] ALU[D&AC] CONDSKP[-ALU=0] $
33 3160                                                  ;TLNN
33 3161 07417 540541002000073544541623257604160000            D[IR] ROT[18.] MASK[LEFT] ALU[D&AC] CONDSKP[-ALU=0] $
33 3161
33 3162
33 3163
33m3164                                                  .opcode[610]
33m3164                                                 [xlist                                  . #
33 3165 07420 540731170000060554161623254204160000       list ]      NEOI $
33 3165                                                       ;TDN
33 3166 07421 540731170000060554161623254204160000            NEOI $
33 3167
33 3168 07422 540731170000060554161623254204160000            NEOI $
33 3168                                                  ;TSN
33 3169 07423 540731170000060554161623254204160000            NEOI $
33 3170
33 3171 07424 560541000000007355416162325460316000            DF/IF D[MEM] ALU[D&AC] CONDSKP[ALU=0] $
33 3171                                                  ;TDNE
33 3172 07425 540141000000007355416162301420416000            ACSEL[MA,AC] ALU[A&B] CONDSKP[ALU=0] $
33 3173
33 3174 07426 560541000000007354456162325460316000            DF/IF D[MEM] ROT[18.] ALU[D&AC] CONDSKP[ALU=0] $
33 3174                                                  ;TSNE
33 3175 07427 010331370000000001416162005431456000            ALU[MEMAC] DEST[AR] NORM JUMP[TSNE1] $
33 3176
33 3177 07430 540731172000073554161623254204160000            CONDSKP[ALWAYS] $
33 3177                                                       ;TDNA
33 3178 07431 540731172000073554161623254204160000            CONDSKP[ALWAYS] $
33 3178
33 3179
33 3180 07432 540731172000073554161623254204160000            CONDSKP[ALWAYS] $
33 3180                                                       ;TSNA
33 3181 07433 540731172000073554161623254204160000            CONDSKP[ALWAYS] $
33 3181
33 3182
33 3183 07434 560541002000073554161623254604160000            DF/IF D[MEM] ALU[D&AC] CONDSKP[-ALU=0] $
33 3183                                                  ;TDNN
33 3184 07435 540141002000073554161623014204160000            ACSEL[MA,AC] ALU[A&B] CONDSKP[-ALU=0] $
33 3185
33 3186 07436 560541002000073544561623254604160000            DF/IF D[MEM] ROT[18.] ALU[D&AC] CONDSKP[-ALU=0] $
33 3186                                                  ;TSNN
33 3187 07437 010331370000000001416162005431456000            ALU[MEMAC] DEST[AR] NORM JUMP[TSNN1] $
33 3188
33m3189                                                  .reloc
33m3189                                                 [.USE[NORMAL]
33m3189                                                 [ xlist
33 3190 05566 540541000000007354456162325420416000       list ]                                  ]TSNE1:      D[AR] ROT[18.]
33 3190                                                 ALU[D&AC] CONDSKP[ALU=0] $
33 3191 05567 540541002000073544561623254204160000      TSNN1:  D[AR] ROT[18.] ALU[D&AC] CONDSKP[-ALU=0] $
33 3192
33m3193                                                  .opcode[620]
33m3193                                                 [xlist
33 3194 07440 540553170000060554045623257604160000       list ]      D[IR] MASK[18.] ALU[-D&AC] DEST[AC] NEOI $
33 3194                                                       ;TRZ
33 3195 07441 540553170000060554045623257604160000            D[IR] MASK[18.] ALU[-D&AC] DEST[AC] NEOI $
33 3196
33 3197 07442 540553170000060544541623257604160000            D[IR] ROT[18.] MASK[LEFT] ALU[-D&AC] DEST[AC] NEOI $
33 3197                                                  ;TLZ
33 3198 07443 540553170000060544541623257604160000            D[IR] ROT[18.] MASK[LEFT] ALU[-D&AC] DEST[AC] NEOI $
33 3199
33 3200 07444 010541002000000001404562065771456000            D[IR] MASK[18.] ALU[D&AC] COND[OBUS=0] NORM JUMPLC[TRZ1] $
33 3200                                                  ;TRZE
33 3201 07445 010541002000000001404562065771456000            D[IR] MASK[18.] ALU[D&AC] COND[OBUS=0] NORM JUMPLC[TRZ1] $
33 3201
33 3202
33 3203 07446 010541002000000000454162065771456000            D[IR] ROT[18.] MASK[LEFT] ALU[D&AC] OBUS=0 NORM JUMPLC[TLZ1] $
33 3203                                                  ;TLZE
33 3204 07447 010541002000000000454162065771456000            D[IR] ROT[18.] MASK[LEFT] ALU[D&AC] COND[OBUS=0] NORM
33 3204                                                 JUMPLC[TLZ1] $
33 3205
33 3206 07450 540553172000073554045623257604160000            D[IR] MASK[18.] ALU[-D&AC] DEST[AC] CONDSKP[ALWAYS] $
33 3206                                                  ;TRZA
33 3207 07451 540553172000073554045623257604160000            D[IR] MASK[18.] ALU[-D&AC] DEST[AC] CONDSKP[ALWAYS] $
33 3208
33 3209 07452 540553172000073544541623257604160000            D[IR] ROT[18.] MASK[LEFT] ALU[-D&AC] DEST[AC] CONDSKP[ALWAYS] $
```

```
SLOE  March 23, 1984 21:09:30  file DSK:F4INST.SLO  --  of -- F41NNF

33 3213 07455 010541000000000001404562065771456000              D[IR] MASK[18.] ALU[D&AC] COND[-OBUS=0] NORM JUMPLC[TRZ1] $
33 3213
33 3214
33 3215 07456 010541000000000000454162065771456000              D[IR] ROT[18.] MASK[LEFT] ALU[D&AC] -OBUS=0 NORM JUMPLC[TLZ1] $
33 3215                                                         ;TLZN
33 3216 07457 010541000000000000454162065771456000              D[IR] ROT[18.] MASK[LEFT] ALU[D&AC] COND[-OBUS=0] NORM
33 3216                                                        JUMPLC[TLZ1] $
33 3217
33 3218
33m3219                                                         .reloc
33m3219                                                        [.USE[NORMAL]
33m3219                                                        [ xlist
33 3220                                                          list ]                                  ]
33 3221
33 3222
33 3223 05570 540553156000073554045623257604160000      TRZ1:   D[IR] MASK[18.] ALU[-D&AC] DEST[AC] CONDSKP[LAST] $
33 3224
33 3225 05571 540553156000073544541623257604160000      TLZ1:   D[IR] ROT[18.] MASK[LEFT] ALU[-D&AC] DEST[AC] CONDSKP[LAST] $
33 3226
33m3227                                                         .opcode[630]
33m3227                                                        [xlist
33 3228 07460 560553170000060554161623254603160000              list ]      DF/IF D[MEM] ALU[-D&AC] DEST[AC] NEOI $
33 3228                                                         ;TDZ
33 3229 07461 540153170000060554161623014204160000              ACSEL[MA,AC] ALU[-A&B]  DEST[AC] NEOI $
33 3230
33 3231 07462 560553170000060544561623254603160000              DF/IF D[MEM] ROT[18.] ALU[-D&AC] DEST[AC] NEOI $
33 3231                                                        ;TSZ
33 3232 07463 010331172000000001416150045431456000              ALU[MEMAC] DEST[HOLD] NORM FALSE JUMPLC[TSZ1] $
33 3233
33 3234 07464 030541002000000001416162065471356000              DF/WT D[MEM] ALU[D&AC] COND[OBUS=0] NORM JUMPLC[TDZX1] $
33 3234                                                         ;TDZE
33 3235 07465 010141002000000001416162041431456000              ACSEL[MA,AC] ALU[A&B]  COND[OBUS=0] NORM JUMPLC[TDZX2] $
33 3235
33 3236
33 3237 07466 030541002000000000456162065471356000              DF/WT D[MEM] ROT[18.] ALU[D&AC] OBUS=0 NORM JUMPLC[TSZ1] $
33 3237                                                        ;TSZE
33 3238 07467 010331170000000001416150005431456000              ALU[MEMAC] DEST[HOLD] NORM JUMP[TSZE1] $
33 3239
33 3240 07470 560553172000073554161623254603160000              DF/IF D[MEM] ALU[-D&AC] DEST[AC] CONDSKP[ALWAYS] $
33 3240                                                        ;TDZA
33 3241 07471 540153172000073554161623014204160000              ACSEL[MA,AC] ALU[-A&B]  DEST[0]  CONDSKP[ALWAYS] $
33 3242
33 3243 07472 560553172000073544561623254603160000              DF/IF D[MEM] ROT[18.] ALU[-D&AC] DEST[AC] CONDSKP[ALWAYS] $
33 3243                                                        ;TSZA
33 3244 07473 010331170000000001416150045431456000              ALU[MEMAC] DEST[HOLD] NORM TRUE JUMPLC[TSZ1] $
33 3245
33 3246 07474 030541000000000001416162065471356000              DF/WT D[MEM] ALU[D&AC] COND[-OBUS=0] NORM JUMPLC[TDZX1] $
33 3246                                                        ;TDZN
33 3247 07475 010141000000000001416162041431456000              ACSEL[MA,AC] ALU[A&B]  COND[-OBUS=0] NORM JUMPLC[TDZX2] $
33 3247
33 3248
33 3249 07476 030541000000000000456162065471356000              DF/WT D[MEM] ROT[18.] ALU[D&AC] -OBUS=0 NORM JUMPLC[TSZ1] $
33 3249                                                        ;TSZN
33 3250 07477 010331170000000001416150005431456000              ALU[MEMAC] DEST[HOLD] NORM JUMP[TSZN1] $
33 3251
33m3252                                                         .reloc
33m3252                                                        [.USE[NORMAL]
33m3252                                                        [ xlist
33 3253                                                          list ]                                  ]
33 3254 05572 540553156000073544561623254605160000      TSZ1:   D[MEM] ROT[18.] ALU[-D&AC] DEST[AC] CONDSKP[LAST] $
33 3255
33 3256 05573 540553156000073554161623254605160000      TDZX1:  D[MEM]          ALU[-D&AC] DEST[AC] CONDSKP[LAST] $
33 3257
33 3258 05574 540153156000073554161623014204160000      TDZX2:  ACSEL[MA,AC]    ALU[-A&B]  DEST[0]  CONDSKP[LAST] $
33 3259
33 3260 05575 010541002000133644561620654715560000      TSZE1:  D[MEM] ROT[18.] ALU[D&AC] COND[OBUS=0] NORM JUMPLC[TSZ1] $
33 3261
33 3262 05576 010541000000133644561620654715560000      TSZN1:  D[MEM] ROT[18.] ALU[D&AC] COND[-OBUS=0] NORM JUMPLC[TSZ1] $
33 3263
33 3264
33m3265                                                         .opcode[640]
33m3265                                                        [xlist
33 3266 07500 540563170000060554045623257604160000              list ]      D[IR] MASK[18.] ALU[D#AC] DEST[AC] NEOI $
33 3266                                                         ;TRC
33 3267 07501 540563170000060554045623257604160000              D[IR] MASK[18.] ALU[D#AC] DEST[AC] NEOI $
33 3268
33 3269 07502 540563170000060544541623257604160000              D[IR] ROT[18.] MASK[LEFT] ALU[D#AC] DEST[AC] NEOI $
33 3269                                                        ;TLC
33 3270 07503 540563170000060544541623257604160000              D[IR] ROT[18.] MASK[LEFT] ALU[D#AC] DEST[AC] NEOI $
33 3271
33 3272 07504 010541002000000001404562065771456000              D[IR] MASK[18.] ALU[D&AC] COND[OBUS=0] NORM JUMPLC[TRC1] $
33 3272                                                        ;TRCE
33 3273 07505 010541002000000001404562065771456000              D[IR] MASK[18.] ALU[D&AC] COND[OBUS=0] NORM JUMPLC[TRC1] $
33 3273
33 3274
33 3275 07506 010541002000000000454162065771456000              D[IR] ROT[18.] MASK[LEFT] ALU[D&AC] OBUS=0 NORM JUMPLC[TLC1] $
33 3275                                                        ;TLCE
33 3276 07507 010541002000000000454162065771456000              D[IR] ROT[18.] MASK[LEFT] ALU[D&AC] COND[OBUS=0] NORM
33 3276                                                        JUMPLC[TLC1] $
33 3277
33 3278 07510 540563172000073554045623257604160000              D[IR] MASK[18.] ALU[D#AC] DEST[AC] CONDSKP[ALWAYS] $
33 3278                                                        ;TRCA
33 3279 07511 540563172000073554045623257604160000              D[IR] MASK[18.] ALU[D#AC] DEST[AC] CONDSKP[ALWAYS] $
33 3280
33 3281 07512 540563172000073544541623257604160000              D[IR] ROT[18.] MASK[LEFT] ALU[D#AC] DEST[AC] CONDSKP[ALWAYS] $
33 3281                                                        ;TLCA
33 3282 07513 540563172000073544541623257604160000              D[IR] ROT[18.] MASK[LEFT] ALU[D#AC] DEST[AC] CONDSKP[ALWAYS] $
33 3283
33 3284 07514 010541000000000001404562065771456000              D[IR] MASK[18.] ALU[D&AC] COND[-OBUS=0] NORM JUMPLC[TRC1] $
33 3284                                                        ;TRCN
33 3285 07515 010541000000000001404562065771456000              D[IR] MASK[18.] ALU[D&AC] COND[-OBUS=0] NORM JUMPLC[TRC1] $
33 3285
33 3286
33 3287 07516 010541000000000000454162065771456000              D[IR] ROT[18.] MASK[LEFT] ALU[D&AC] -OBUS=0 NORM JUMPLC[TLC1] $
33 3287                                                        ;TLCN
33 3288 07517 010541000000000000454162065771456000              D[IR] ROT[18.] MASK[LEFT] ALU[D&AC] COND[-OBUS=0] NORM
33 3288                                                        JUMPLC[TLC1] $
33 3289
33m3290                                                         .reloc
33m3290                                                        [.USE[NORMAL]
```

```
33 3295
33 3296
33m3297                                                  .opcode[650]
33m3297                                                 [xlist
33 3298 07520 560563170000060554161623254603160000       list ]      DF/IF D[MEM] ALU[D#AC] DEST[AC] NEOI $
33 3298                                                        ;TDC
33 3299 07521 540163170000060554161623014204160000            ACSEL[MA,AC] ALU[A#B]  DEST[AC] NEOI $
33 3300
33 3301 07522 560563170000060544561623254603160000            DF/IF D[MEM] ROT[18.] ALU[D#AC] DEST[AC] NEOI $
33 3301                                                 ;TSC
33 3302 07523 010331172000000001416150045431456000            ALU[MEMAC] DEST[HOLD] NORM FALSE JUMPLC[TSC1] $
33 3303
33 3304 07524 030541002000000001416162065471356000            DF/WT D[MEM] ALU[D&AC] COND[OBUS=0] NORM JUMPLC[TDCX1] $
33 3304                                                 ;TDCE
33 3305 07525 010141002000000001416162041431456000            ACSEL[MA,AC] ALU[A&B]  COND[OBUS=0] NORM JUMPLC[TDCXZ] $
33 3305
33 3306
33 3307 07526 030541002000000000456162065471356000            DF/WT D[MEM] ROT[18.] ALU[D&AC] OBUS=0 NORM JUMPLC[TSC1] $
33 3307                                                 ;TSCE
33 3308 07527 010331170000000001416150005431456000            ALU[MEMAC] DEST[HOLD] NORM JUMP[TSCE1] $
33 3309
33 3310 07530 560563172000073554161623254603160000            DF/IF D[MEM] ALU[D#AC] DEST[AC] CONDSKP[ALWAYS] $
33 3310                                                 ;TDCA
33 3311 07531 540163172000073554161623014204160000            ACSEL[MA,AC] ALU[A#B]  DEST[B]  CONDSKP[ALWAYS] $
33 3312
33 3313 07532 560563172000073544561623254603160000            DF/IF D[MEM] ROT[18.] ALU[D#AC] DEST[AC] CONDSKP[ALWAYS] $
33 3313                                                 ;TSCA
33 3314 07533 010331170000000001416150045431456000            ALU[MEMAC] DEST[HOLD] NORM TRUE JUMPLC[TSC1] $
33 3315
33 3316 07534 030541000000000001416162065471356000            DF/WT D[MEM] ALU[D&AC] COND[-OBUS=0] NORM JUMPLC[TDCX1] $
33 3316                                                 ;TDCN
33 3317 07535 010141000000000001416162041431456000            ACSEL[MA,AC] ALU[A&B]  COND[-OBUS=0] NORM JUMPLC[TDCXZ] $
33 3317
33 3318
33 3319 07536 030541000000000000456162065471356000            DF/WT D[MEM] ROT[18.] ALU[D&AC] -OBUS=0 NORM JUMPLC[TSC1] $
33 3319                                                 ;TSCN
33 3320 07537 010331170000000001416150005431456000            ALU[MEMAC] DEST[HOLD] NORM JUMP[TSCN1] $
33 3321
33m3322                                                  .reloc
33m3322                                                 [.USE[NORMAL]
33m3322                                                 [ xlist
33 3323                                                   list ]                                    ]
33 3324 05601 540563156000073544561623254605160000    TSC1:   D[MEM] ROT[18.] ALU[D#AC] DEST[AC] CONDSKP[LAST] $
33 3325
33 3326 05602 540563156000073554161623254605160000    TDCX1:  D[MEM]          ALU[D#AC] DEST[AC] CONDSKP[LAST] $
33 3327
33 3328 05603 540163156000073554161623014204160000    TDCXZ:  ACSEL[MA,AC]    ALU[A#B]  DEST[B]  CONDSKP[LAST] $
33 3329
33 3330 05604 010541002000134024561620654715560000    TSCE1:  D[MEM] ROT[18.] ALU[D&AC] COND[OBUS=0] NORM JUMPLC[TSC1] $
33 3331
33 3332 05605 010541000000134024561620654715560000    TSCN1:  D[MEM] ROT[18.] ALU[D&AC] COND[-OBUS=0] NORM JUMPLC[TSC1] $
33 3333
33 3334
33 3335
33m3336                                                  .opcode[660]
33m3336                                                 [xlist
33 3337 07540 540533170000060554045623257604160000       list ]      D[IR] MASK[18.] ALU[DORAC] DEST[AC] NEOI $
33 3337                                                        ;TRO
33 3338 07541 540533170000060554045623257604160000            D[IR] MASK[18.] ALU[DORAC] DEST[AC] NEOI $
33 3339
33 3340 07542 540533170000060544541623257604160000            D[IR] ROT[18.] MASK[LEFT] ALU[DORAC] DEST[AC] NEOI $
33 3340                                                 ;TLO
33 3341 07543 540533170000060544541623257604160000            D[IR] ROT[18.] MASK[LEFT] ALU[DORAC] DEST[AC] NEOI $
33 3342
33 3343 07544 010541002000000001404562065771456000            D[IR] MASK[18.] ALU[D&AC] COND[OBUS=0] NORM JUMPLC[TRO1] $
33 3343                                                 ;TROE
33 3344 07545 010541002000000001404562065771456000            D[IR] MASK[18.] ALU[D&AC] COND[OBUS=0] NORM JUMPLC[TRO1] $
33 3344
33 3345
33 3346 07546 010541002000000000454162065771456000            D[IR] ROT[18.] MASK[LEFT] ALU[D&AC] OBUS=0 NORM JUMPLC[TLO1] $
33 3346                                                 ;TLOE
33 3347 07547 010541002000000000454162065771456000            D[IR] ROT[18.] MASK[LEFT] ALU[D&AC] COND[OBUS=0] NORM
33 3347                                                JUMPLC[TLO1] $
33 3348
33 3349 07550 540533172000073554045623257604160000            D[IR] MASK[18.] ALU[DORAC] DEST[AC] CONDSKP[ALWAYS] $
33 3349                                                 ;TROA
33 3350 07551 540533172000073554045623257604160000            D[IR] MASK[18.] ALU[DORAC] DEST[AC] CONDSKP[ALWAYS] $
33 3351
33 3352 07552 540533172000073544541623257604160000            D[IR] ROT[18.] MASK[LEFT] ALU[DORAC] DEST[AC] CONDSKP[ALWAYS] $
33 3352                                                 ;TLOA
33 3353 07553 540533172000073544541623257604160000            D[IR] ROT[18.] MASK[LEFT] ALU[DORAC] DEST[AC] CONDSKP[ALWAYS] $
33 3354
33 3355 07554 010541000000000001404562065771456000            D[IR] MASK[18.] ALU[D&AC] COND[-OBUS=0] NORM JUMPLC[TRO1] $
33 3355                                                 ;TRON
33 3356 07555 010541000000000001404562065771456000            D[IR] MASK[18.] ALU[D&AC] COND[-OBUS=0] NORM JUMPLC[TRO1] $
33 3356
33 3357
33 3358 07556 010541000000000000454162065771456000            D[IR] ROT[18.] MASK[LEFT] ALU[D&AC] -OBUS=0 NORM JUMPLC[TLO1] $
33 3358                                                        ;TLON
33 3359 07557 010541000000000000454162065771456000            D[IR] ROT[18.] MASK[LEFT] ALU[D&AC] COND[-OBUS=0] NORM
33 3359                                                JUMPLC[TLO1] $
33 3360
33 3361
33m3362                                                  .reloc
33m3362                                                 [.USE[NORMAL]
33m3362                                                 [ xlist
33 3363                                                   list ]                                    ]
33 3364 05606 540533156000073554045623257604160000    TRO1:   D[IR] MASK[18.] ALU[DORAC] DEST[AC] CONDSKP[LAST] $
33 3365
33 3366 05607 540533156000073544541623257604160000    TLO1:   D[IR] ROT[18.] MASK[LEFT] ALU[DORAC] DEST[AC] CONDSKP[LAST] $
33 3367
33m3368                                                  .opcode[670]
33m3368                                                 [xlist
33 3369 07560 560533170000060554161623254603160000       list ]      DF/IF D[MEM] ALU[DORAC] DEST[AC] NEOI $
33 3369                                                        ;TDO
33 3370 07561 540133170000060554161623014204160000            ACSEL[MA,AC] ALU[AORB]  DEST[AC] NEOI $
33 3371
33 3372 07562 560533170000060544561623254603160000            DF/IF D[MEM] ROT[18.] ALU[DORAC] DEST[AC] NEOI $
33 3372                                                 ;TSO
33 3373 07563 010331172000000001416150045431456000            ALU[MEMAC] DEST[HOLD] NORM FALSE JUMPLC[TSO1] $
```

```
33 3377
33 3378 07566 030541002000000000456162065471356000            DF/WT D[MEM] ROT[18.] ALU[D&AC] OBUS=0 NORM JUMPLC[TSO1] $
33 3378                                                  ;TSOE
33 3379 07567 010331170000000001416150005431456000            ALU[MEMAC] DEST[HOLD] NORM JUMP[TSOE1] $
33 3380
33 3381 07570 560533172000073554161623254603160000            DF/IF D[MEM] ALU[DORAC] DEST[AC] CONDSKP[ALWAYS] $
33 3381                                                  ;TDOA
33 3382 07571 540133172000073554161623014204160000            ACSEL[MA,AC] ALU[AORB]  DEST[B]  CONDSKP[ALWAYS] $
33 3383
33 3384 07572 560533172000073544561623254603160000            DF/IF D[MEM] ROT[18.] ALU[DORAC] DEST[AC] CONDSKP[ALWAYS] $
33 3384                                                  ;TSOA
33 3385 07573 010331170000000001416150045431456000            ALU[MEMAC] DEST[HOLD] NORM TRUE JUMPLC[TSO1] $
33 3386
33 3387 07574 030541000000000001416162065471356000            DF/WT D[MEM] ALU[D&AC] COND[-OBUS=0] NORM JUMPLC[TDOX1] $
33 3387                                                  ;TDON
33 3388 07575 010141000000000001416162041431456000            ACSEL[MA,AC] ALU[A&B]  COND[-OBUS=0] NORM JUMPLC[TDOX2] $
33 3388
33 3389
33 3390 07576 030541000000000004561620654713560000            DF/WT D[MEM] ROT[18.] ALU[D&AC] -OBUS=0 NORM JUMPLC[TSO1] $
33 3390                                                  ;TSON
33 3391 07577 010331170000000001416150005431456000            ALU[MEMAC] DEST[HOLD] NORM JUMP[TSON1] $
33 3392
33 3393                                                   .reloc
33 3393                                                  [.USE[NORMAL]
33 3393                                                  [ xlist
33 3394                                                    list ]                                  ]
33 3395 05610 540533156000073544561623254605160000    TSO1:   D[MEM] ROT[18.] ALU[DORAC] DEST[AC] CONDSKP[LAST] $
33 3396
33 3397 05611 540533156000073554161623254605160000    TDOX1:  D[MEM]          ALU[DORAC] DEST[AC] CONDSKP[LAST] $
33 3398
33 3398 05612 540133156000073554161623014204160000    TDOX2:  ACSEL[MA,AC]    ALU[AORB]  DEST[B]  CONDSKP[LAST] $
33 3400
33 3401 05613 010541002000134204561620654715560000    TSOE1:  D[MEM] ROT[18.] ALU[D&AC] COND[OBUS=0] NORM JUMPLC[TSO1] $
33 3402
33 3403 05614 010541000000134204561620654715560000    TSON1:  D[MEM] ROT[18.] ALU[D&AC] COND[-OBUS=0] NORM JUMPLC[TSO1] $
33 3404
33 3405
33 3406                                                SET-HALF: ;Set the First Part Done flag (PC FLAGS bit 4).
33 3407 05615 010730570000060554161623646314160000            D[PC-FLAGS]  DEST[ALU1 Q] NORM $
33 3408 05616 010631570000060547602424255714170000            D[CONST 1] ROT[35. - 4] ALU[ALU1 DORQ] DEST[PC-FLAGS] NORM POPJ $
33 3408
```

```
01 0064                                                 ;;The CPU instrs.
01 0065
01 0066                                         .repeat AUGGYF [ .INSERT F4AUG.SLO ]
01 0067                                                 ;;the augment instructions
```

SLOE March 23, 1984 21:09:52 file DSK:F41NNF.SLO -- of -- F41NNF

02 0009   .INSERT F41BS

```
01 0001
01 0002
01m0003                                         .use[HILOC]
01m0003                                          [ xlist
01 0004                                            list ]
01 0005                                         .insert F4FLOT
```

```
01 0001                                                 ;-------------------------------------------------------------------------------
01 0001                                                 ------
01 0002                                                 ;
01 0003                                                 ;       Single Procession Floating Point
01 0004                                                 ;
01 0005                                                 ;-------------------------------------------------------------------------------
01 0005                                                 ------
01 0006
01m0007                                                   .opcode[130]   [xlist
01 0007                                                    list  ];UFA
01 0008 06260 010731170000000001416162025431256000     UFA:    DFRQ JUMP[UFA1] $
01 0009 06261 010331170000000001416150005431456000             ALU[MEMAC] DEST[HOLD] JUMP[UFA1] $
01 0010
01m0011                                                   .opcode[131]   [xlist
01 0011                                                    list  ];DFN
01 0012 06262 010731170000000001416162025431256000     DFN:    DFRQ JUMP[DFN0] $
01 0013 06263 010331170000000001416150005431456000             ALU[MEMAC] DEST[HOLD] JUMP[DFN0] $
01 0014
01m0015                                                   .opcode[132]   [xlist
01 0015                                                    list  ];FSC
01 0016 06264 010541000000000001406762124721456000     FSC:    D[MASK 27.] ALU[D&AC]  COND[-OBUS=0] LBJUMP[FSCZAP] C550 $
01 0017 06265 010541000000000001406762124721456000             D[MASK 27.] ALU[D&AC]  COND[-OBUS=0] LBJUMP[FSCZAP] C550 $
01 0018
01 0019                                                 ;
01 0020                                                 ;FAD FADL FADM FADB FADR FADRI FADRM FADRB
01 0021                                                 ;
01 0022
01m0023                                                   .opcode[140]   [xlist
01 0023                                                    list  ]      ;Opcodes 140-147
01 0024 06300 010731170000000001416162025431256000             DFRQ JUMP[FAOS1] $
01 0025 06301 010331170000000001416150005431456000             ALU[MEMAC] DEST[HOLD] JUMP[FAOS1] $
01 0026
01 0027 06302 010731170000000001416162025431256000             DFRQ JUMP[FAOS2] $
01 0028 06303 010331170000000001416150005431456000             ALU[MEMAC] DEST[HOLD] JUMP[FAOS2] $
01 0029
01 0030 06304 030731171000000001416162025471356000             DF/WT R-M-W D[MEM] JUMP[FAOS3] $
01 0031 06305 010331170000000001416150005431456000             ALU[MEMAC] DEST[HOLD] JUMP[FAOS3] $
01 0032
01 0033 06306 030731171000000001416162025471356000             DF/WT R-M-W D[MEM] JUMP[FAOS4] $
01 0034 06307 010331170000000001416150005431456000             ALU[MEMAC] DEST[HOLD] JUMP[FAOS4] $
01 0035
01 0036 06310 010731170000000001416162025431256000             DFRQ JUMP[FAOS1] $
01 0037 06311 010331170000000001416150005431456000             ALU[MEMAC] DEST[HOLD] JUMP[FAOS1] $
01 0038
01 0039 06312 010731170000000000454150025671456000             D[MA] ROT[18.] DEST[HOLD] MASK[LEFT] JUMP[FAOSS] NORM $
01 0040 06313 010731170000000000454150025671456000             D[MA] ROT[18.] DEST[HOLD] MASK[LEFT] JUMP[FAOSS] NORM $
01 0041
01 0042 06314 030731171000000001416162025471356000             DF/WT R-M-W D[MEM] JUMP[FAOS3] $
01 0043 06315 010331170000000001416150005431456000             ALU[MEMAC] DEST[HOLD] JUMP[FAOS3] $
01 0044
01 0045 06316 030731171000000001416162025471356000             DF/WT R-M-W D[MEM] JUMP[FAOS4] $
01 0046 06317 010331170000000001416150005431456000             ALU[MEMAC] DEST[HOLD] JUMP[FAOS4] $
01 0047
01 0048                                                 ;
01 0049                                                 ;FSB FSBL FSBM FSBB FSBR FSBRI FSBRM FSBRB
01 0050                                                 ;
01m0051                                                   .opcode[150]
01m0051                                                  [xlist
01 0052 06320 031711170000000001416150025471356000       list  ]      DF/WT D[MEM] ALU[0-D] DEST[HOLD] JUMP[FAOS1] $
01 0053 06321 011321170000000001416150005431456000             ALU[0-MEMAC] DEST[HOLD] JUMP[FAOS1] $
01 0054
01 0055 06322 031711170000000001416150025471356000             DF/WT D[MEM] ALU[0-D] DEST[HOLD]  JUMP[FAOS2] $
01 0056 06323 011321170000000001416150005431456000             ALU[0-MEMAC] DEST[HOLD] JUMP[FAOS2] $
01 0057
01 0058 06324 031711171000000001416150025471356000             DF/WT R-M-W  D[MEM] ALU[0-D] DEST[HOLD] JUMP[FAOS3] $
01 0059 06325 011321170000000001416150005431456000             ALU[0-MEMAC] DEST[HOLD] JUMP[FAOS3] $
01 0060
01 0061 06326 031711171000000001416150025471356000             DF/WT R-M-W D[MEM] ALU[0-D] DEST[HOLD]  JUMP[FAOS4] $
01 0062 06327 011321170000000001416150005431456000             ALU[0-MEMAC] DEST[HOLD] JUMP[FAOS4] $
01 0063
01 0064 06330 031711170000000001416150025471356000             DF/WT  D[MEM] ALU[0-D] DEST[HOLD] JUMP[FAOS1] $
01 0065 06331 011321170000000001416150005431456000             ALU[0-MEMAC] DEST[HOLD] JUMP[FAOS1] $
01 0066
01 0067 06332 011711170000000000454150025671456000             D[MA] ROT[18.] MASK[LEFT] ALU[0-D] DEST[HOLD] JUMP[FAOSS] NORM $
01 0067
01 0068 06333 011711170000000000454150025671456000             D[MA] ROT[18.] MASK[LEFT] ALU[0-D] DEST[HOLD] JUMP[FAOSS] NORM $
01 0068
01 0069
01 0070 06334 031711171000000001416150025471356000             DF/WT R-M-W D[MEM] ALU[0-D] DEST[HOLD]  JUMP[FAOS3] $
01 0071 06335 011321170000000001416150005431456000             ALU[0-MEMAC] DEST[HOLD] JUMP[FAOS3] $
01 0072
01 0073 06336 031711171000000001416150025471356000             DF/WT R-M-W D[MEM] ALU[0-D] DEST[HOLD]  JUMP[FAOS4] $
01 0074 06337 011321170000000001416150005431456000             ALU[0-MEMAC] DEST[HOLD] JUMP[FAOS4] $
01 0075
01 0076
01 0077                                                 ;
01 0078                                                 ;FMP FMPL FMPM FMPB FMPR FMPRI FMPRM FMPRB
01 0079                                                 ;
01m0080                                                   .opcode[160]  [xlist
01 0080                                                   list  ]      ;Opcodes 160-167
01 0081 06340 010731170000000001416162025431256000             DFRQ JUMP[FMP1] $
01 0082 06341 010331170000000001416150005431456000             ALU[MEMAC] DEST[HOLD] JUMP[FMP1] $
01 0083
01 0084 06342 010731170000000001416162025431256000             DFRQ JUMP[FMP2] $
01 0085 06343 010331170000000001416150005431456000             ALU[MEMAC] DEST[HOLD] JUMP[FMP2] $
01 0086
01 0087 06344 030731171000000001416162025471356000             DF/WT R-M-W D[MEM] JUMP[FMP3] $
01 0088 06345 010331170000000001416150005431456000             ALU[MEMAC] DEST[HOLD] JUMP[FMP3] $
01 0089
01 0090 06346 030731171000000001416162025471356000             DF/WT R-M-W D[MEM] JUMP[FMP4] $
01 0091 06347 010331170000000001416150005431456000             ALU[MEMAC] DEST[HOLD] JUMP[FMP4] $
01 0092
01 0093 06350 010731170000000001416162025431256000             DFRQ JUMP[FMP1] $
01 0094 06351 010331170000000001416150005431456000             ALU[MEMAC] DEST[HOLD] JUMP[FMP1] $
01 0095
01 0096 06352 010731170000000000454150025671456000             D[MA] ROT[18.] DEST[HOLD] MASK[LEFT] JUMP[FMPS] NORM $
01 0097 06353 010731170000000000454150025671456000             D[MA] ROT[18.] DEST[HOLD] MASK[LEFT] JUMP[FMPS] NORM $
01 0098
01 0099 06354 030731171000000001416162025471356000             DF/WT R-M-W D[MEM] JUMP[FMP3] $
01 0100 06355 010331170000000001416150005431456000             ALU[MEMAC] DEST[HOLD] JUMP[FMP3] $
```

```
01 0106                                                 ;FDV FDVL FDVM FDVB FDVR FDVRI FDVRM FDVRB
01 0107
01m0108                                                   .opcode[170]   [xlist
01 0108                                                    list  ]      ;Opcodes 170-177
01 0109
01 0110 06360 010731170000030000141616202543125G6000               DFRQ JUMP[FDV1] $
01 0111 06361 010331170000000001416150005431456000                 ALU[MEMAC] DEST[HOLD] JUMP[FDV1] $
01 0112
01 0113 06362 010731170000000001416162025431256000                 DFRQ JUMP[FDV2] $
01 0114 06363 010331170000000001416150005431456000                 ALU[MEMAC] DEST[HOLD] JUMP[FDV2] $
01 0115
01 0116 06364 030731171000000001416162025471356000                 DF/WT R-M-W D[MEM] JUMP[FDV3] $
01 0117 06365 010331170000000001416150005431456000                 ALU[MEMAC] DEST[HOLD] JUMP[FDV3] $
01 0118
01 0119 06366 030731171000000001416162025471356000                 DF/WT R-M-W D[MEM] JUMP[FDV4] $
01 0120 06367 010331170000000001416150005431456000                 ALU[MEMAC] DEST[HOLD] JUMP[FDV4] $
01 0121
01 0122 06370 010731170000000001416162025431256000                 DFRQ JUMP[FDV1] $
01 0123 06371 010331170000000001416150005431456000                 ALU[MEMAC] DEST[HOLD] JUMP[FDV1] $
01 0124
01 0125 06372 010731170000000000454150025671456000                 D[MA] ROT[18.] DEST[HOLD] MASK[LEFT] JUMP[FDV1] NORM $
01 0126 06373 010731170000000000454150025671456000                 D[MA] ROT[18.] DEST[HOLD] MASK[LEFT] JUMP[FDV1] NORM $
01 0127
01 0128 06374 030731171000000001416162025471356000                 DF/WT R-M-W D[MEM] JUMP[FDV3] $
01 0129 06375 010331170000000001416150005431456000                 ALU[MEMAC] DEST[HOLD] JUMP[FDV3] $
01 0130
01 0131 06376 030731171000000001416162025471356000                 DF/WT R-M-W D[MEM] JUMP[FDV4] $
01 0132 06377 010331170000000001416150005431456000                 ALU[MEMAC] DEST[HOLD] JUMP[FDV4] $
01 0133
01 0134
01m0135                                                   .reloc
01m0135                                                   [.USE[HILOC]
01m0135                                                   [ xlist
01 0136                                                     list ]                                ]
01 0137
```

```
02 0138                                                 ;FLOATING POINT, KA10 STYLE.
02 0139                                                 ;NONDISPATCH CODE.
02 0140                                                 .DEFINE CFPLOW(,) [D(AR) ACSEL(AC+1) DEST(AC) JUMP(FPLOW) NORM ]
02 0141                                                 .DEFINE JINSEXP(,) [D(AR) ROT(27.) DEST(Q) JUMP(INSEX1) NORM ]
02 0142                                                 .DEFINE CINSEXP(,) [D(AR) ROT(27.) DEST(Q) PUSHJ(INSEX1) NORM ]
02 0143                                                 .DEFINE CFDS(,) [D(MEM) ALU(D#AC) DEST(AR) PUSHJ(FDS1) NORM]
02 0144
02 0145
02 0146                                                 ;***** DON'T USE A-MEM 0 FOR ANYTHING BESIDES DISPATCHES!!!      TVR-Sep80
02 0146                                                   *****
02 0147                                                 ;***** This must be fixed!!!!                                    TVR-Sep80
02 0147                                                   *****
02 0148 11552 010731170041060554022463655571417000     UFA1:   D(CONST 11) DEST(DEV-ADR) $
02 0149 11553 010331370000000001416100225431456000             ALU(AC) DEST(D%AMEM0 AR) PUSHJ(FADSUB) NORM $
02 0150 11554 010732370000060554161623660374160000             D(D%AMEM0) DEST(AR 0-AC) SHORT $
02 0151 11555 740733170000060554161623154204160000             D(AR) ACSEL(AC+1) DEST(AC MA) NEOI $
02 0152
02 0153 11556 011711203000000001406762125451556000     DFN0:   R-M-W D(MEM) MASK(27.) ALU(0-D) DEST(AR) COND(OBUS=0)
                                                         LBJUMP(DFN1) C600 $
                                                          .pair
                                                         [:. \ Z + .
02 0155 11560 010373170000233454161620254314560000        ]DFN1:       ALU(NOTAC) DEST(AC) JUMP(. + 2) NORM $
02 0156 11561 011323170000060554161623654374160000             ALU(0-AC) DEST(AC) SHORT $
02 0157 11562 010730170000060554161623654775160000             D(MEM) DEST(Q) SHORT $
02 0158 11563 010640170000060546623623647374160000             D(MASK 9) ROT(27.) ALU(D&Q) DEST(Q) SHORT $
02 0159 11564 010631310000226654067561254304560000             D(AR) MASK(27.) ALU(DORQ) SMAC  [ IFRQ DEST(MEMSTO AR)
                                                         COND(MA<20) LBJUMP(SEOI) ]$

                                                          .pair
                                                         [:. \ Z + .
02 0162 11566 740243170000060554161623254204160000        ]FSC2AP:     ALU(0) DEST(MA AC) NEOI $
02 0163 11567 010331370000060554161623654314160000             ALU(AC) DEST(AR) NORM $
02 0164 11570 010730370000060542223623654374160000             D(AR) MASK(9) ROT(9) DEST(Q AR) SHORT $
02 0165 11571 010553006000000000662162024721456000             D(MASK 8) ROT(27.) ALU(-D&AC) DEST(AC) CONT
                                                         C550 $
02 0166 11572 010660370000060554023623647374160000             D(MASK 9) ALU(D#Q) DEST(Q AR) SHORT $
02 0167 11573 010533170000060546621623647374160000             D(MASK 8) ROT(27.) ALU(DORAC) DEST(AC) SHORT $
02 0168 11574 610601170000060554045443656774160000     FSC2:   D(MA) MASK(18.) ALU(D+Q) DEST(IR-ADR) SHORT $
02 0169 11575 010240170000000001416162225431456000             ALU(0) DEST(Q) PUSHJ(NRMLIZ) NORM $
02 0170 11576 010730170000000006761622254314560000             CINSEXP  [D(AR) ROT(27.) DEST(Q) PUSHJ(INSEX1) NORM ]$
02 0171 11577 540731170000060554161623254204160000             NEOI $
02 0172
02 0173                                                 FAOS1:
02 0174                                                 FAOS5:
02 0175 11600 010331370000000001416162225431456000             ALU(AC) DEST(AR) PUSHJ(FADSUB) NORM $
02 0176 11601 540731170000060554161623254204160000             NEOI $
02 0177 11602 010331370000000001416162225431456000     FAOS2:  ALU(AC) DEST(AR) PUSHJ(FADSUB) NORM $
02 0178 11603 010733170000000001416162015431456000             CFPLOW  [D(AR) ACSEL(AC+1) DEST(AC) JUMP(FPLOW) NORM ]$
02 0179 11604 010331370000060554161623654314160000     FAOS3:  ALU(AC) DEST(AR) NORM $
02 0180 11605 010731170041060554022463655774170000             D(CONST 11) DEST(DEV-ADR) SHORT $
02 0181 11606 010331170000000001416100225431456000             ALU(AC) DEST(D%AMEM0) PUSHJ(FADSUB) NORM $
02 0182 11607 010732310000226654161561256030456000             D(D%AMEM0) DEST(0-AC) SMAC  [ IFRQ DEST(MEMSTO AR) COND(MA<20)
                                                         LBJUMP(SEOI) ]$
02 0183 11610 010331370000000001416162225431456000     FAOS4:  ALU(AC) DEST(AR) PUSHJ(FADSUB) NORM $
02 0184 11611 010331310000226654161561254304560000             ALU(AC) SMAC  [ IFRQ DEST(MEMSTO AR) COND(MA<20) LBJUMP(SEOI) ]$
02 0184
02 0185
02 0186                                                 FMP1:
02 0187                                                 FMP5:
02 0188 11612 010331370000000001416162225431456000             ALU(AC) DEST(AR) PUSHJ(FM) NORM $
02 0189 11613 540731170000060554161623254204160000             NEOI $
02 0190 11614 010331370000000001416162225431456000     FMP2:   ALU(AC) DEST(AR) PUSHJ(FM) NORM $
02 0191 11615 010733170000000001416162015431456000             CFPLOW  [D(AR) ACSEL(AC+1) DEST(AC) JUMP(FPLOW) NORM ]$
02 0192 11616 010331370000060554161623654314160000     FMP3:   ALU(AC) DEST(AR) NORM $
02 0193 11617 010731170041060554022463655774170000             D(CONST 11) DEST(DEV-ADR) SHORT $
02 0194 11620 010331170000000001416100225431456000             ALU(AC) PUSHJ(FM) DEST(D%AMEM0) NORM $
02 0195 11621 010732310000226654161561256030456000             D(D%AMEM0) DEST(0-AC) SMAC  [ IFRQ DEST(MEMSTO AR) COND(MA<20)
                                                         LBJUMP(SEOI) ]$
02 0196 11622 010331370000000001416162225431456000     FMP4:   ALU(AC) DEST(AR) PUSHJ(FM) NORM $
02 0197 11623 010331310000226654161561254304560000             ALU(AC) SMAC  [ IFRQ DEST(MEMSTO AR) COND(MA<20) LBJUMP(SEOI) ]$
02 0197
02 0198
02 0199                                                 ;Macro to test for divide by zero
02 0200                                                 .DEFINE DIVTST(,) [D(MEM) MASK(27.) COND(OBUS=0) JUMP(FPNDIV) C550]
02 0201
02 0202                                                 ;;;FD5: D(MA) ROT(18.) DEST(HOLD) SPEC(LEFT) NORM $     ;Not needed.
02 0202                                                  TVR-Apr80
02 0203
02 0204                                                 ;FDV,FDVR
02 0205 11624 010731170041060554022463655571417000     FDV1:   D(CONST 11) DEST(DEV-ADR) $
02 0206 11625 010731002000000001406762025461556000     FD1:    DIVTST  [D(MEM) MASK(27.) COND(OBUS=0) JUMP(FPNDIV) C550]$
02 0206                                                  ;Check for divide by zero
02 0207 11626 010561370000000001416162225471556000             CFDS  [D(MEM) ALU(D#AC) DEST(AR) PUSHJ(FDS1) NORM]$
02 0208 11627 540731170000060554161623254204160000             NEOI $
02 0209
02 0210                                                 ;-----------------------------------------------------------------
02 0210                                                 ------
02 0211                                                 ;LONG MODE FLOATING DIVIDE.
02 0212                                                 ;-----------------------------------------------------------------
02 0212                                                 ------
02 0213 11630 010731170041060554022463655571417000     FDV2:   D(CONST 11) DEST(DEV-ADR) $
02 0214 11631 010731002000000001406762025461556000             DIVTST  [D(MEM) MASK(27.) COND(OBUS=0) JUMP(FPNDIV) C550]$
02 0215                                                  ;AR.0,IR.35:=XOR(DIVIDEND SIGN,DIVISOR SIGN).
02 0216 11632 010561370000060554161623654775160000             D(MEM) ALU(D#AC) DEST(AR) SHORT $
02 0217 11633 610731170000060540203443654374160000             D(AR) MASK(1) ROT(1) DEST(IR-ADR) SHORT $
02 0218                                                  ;Flush sign & exponent from Dividend Low
02 0219 11634 010541370000060554067623547374160000             D(MASK 27.) ALU(D&AC) ACSEL(AC+1) DEST(AR) SHORT $
02 0220 11635 010733170000060542361623654374160000             D(AR) ROT(9) DEST(AC) ACSEL(AC+1) SHORT $
02 0221                                                  ;Positive form of Dividend to AC, AR; original form to AMEM[1]
02 0222 11636 010331206000000001416102025411456000             ALU(AC) DEST(AR D%AMEM1) COND(-OBUS<0) JUMP(FD2A) CYLEN(C450) $
02 0223 11637 011323002000235034161620154214560000             ALU(0-AC) ACSEL(AC+1) DEST(AC) COND(OBUS=0) JUMP(. + 2) C550 $
02 0224 11640 010373370000235054161620254374560000             ALU(NOTAC) DEST(AC AR) JUMP(. + 2) SHORT $
02 0225 11641 011323370000060554161623654314160000             ALU(0-AC) DEST(AC AR) NORM $
02 0226 11642 010501370000060546603623655714160000     FD2A:   D(CONST 1) ROT(27.) ALU(AC+D) DEST(AR) NORM $
02 0227                                                           ;Add 1 to exp. of dividend, since we will do only a 27-bit
                                                         divide.
02 0228 11643 010331170000060554161143554314160000             ALU(AC) ACSEL(AC+1) DEST(D%AMEM6) NORM $
02 0229                                                           ;Put low-order mantissa in AMEM[6] for FD.
02 0230 11644 010721170000060554001363655571416000             ALU(-1) DEST(D%AMEM17) $
02 0231                                                           ;Flag t do only a 27-bit divide, so remainder will come out
```

```
02 0236                                                  ;Calculate exponent of remainder.
02 0237 11647 010730006000235254161620260614560000             D[D%AMEM1]  DEST[Q] C550 -OBUS<0 JUMP[. + 3] $
02 0238                                                          ;Check sign of dividend, jump if positive.
02 0239 11650 011221170000060554161023654314160000             ALU[0-Q] DEST[D%AMEM1] NORM $
02 0240                                                          ;Negate the dividend (so the exponent will be in pos. form)
02 0241 11651 011323170000060554161623554314160000             ACSEL[AC+1] ALU[0-AC] DEST[AC] NORM $
02 0242                                                          ;Negate the remainder.
02 0243 11652 010730170000060542ZZ16236607141600O               D[D%AMEM1] ROT[9.] MASK[8] DEST[Q] NORM $
02 0244                                                          ;Recover exp. of dividend.
02 0245                                                  ;A-MEM[?] has (dividend MOD)-(divisor).
02 0246 11653 010731006000235334161620263614560000             D[D%AMEM?] COND[-OBUS<0] JUMP[. + 2] C550 $
02 0247 11654 011610170000060554003623655774160000              D[CONST 1] ALU[Q-D] DEST[Q] SHORT $
02 0248 11655 011610020000000001406562025551456000             D[CONST 26.] ALU[Q-D] DEST[Q] COND[OBUS18] JUMP[FDZC] C600 $
02 0249                                                          ;Jump if exponent underflow.
02 0250                                                  .DEFINE SWAPAC[] [ ;Interchange (AC) and (AC+1)
02 0251                                                         ALU[AC] ACSEL[AC+1] DEST[AR] SHORT $
02 0252                                                         D[AR] DEST[AR 0-AC] SHORT $
02 0253                                                         D[AR] ACSEL[AC+1] DEST[AC] SHORT $
02 0254                                                   ]
02 0255
02 0256                                                  ;SWAP QUOTIENT (AC) AND REMAINDER (AC+1)
02m0257                                                         SWAPAC
02m0257                                                  [ ;Interchange (AC) and (AC+1)
02m0257 11656 010331370000060554161623554374160000             ALU[AC] ACSEL[AC+1] DEST[AR] SHORT $
02m0257 11657 010732370000060554161623654374160000             D[AR] DEST[AR 0-AC] SHORT $
02m0257 11660 010733170000060554161623554374160000             D[AR] ACSEL[AC+1] DEST[AC] SHORT $
02 0258                                                   ] ;INSERT EXPONENT INTO REMAINDER.
02 0259 11661 010231370000000001416162225431456000             ALU[Q] DEST[AR] PUSHJ[INSEXP] NORM $
02m0260                                                         SWAPAC
02m0260                                                  [ ;Interchange (AC) and (AC+1)
02m0260 11662 010331370000060554161623554374160000             ALU[AC] ACSEL[AC+1] DEST[AR] SHORT $
02m0260 11663 010732370000060554161623654374160000             D[AR] DEST[AR 0-AC] SHORT $
02m0260 11664 010733170000060554161623554374160000             D[AR] ACSEL[AC+1] DEST[AC] SHORT $
02 0261 11665 540731170000060554161623254204160000      ]FDZD:         NEOI $
02 0262
02 0263 11666 540243170000060554161623154204160000      FDZC:   ALU[0] ACSEL[AC+1] DEST[AC] NEOI $
02 0264
02 0265 11667 010731170040060554022463655714170000      FDV3:   D[CONST 11] DEST[DEV-ADR] $
02 0266 11670 010731002000000014067620254615560000      FD3:    DIVTST  [D[MEM] MASK[27.] COND[OBUS=0] JUMP[FPNDIV] C550]$
02 0267 11671 010331170000000014161002254314560000             ALU[AC] DEST[D%AMEM0] PUSHJ[FDS] NORM $
02 0268 11672 010732310000226654161561260304560000             D[D%AMEM0] DEST[0-AC] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<Z0]
02 0268                                                  LBJUMP[SEQI] ]$
02 0269
02 0270 11673 010731170040060554022463655714170000      FDV4:   D[CONST 11] DEST[DEV-ADR] $
02 0271 11674 010731002000000014067620254615560000      FD4:    DIVTST  [D[MEM] MASK[27.] COND[OBUS=0] JUMP[FPNDIV] C550]$
02 0272 11675 010561370000000001416162225471555000             CFDS  [D[MEM] ALU[D*AC] DEST[AR] PUSHJ[FDS1] NORM]$
02 0273 11676 010331310000226654161561254304560000             ALU[AC] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<Z0] LBJUMP[SEQI] ]$
02 0273
02 0274
02 0275                                                  ;HERE ON DIVISOR=0.  SET NO DIVIDE, OVF,FLOV,RETURN.
02 0276 11677 010730170000060554161623646374160000      FPNDIV: D[PC-FLAGS] DEST[Q] SHORT $
02 0277 11700 010630170000000005603622255571456000             D[CONST 1] ROT[23.] ALU[DORQ] DEST[Q] PUSHJ[SETFOV] NORM $
02 0278 11701 540731170000060554161623254204160000             NEOI $
02 0279
02 0280                                                  ;-----------------------------------------------------------------------
02 0280                                                  ------
02 0281                                                  ;NORMALIZE DOUBLE PRECISION NUMBER IN AC,Q WHOSE EXPONENT
02 0282                                                  ; IS IN IR-ADR.  LEAVE EXPONENT IN AR, NORMALIZED NUMBER IN AC,Q.
02 0283                                                  ; IF NUMBER IS 0, LEAVE 0 IN AR,AC, AND Q.
02 0284                                                  ;-----------------------------------------------------------------------
02 0284                                                  ------
02 0285
02 0286                                                  .DEFINE CNORM[] [D[AR] DEST[AC] PUSHJ[NRMLIZ] NORM]
02 0287
02 0288 11702 010031202000060554161624254214160000      ;A. Return immediately if 0.
02 0289                                                  NRMLIZ: ALU[QORAC] DEST[AR] COND[OBUS=0] POPJ C550 $
02 0290 11703 010331370000000001416162225431456000      ;B. Check for mantissa overflow, move exponent to AR.
02 0291                                                         ALU[AC] DEST[AR] PUSHJ[NORMOV] NORM $
02 0292 11704 010731000000060554045624257614160000                     ;Check mantissa and shift right if necessary
02 0293                                                         D[IR] MASK[18.] COND[-OBUS=0] POPJ C550 $
02 0294                                                                 ;If we adjusted right, then we definitely don't have to
02 0295                                                                 ;shift left.  We're done here.
02 0296 11705 010331004000000001416162025411456000      ;C. Left shift until normalized.
02 0297                                                         ALU[AC] COND[OBUS<0] JUMP[NEGNOR] CYLEN[C450] $
02 0298 11706 010541000000060546403624255614160000                     ;Separate into two cases, positive and negative
02 0299                                                  POSNOR: D[CONST 1] ROT[26.] ALU[D&AC] COND[-OBUS=0] POPJ C550 $
02 0300 11707 010721370000060554161623654374160000                     ;If high order bit of mantissa is one, we're done
02 0301                                                         D[AR] ALU[D-1] DEST[AR] SHORT $
02 0302 11710 010436170000236154165620254314560000                     ;Decrement the exponent
02 0303                                                         ALU[SH-AC] ENDCONN[LSH] DEST[DG] JUMP[POSNOR] NORM $
02 0304 11711 010541002000000006403620255614560000                     ;Shift left and try again.
02 0305                                                  NEGNOR: D[CONST 1] ROT[26.] ALU[D&AC] COND[OBUS=0] JUMP[NN1] C550 $
02 0306 11712 010721370000060554161623654374160000                     ;If high order bit of mantissa is zero, we're done
02 0307                                                         D[AR] ALU[D-1] DEST[AR] SHORT $
02 0308 11713 010436170000236234165620254314560000                     ;Decrement the exponent
02 0309                                                         ALU[SH-AC] ENDCONN[LSH] DEST[DG] JUMP[NEGNOR] NORM $
02 0310 11714 010541000000060554067624247214160000                     ;Shift left and try again.
02 0311                                                  NN1:    D[MASK 27.] ALU[D&AC] COND[-OBUS=0] POPJ C550 $
02 0311                                                                 ;Check for case of exactly -(2^n).  If it isn't, we're
02 0312 11715 010533170000060546403623655774160000      done
02 0313                                                         D[CONST 1] ROT[26.] ALU[DORAC] DEST[AC] SHORT $
02 0313                                                                 ;We went one too far, backup by simply OR'ing in the
02 0314                                                  relevant
02 0315                                                                 ;bit in the same manner as shifting would.
02 0316 11716 011701370000060554161624254314560000                     ;(*** What about Q??  TVR-Jul80)
02 0317                                                         D[AR] ALU[D+1] DEST[AR] POPJ NORM $
02 0317                                                                 ;Increment exponent to account for simulated left shift.
02 0318
02 0319                                                  ;-----------------------------------------------------------------------
02 0319                                                  ------
02 0320                                                  ;CHECK FOR MANTISSA OVERFLOW INTO EXPONENT FIELD.  CALL WITH
02 0321                                                  ; AR&AC,Q: # TO NORMALIZE, IR-ADR: ITS EXPONENT.
02 0322                                                  ; LEAVE EXPONENT IN AR.  SET IR-ADR=0 IF NO OVERFLOW.  SET IR-ADR<>0
02 0323                                                  ; IF OVERFLOW, AND NORMALIZE.
02 0324                                                  ;
02 0325                                                  ;M.O. <=> [AR0<>AR8] OR [(AR8=1) AND (AR9-35=0)]
02 0326                                                  ;-----------------------------------------------------------------------
02 0326                                                  ------
02 0327 11717 010561004000000002161622254214560000      NORMOV: D[AR] ROT[8] ALU[D=AC] COND[OBUS<0] JUMP[NOVYES] C550 $
02 0328                                                                 ;Check to make sure AR<0>=AR<8>.  If they don't, the
```

```
02 0333                                                         ;Check AR<8>.  If zero, we're OK
02 0334                                                 ;       \ /
02 0335                                                 ;Mantissa has overflowed, adjust it.  Right by one should be enough.
02 0336 11722 010434170000060554163623654374160000     NOVYES: ALU[SH-AC] ENDCONN[ASH] DEST[D4] SHORT $
02 0337                                                         ;Shift mantissa right (i think?)
02 0338 11723 611701370000060554045444257714160000             D[IR] MASK[18.] ALU[D+1] DEST[AR IR-ADR] POPJ NORM $
02 0339                                                         ;Increment mantissa and flag that we changed it by
02 0339                                                  setting
02 0340                                                         ;IR<18:35> to be non-zero.
02 0341
02 0342                                                 ;Mantissa has not overflowed.
02 0343 11724 010731370000060554045623657774160000     NOVNO:  D[IR] MASK[18.] DEST[AR] SHORT $
02 0344                                                         ;Save exponent in AR
02 0345 11725 610241170000060554161444254314160000             ALU[0] DEST[IR-ADR] POPJ NORM $
02 0346                                                         ;Zero IR<18:35> to signify success
02 0347
02 0348                                                 ;---------------------------------------------------------------------------
02 0348                                                 ------
02 0349                                                 ;INSERT EXPONENT IN AR INTO WORD IN AC.
02 0350                                                 ;SET FLOV,FLUN,OVF FLAGS AS REQUIRED.
02 0351                                                 ;PRESERVES AR, CLOBBERS Q.
02 0352                                                 ;---------------------------------------------------------------------------
02 0352                                                 ------
02 0353 11726 010730170000060546761623654314160000     INSEXP: D[AR] ROT[27.] DEST[Q] NORM $
02 0354                                                         ;Move exponent into position
02 0355 11727 010641070000060546621623647374160000     INSEX1: D[MASK 8.] ROT[27.] ALU[D&Q] DEST[Q] SHORT $
02 0356                                                         ;Flush stuff in mantissa area
02 0357 11730 010063170000060554161623654374160000             ALU[Q#AC] DEST[AC] SHORT $
02 0358                                                         ;Fill in exponent (complementing it if (AC) is negative
02 0358                                                  !)
02 0359 11731 010731020000000014161620254214560000             D[AR] COND[OBUS18] JUMP[EXPUF] C550 $
02 0360                                                         ;Check for underflow (i.e. below range of exponent
02 0360                                                  offset)
02 0361 11732 010731002000060547025624254214160000             D[AR] MASK[10.] ROT[28.] COND[OBUS=0] POPJ C550 $
02 0362                                                         ;Check for overflow (i.e. above range of exponent offset)
02 0362
02 0363 11733 010730170000060554161623646374160000             D[PC-FLAGS] DEST[Q] SHORT $
02 0364                                                         ;Setup to set floating overflow
02 0365 11734 010631170000060550022423655614174000     SETFOV: D[CONST 11] ROT[32.] ALU[DORQ] DEST[PC-FLAGS] C550 $
02 0366                                                         ;Turn on overflow and floating overflow
02 0367 11735 010730570000000047161623670314160000     SETOVX: D[AMEM-ABS APR-STATUS] ROT[28.] DEST[ALU1 Q] $
02 0368                                                         ;Get the Floating Overflow Interrupt Enable bit in 000.
02 0369 11736 010601570044060550603624255714160000             D[CONST 1] ROT[35.] ALU[ALU1 D+Q] SET-PC-FLAGS POPJ $
02 0370                                                         ;This causes an AROV trap if the enable bit is on.
02 0371 11737 010730170000060554161623646374160000     EXPUF:  D[PC-FLAGS] DEST[Q] SHORT $
02 0372                                                         ;Setup to set no divide and floating overflow
02 0373 11740 010630170000236706003620255714560000             D[CONST 1] ROT[24.] ALU[DORQ] DEST[Q] JUMP[SETFOV] NORM$
02 0374                                                         ;Turn on no divide, then do overflows
02 0375
02 0376                                                 ;---------------------------------------------------------------------------
02 0376                                                 ------
02 0377                                                 ;FLOATING ADD AC&AR TO MEM.  RESULT, NORMALIZED AND ROUNDED
02 0378                                                 ; IF REQUIRED IN AC,Q.
02 0379                                                 ;---------------------------------------------------------------------------
02 0379                                                 ------
02 0380                                                   .define BLEXPS[DES1 TIME DES2] [
02 0381                                                         D[MASK 8] ROT[27.] ALU[DORAC] DEST[DES2] OBUS<0 JUMP[. + 2] C550
02 0381                                                  $
02 0382                                                         D[MASK 8] ROT[27.] ALU[-D&AC] DEST[DES2] SHORT $
02 0383                                                         D[MEM] DEST[Q] COND[OBUS<0] JUMP[. + 2] C550 $
02 0384                                                         D[MASK 8] ROT[27.] ALU[-D&Q] DEST[DES1] JUMP[. + 2] NORM $
02 0385                                                         D[MASK 8] ROT[27.] ALU[DORQ] DEST[DES1] CYLEN[TIME] $
02 0386                                                     ];blexps
02 0387
02 0388                                                      ;A. + form of exponent of AC,MEM to AR,Q resp.
02 0389 11741 010730370000060542223623654374160000     FADSUB: D[AR] MASK[9] ROT[9] DEST[Q AR] SHORT $
02 0390 11742 010641002000000002003620255614560000     FADSU1: D[CONST 1] ROT[8] ALU[D&Q] COND[OBUS=0] JUMP[FS1] C550 $
02 0391 11743 010661370000060554023623647374160000             D[MASK 9] ALU[D#Q] DEST[AR] SHORT $
02 0392 11744 010730170000060542223623654775160000     FS1:    D[MEM] MASK[9] ROT[9] DEST[Q] SHORT $
02 0393 11745 010641002000000002003620255614560000             D[CONST 1] ROT[8] ALU[D&Q] COND[OBUS=0] JUMP[FS2] C550 $
02 0394 11746 010660170000060554023623647374160000             D[MASK 9] ALU[D#Q] DEST[Q] SHORT $
02 0395                                                      ;B. Compute difference, swap oprands if AC has larger exp.
02 0396                                                         ;   Leave + difference in Q, diff -1 in LOOP CTR.
02 0397                                                         ;   Leave larger exponent in IR-ADR.
02 0398 11747 610231170000060554161443654374160000     FS2:    ALU[Q] DEST[IR-ADR] SHORT $
02 0399 11750 011610006000000001416162025411456000             D[AR] ALU[Q-D] DEST[Q] COND[-OBUS<0] JUMP[FS3] C600 $
02 0400 11751 610731170000060554161443654374160000             D[AR] DEST[IR-ADR] SHORT $
02 0401 11752 010732170000060554161503654715160000             D[MEM] DEST[HOLD D-AC] NORM $
02 0402 11753 011220170000060554161623654314160000             ALU[0-Q] DEST[Q] NORM $
02 0403 11754 010211370000060554161623654314160000     FS3:    ALU[Q-1] DEST[AR] NORM $
02 0404 11755 010731170010060540160003654314174000             D[AR] ROT[LLOAD-ROT] LLOAD $
02 0405                                                      ;C. Blank exponents.
02m0406                                                         BLEXPS[HOLD NORM AC]
02m0406                                                  [
02m0406 11756 010533004000237406621620247214560000             D[MASK 8] ROT[27.] ALU[DORAC] DEST[AC] OBUS<0 JUMP[. + 2] C550 $
02m0406
02m0406 11757 010553170000060546621623647374160000             D[MASK 8] ROT[27.] ALU[-D&AC] DEST[AC] SHORT $
02m0406 11760 010730004000237454161620254615560000             D[MEM] DEST[Q] COND[OBUS<0] JUMP[. + 2] C550 $
02m0406 11761 010651170000237466621500247314560000             D[MASK 8] ROT[27.] ALU[-D&Q] DEST[HOLD] JUMP[. + 2] NORM $
02m0406 11762 010631170000060546621503647314160000             D[MASK 8] ROT[27.] ALU[DORQ] DEST[HOLD] CYLEN[NORM] $
02 0407                                                      ]     ;D. If difference between exponents >62., AC goes to oblivion.
02 0407
02 0408 11763 011700002000000001416162025411456000             D[AR] ALU[D+1] DEST[Q] COND[OBUS=0] JUMP[FS5] C600 $
02 0409 11764 011611004000000001417762025551456000             D[CONST 63.] ALU[Q-D] COND[OBUS<0] JUMP[FS4] C600 $
02 0410 11765 010243170000060554161623654374160000             ALU[0] DEST[AC] SHORT $
02 0411 11766 010240170000000001416162025431456000             ALU[0] DEST[Q] JUMP[FS5] NORM $
02 0412 11767 010240170000060554161623654374160000     FS4:    ALU[0] DEST[Q] SHORT $
02 0413 11770 010434070000237614163620254314560000             ALU[SH-AC] DEST[D4] ENDCONN[ASH] LOOP [.] NORM $
02 0414                                                      ;E. Add.
02 0415 11771 010501370000060554161623654775160000     FS5:    D[MEM] ALU[D+AC] DEST[AR] SHORT $
02 0416                                                      ;F. Normalize result.
02 0417 11772 010733170000060554161623657774160000             D[IR] DEST[AC] SHORT $  ;CHECK FOR UFA
02 0418 11773 010541002000000001000362025561456000             D[CONST 1] ROT[40] ALU[D&AC] COND[OBUS=0] JUMP[UFANOR] C550 $
02 0419
02 0420                                                 ;---------------------------------------------------------------------------
02 0420                                                 ------
02 0421                                                 ;ENTER HERE WITH RESULT IN AR,Q, EXP IN IR-ADR.
02 0422                                                 ;---------------------------------------------------------------------------
02 0422                                                 ------
02 0423 11774 010733170000236054161622254314560000     FPNAR:  CNORM  [D[AR] DEST[AC] PUSHJ[NRMLIZ] NORM]$
02 0424                                                 ;Call NORMLIZ, then change exponent range from above to adjusted
```

```
02 0428                                                         ;Save Q in HOLD (i.e. MEM)
02 0429                                                         ;If Q<0> is zero, don't round
02 0430 11776 0107301700000605541616236577141600       D[IR] DEST[Q] NORM $
02 0431                                                         ;Get Q so we check bit meaning rounding (??? Can't this
02 0431                                                  be
02 0432                                                         ;done with D[IR] ROT[7] COND[OBUS<0]... ???)
02 0433 11777 0106410020000000007203620255614560000    D[CONST 1] ROT[35] ALU[D&Q] COND[OBUS=0] JUMP[NORND] C550 $
02 0434                                                         ;Check opcode to see if rounding is requested.
02 0435                                                         ;If not, we're done
02 0436 12000 0107301700000605541616236547751600       D[MEM] DEST[Q] SHORT $
02 0437                                                         ;Restore Q
02 0438 12001 0106410000000000014107620247214560000    D[MASK 43] ALU[D&Q] COND[-OBUS=0] JUMP[YESRND] C550 $
02 0439                                                         ;Round if Q<1:35> is non-zero (? what does this
02 0439                                                  signify???)
02 0440 12002 0103310040002365541616202541145600       ALU[AC] COND[OBUS<0] JUMP[INSEXP] CYLEN[C450] $
02 0441                                                         ;If mantissa is negative, we don't round (???)
02 0442                                                 ;       \ /
02 0443 12003 6107311700000605541614436543716000       YESRND: D[AR] DEST[IR-ADR] SHORT $
02 0444                                                         ;Setup IR for NRMLIZ
02 0445 12004 0113031700002360541616222543145600       ALU[AC+1] DEST[AC] PUSHJ[NRMLIZ] NORM $
02 0446                                                         ;Increment high order word and normalize once more (???)
02 0446
02 0447 12005 0107301700002365667616202543145600       JINSEXP  [D[AR] ROT[27.] DEST[Q] JUMP[INSEX1] NORM ]$
02 0448                                                         ;Now stick in exponent
02 0449                                                 ;       ---
02 0450 12006 0107301700002365541616202547155600       NORND:  D[MEM] DEST[Q] JUMP[INSEXP] NORM $
02 0451                                                         ;Restore Q and insert exponent(?)
02 0452
02 0453                                                 ;UFA NORMALIZATION -- ONLY ON MANTISSA OVERFLOW.
02 0454 12007 0106330020000605541616242541141600       UFANOR: D[AR] ALU[DORQ] DEST[AC] COND[OBUS=0] POPJ C600 $
02 0455 12010 0107331700002363741616222543145600       D[AR] DEST[AC] PUSHJ[NORMOV] NORM $
02 0456 12011 0107301700002365667616202543145600       JINSEXP  [D[AR] ROT[27.] DEST[Q] JUMP[INSEX1] NORM ]$
02 0457                                                 ;       ---
02 0458
02 0459                                                 ;-----------------------------------------------------------------------
02 0459                                                 ------
02 0460                                                 ;SETUP LOW ORDER FP RESULT, STORE BOTH IN AC,AC+1
02 0461                                                 ;ENTER WITH LOW WORD IN MEM, HIGH WORD IN AC, HIGH
02 0462                                                 ;WORD'S EXPONENT IN AR&AC+1.
02 0463                                                 ;-----------------------------------------------------------------------
02 0463                                                 ------
02 0464 12012 0115133700000605540676235557741600       FPLOW:  D[CONST 27.] ACSEL[AC+1] ALU[AC-D] DEST[AC AR] SHORT $
02 0465 12013 0105410000000000020036201556145600       D[CONST 1] ROT[8] ACSEL[AC+1] ALU[D&AC] COND[-OBUS=0] JUMP[ZLOW]
02 0465                                                  C550 $
02 0466 12014 0107300020000000066676202546155600       D[MEM] ROT[27.] MASK[27.] DEST[Q] COND[OBUS=0] JUMP[ZLOW] C550 $
02 0466
02 0467 12015 0107331700000605467616235543716000       D[AR] ROT[27.] ACSEL[AC+1] DEST[AC] SHORT $
02 0468 12016 0105431700000605466216235473716000       D[MASK 8] ROT[27.] ALU[D&AC] ACSEL[AC+1] DEST[AC] SHORT $
02 0469 12017 5400331700000605541616231542016000       ACSEL[AC+1] ALU[QORAC] DEST[AC] NEOI $
02 0470 12020 5402431700000605541616231542016000       ZLOW:   ACSEL[AC+1] ALU[0] DEST[AC] NEOI $
02 0471
02 0472                                                 ;-----------------------------------------------------------------------
02 0472                                                 ------
02 0473                                                 ;
02 0474                                                 ;       FLOATING MULTIPLY.  AC&AR BY MEM.
02 0475                                                 ;
02 0476                                                 ;*** This code produces non-zero results if MEM=-1.0 and AC=0  !!!
02 0476                                                  TVR-Jun80
02 0477                                                 ;-----------------------------------------------------------------------
02 0477                                                 ------
02 0478                                                 ;A. SUM OF + FORM EXPONENTS -128.+1 TO IR-ADR.  +1 BECAUSE Z8
02 0479                                                 ; STEPS ARE USED TO PROVIDE ONE GUARD DIGIT.
02 0480 12021 0107303700000605422236236543141600       FM:     D[AR] MASK[9] ROT[9] DEST[Q AR] NORM$
02 0481                                                         ;Extract the exponent from AC (which was copied into AR)
02 0481
02 0482 12022 0106410020000000020036202556145600       D[CONST 1] ROT[8] ALU[D&Q] COND[OBUS=0] JUMP[FM1] C550 $
02 0483                                                         ;Check sign bit of number.  Nothing special if positive
02 0484 12023 0106613700000605540236236473716000       D[MASK 9] ALU[D#Q] DEST[AR] SHORT $
02 0485                                                         ;Sign is negative, we want the one-complement of the
02 0485                                                  exponent
02 0486                                                         ;field to get the actual exponent.
02 0487 12024 0107301700000605422236236547751600       FM1:    D[MEM] MASK[9] ROT[9] DEST[Q] SHORT $
02 0488                                                         ;Extract the exponent from MEM
02 0489 12025 0106410020000000020036202556145600       D[CONST 1] ROT[8] ALU[D&Q] COND[OBUS=0] JUMP[FM2] C550 $
02 0490                                                         ;Check sign bit of number.  Nothing special if positive
02 0491 12026 0106601700000605540236236473716000       D[MASK 9] ALU[D#Q] DEST[Q] SHORT $
02 0492                                                         ;Sign is negative, we want the one-complement of the
02 0492                                                  exponent
02 0493                                                         ;field to get the actual exponent.
02 0494 12027 0116001700000605541616236543716000       FM2:    D[AR] ALU[D+Q] CARRY DEST[Q] SHORT $
02 0495                                                         ;Add the two exponents
02 0496                                                         ;+1. FOR GUARD DIGIT.
02 0497 12030 6116111700000605416034436557741600       D[CONST 1] ROT[7] ALU[Q-D] DEST[IR-ADR] SHORT $
02 0498                                                         ;Account for exponent bias.  Put exponent in a safe
02 0498                                                  place.
02 0499                                                 ;B. BLANK EXPONENTS.
02m0500                                                         BLEXPS[Q SHORT AR]
02m0500                                                 [
02m0500 12031 0105312040002406666216202472145600       D[MASK 8] ROT[27.] ALU[DORAC] DEST[AR] OBUS<0 JUMP[. + 2] C550 $
02m0500
02m0500 12032 0105513700000605466216236473716000       D[MASK 8] ROT[27.] ALU[-D&AC] DEST[AR] SHORT $
02m0500 12033 0107300040002407341616202546155600       D[MEM] DEST[Q] COND[OBUS<0] JUMP[. + 2] C550 $
02m0500 12034 0106501700002407466216202473145600       D[MASK 8] ROT[27.] ALU[-D&Q] DEST[Q] JUMP[. + 2] NORM $
02m0500 12035 0106301700000605466216236473716000       D[MASK 8] ROT[27.] ALU[DORQ] DEST[Q] CYLEN[SHORT] $
02 0501                                                    ]    ;Extend the sign to blank out the exponent field.  We
02 0501                                                  will
02 0502                                                         ;now have perfectly good integers here, of 27 bit
02 0502                                                  magnitude
02 0503                                                         ;(assuming that the numbers were normalized to begin
02 0503                                                  with).
02 0504 12036 0102431700000000014161622254314560000    ALU[0] DEST[AC] PUSHJ[TESMUL] NORM $ ;28. STEPS.
02 0505                                                 ;;;     ALU[Q] DEST[AR] SHORT $ ;FLUSH SIGN IN LO WD.
02 0506 12037 0106503700000605540176236473716000       D[MASK 7] ALU[-D&Q] DEST[Q AR] SHORT $  ;FLUSH SIGN IN LO WD.
02 0507                                                         ;Flush remenants of multiplier in low order part of
02 0508                                                         ;word.  MASK field determined empirically (35-28???)
02 0509 12040 0106001700000605541616236543716000       D[AR] ALU[D+Q] DEST[Q] SHORT $
02 0510                                                         ;Make low order word unsigned.
02 0511 12041 0103313700002377141616202543145600       ALU[AC] DEST[AR] JUMP[FPNAR] NORM $
02 0512
02 0513                                                 ;28. STEP INTEGER MULTIPLY FOR USE BY FM.
02 0514                                                 TESMUL:         ;LOOP 28 TIMES
```

```
02 0518                                                 ;SHORT STYLE FLOATING DIVIDE AC BY MEM.
02 0519                                                 ; XOR OF DIVISOR&DIVIDEND SIGNS IN AR BIT 0.
02 0520                                                 ;---------------------------------------------------------------------
02 0520                                                 ------
02 0521 12043 010561370000060554161623654775160000     FDS:    D[MEM] ALU[D#AC] DEST[AR] SHORT $
02 0522 12044 610731170000060540203443654374160000     FDS1:   D[AR] MASK[1] ROT[1] DEST[IR-ADR] SHORT $
02 0523 12045 010331206000241174161620254214560000             ALU[AC] DEST[AR] COND[-OBUS<0] JUMP[. + 2] C550 $
02 0524 12046 011323370000060554161623654374160000             ALU[0-AC] DEST[AC AR] SHORT $
02 0525 12047 010241170000060554161143654314160000             ALU[0] DEST[D%AMEM6] NORM $
02 0526 12050 010241170000060554161363654314160000             ALU[0] DEST[D%AMEM17] $
02 0527                                                          ;Flag to do a 28-bit divide.
02 0528
02 0529                                                 ;---------------------------------------------------------------------
02 0529                                                 ------
02 0530                                                 ;       Floating Divide
02 0531                                                 ;
02 0532                                                 ; DIVIDEND HIGH ORDER WORD IN AR, LOW IN A-MEM[6]. DIVISOR IN MEM.
02 0533                                                 ; BOTH IN POSITIVE FORM.  IR.35=XOR(DIVIDEND SIGN,DIVISOR SIGN).
02 0534                                                 ;---------------------------------------------------------------------
02 0534                                                 ------
02 0535                                                 FD:
02 0536                                                 ;A. PUT DIVISOR IN + FORM, GET DIVISOR EXPONENT.
02 0537                                                 ;.1 Shuffle IR.35 into A-MEM[4].
02 0538 12051 010731170000060554045103654771416000             D[IR] MASK[18.] DEST[D%AMEM4] NORM $
02 0539                                                          ;Save IR-ADR in A-MEM
02 0540 12052 010730006000241314161620254615560000             D[MEM] DEST[Q] COND[-OBUS<0] JUMP[. + 2] C550 $
02 0541                                                          ;Check for negative divisor
02 0542 12053 011710170000060554161503654715160000             D[MEM] ALU[0-D] DEST[Q HOLD] NORM $
02 0543                                                          ;Negate divisor
02 0544 12054 610731170000060542222144365471516000             D[MEM] ROT[9] MASK[8] DEST[IR-ADR] NORM $
02 0545                                                          ;Extract exponent from divisor, store in IR<18:35>
02 0546 12055 010651170000060546621503647314160000             D[MASK 8] ROT[27.] ALU[-D&Q] DEST[HOLD] NORM $
02 0547                                                          ;Extract mantissa from divisor
02 0548                                                 ;B. Fetch and blank HOW Dividend exponent.
02 0549 12056 010730170000060542221623654314160000             D[AR] ROT[9] MASK[8] DEST[Q] NORM $
02 0550                                                          ;Extract dividend exponent.
02 0551 12057 010733170000060554067623654374160000             D[AR] MASK[27.] DEST[AC] SHORT $
02 0552                                                          ;Extract mantissa from dividend
02 0553                                                 ;C. Compute resultant exponent.
02 0554 12060 011610170000060554045623657774160000             D[IR] MASK[18.] ALU[Q-D] DEST[Q] SHORT $
02 0555                                                          ;Subtract divisor exponent from dividend exponent
02 0556 12061 610801170000060541603443655714160000             D[CONST 1] ROT[7] ALU[D+Q] DEST[IR-ADR] NORM $
02 0557                                                          ;Include exponent offset
02 0558                                                 ;D. Move LOW Dividend to Q.
02 0559 12062 010730170000060554161623663374160000             D[D%AMEM6] DEST[Q] SHORT $
02 0560                                                 ;E. If divisor<=dividend,  shift dividend right, increment the
02 0561                                                          ; resultant exponent.  save diff. in A-MEM[7] for long mode.
02 0562 12063 011511006000000014161160254515560000             D[MEM] ALU[AC-D] DEST[D%AMEM7] COND[-OBUS<0] JUMP[FDAD] C600 $
02 0563                                                 ;F. Save exp in A-MEM[5]
02 0564 12064 010731170000060554161123657714160000     FDSHFT: D[IR] DEST[D%AMEM5] NORM $
02 0565 12065 010731170000114514161622254314560000             PUSHJ[FDODIV] NORM $
02 0566                                                          ;Do the divide. Result DOES NOT have signs adjusted.
02 0567                                                 ;G. Save remainder in A-MEM[6], put quotient in AC, AR; put 0 in Q.
02 0568 12066 010331170000060554161143654314160000             ALU[AC] DEST[D%AMEM6] NORM $
02 0569 12067 010233370000060554161623654374160000             ALU[Q] DEST[AC AR] SHORT $
02 0570 12070 010240170000060554161623654374160000             ALU[0] DEST[Q] SHORT $
02 0571                                                 ;H. Prepare to round 28. bit result.
02 0572 12071 010434170000060554163623654374160000             ALU[SH-AC] ENDCONN[ASH] DEST[D4] SHORT $
02 0573                                                          ;Shift AC,Q rt 1 bit
02 0574 12072 610721170000237714161442262714560000             D[D%AMEM5] ALU[D-1] DEST[IR-ADR] NORM PUSHJ[FPNAR] $
02 0575                                                          ;Move exp for FPNAR, normalize and insert exponent into
02 0575                                                  quotient.
02 0576 12073 010731002000060554003624262214160000             D[D%AMEM4] MASK[1] C550 OBUS=0 POPJ $
02 0577                                                          ;Done if positive result.
02 0578 12074 011323170000060554161624254314160000             ALU[0-AC] DEST[AC] NORM POPJ $
02 0579                                                          ;Negate quotient.
02 0580                                                 ;---------------------------------------------------------------------
02 0580                                                 ------
02 0581                                                 ;HERE WHEN DIVIDEND IS >= DIVISOR.  SHIFT DIVIDEND RIGHT
02 0582                                                 ; AND INCREMENT RESULTANT AC.  THIS WILL ALLOW DIVIDE TO
02 0583                                                 ; SUCCEED IF BOTH DIVISOR & DIVIDEND WERE NORMALIZED
02 0584                                                 ; AND DIVISOR <> 0.
02 0585                                                 ;---------------------------------------------------------------------
02 0585                                                 ------
02 0586 12075 010434170000060554165623654374160000     FDAD:   ALU[SH-AC] ENDCONN[LSH] DEST[D4] SHORT $
02 0587 12076 011511006000236774161620254515560000             D[MEM] ALU[AC-D] ACSEL[AC] COND[-OBUS<0] JUMP[EXPUF] C600 $
02 0588                                                          ;If divisor is still too big, give up and set no divide,
02 0588                                                  etc.
02 0589 12077 611701170000241514045440254771456000             D[IR] MASK[18.] ALU[D+1] DEST[IR-ADR] JUMP[FDSHFT] NORM $
02 0590
```

```
03 0591                                                 ;
03 0592                                                 ;Strays from KI instructions
03 0593                                                 ;
03 0594
03 0595                                                 .REPEAT WAITS [
03 0596
03 0597                                                   .opcode[247]  ;(KAFIX at SAIL: "A. Kotok should have done this")
03 0598                                                         ACSEL[AC] ALU[AC] DEST[HOLD AR] COND[-OBUS<0] LBJUMP[KAFIX1] C550
03 0598                                                  $
03 0599                                                         ACSEL[AC] ALU[AC] DEST[HOLD AR] COND[-OBUS<0] LBJUMP[KAFIX1] C550
03 0599                                                  $
03 0600                                                                 ;Load up things for FIXER, check for positive mantissa
03 0601                                                   .reloc
03 0602
03 0603                                                 ;KAFIX (Opcode 247) continued
03 0604                                                 ;Positive number to fix
03 0605                                                         .even
03 0606                                                 KAFIX1: ACSEL[AC] ALU[0-AC] DEST[AR] JUMP[KAFIXN] NORM $
03 0607                                                                 ;Negate AR so that its exponent can be used.
03 0608                                                 KAFIXP: D[IR] ROT[27.] MASK[9.] DEST[Q] COND[-OBUS=0] PUSHJ[FIXER] $
03 0609                                                                 ;Extract exponent adjustment factor and do fixing
03 0610                                                         ACSEL[AC] ALU[Q] DEST[AC] JUMP[MAIN] NORM $
03 0611                                                                 ;Put result into AC and we're done
03 0612                                                 ;**** Here's another JUMP[MAIN] that could be faster
03 0613
03 0614                                                 ;Negative number to fix
03 0615                                                 KAFIXN: D[IR] ROT[27.] MASK[9.] DEST[Q] COND[-OBUS=0] PUSHJ[FIXER] $
03 0616                                                                 ;Extract exponent adjustment factor and do fixing
03 0617                                                 ;;;     ACSEL[AC] D[MASK R] ALU[D/#Q] DEST[AC MA]       ;*** Assembler
03 0617                                                  loses!
03 0618                                                         ACSEL[AC] D[MASKR] ALU[D/#Q] DEST[AC] JUMP[MAIN] $      ;Sigh...
03 0618
03 0619                                                                 ;Put sign extended result into AC and we're done
03 0620                                                 ;**** Here's another JUMP[MAIN] that could be faster
03m0621                                                 ];[
03m0621
03m0621                                                   .opcode[247]   [xlist
03m0621                                                    list  ];(KAFIX at SAIL: "A. Kotok should have done this")
03m0621 06516 010331206000000001416150125421456000             ACSEL[AC] ALU[AC] DEST[HOLD AR] COND[-OBUS<0] LBJUMP[KAFIX1] C550
03m0621                                                  $
03m0621 06517 010331206000000001416150125421456000             ACSEL[AC] ALU[AC] DEST[HOLD AR] COND[-OBUS<0] LBJUMP[KAFIX1] C550
03m0621                                                  $
03m0621                                                                 ;Load up things for FIXER, check for positive mantissa
03m0621                                                   .reloc
03m0621                                                  [.USE[HILOC]
03m0621                                                  [ xlist
03m0622                                                    list ]                                       ]
03m0622                                                 ;KAFIX (Opcode 247) continued
03m0622                                                 ;Positive number to fix
03m0622                                                         .even
03m0622                                                  [:. \ Z + .
03m0623 12100 011321370000000001416162025431456000               ]KAFIX1:      ACSEL[AC] ALU[0-AC] DEST[AR] JUMP[KAFIXN] NORM $
03m0623                                                                 ;Negate AR so that its exponent can be used.
03m0623 12101 010730000000000006623622Z25771456000             KAFIXP: D[IR] ROT[27.] MASK[9.] DEST[Q] COND[-OBUS=0] PUSHJ[FIXER] $
03m0623                                                                 ;Extract exponent adjustment factor and do fixing
03m0623 12102 010233170000227114161620254314S6000                      ACSEL[AC] ALU[Q] DEST[AC] JUMP[MAIN] NORM $
03m0623                                                                 ;Put result into AC and we're done
03m0623                                                 ;**** Here's another JUMP[MAIN] that could be $aster
03m0623
03m0623                                                 ;Negative number to fix
03m0623 12103 010730000000000006623622Z25771456000             KAFIXN: D[IR] ROT[27.] MASK[9.] DEST[Q] COND[-OBUS=0] PUSHJ[FIXER] $
03m0623                                                                 ;Extract exponent adjustment factor and do fixing
03m0623                                                 ;;;     ACSEL[AC] D[MASK R] ALU[D/#Q] DEST[AC MA]       ;*** Assembler
03m0623                                                  loses!
03m0623 12104 010673170000227114121620247314S6000                      ACSEL[AC] D[MASKR] ALU[D/#Q] DEST[AC] JUMP[MAIN] $      ;Sigh...
03m0623
03m0623                                                                 ;Put sign extended result into AC and we're done
03m0623                                                 ;**** Here's another JUMP[MAIN] that could be faster
03 0621                                                 ]REPEAT WAITS
03 0622
03m0623
03m0623                                                   .opcode[126]
03 0624 06254 010730170000000000601162025571256000      [xlist
03 0625 06255 010730170000000000601162025571456000        list  ]FIXR: DFRQ D[CONST 04] ROT[24.] DEST[Q] JUMP[FIXR1] $
03 0626                                                         D[CONST 04] ROT[24.] DEST[Q] JUMP[FIXR0] NORM $
03m0627                                                                 ;Start making constant 0.5
03m0627                                                   .reloc
03m0627                                                  [.USE[HILOC]
03 0628                                                  [ xlist
03 0629 12105 010331170000000001416150005431456000               list ]                                       ]
03 0630 12106 010731004000000001416162025461556000             FIXR0:  ALU[MEMAC] DEST[HOLD] JUMP[FIXR1] NORM $
03 0631                                                         FIXR1:  D[MEM] COND[OBUS<0] JUMP[FIXRN] C550 $
03 0632 12107 010603370000237027441622Z25571456000                     ;Round different directions
03 0633                                                         D[CONST 20] ROT[30.] ALU[D+Q] DEST[AC AR] PUSHJ[FADSUB] NORM $
03 0634 12110 010730170000060554067623655774160O0                      ;Positive. Finish making constant 0.5 and add it
03 0635                                                         D[CONST 33] DEST[Q] SHORT $
03 0636 12111 010331370000000001416150025431456000                     ;Start making magic constant
03 0637                                                         ALU[AC] DEST[AR HOLD] JUMP[FIXRZ] NORM $
03 0638                                                                 ;Setup for FIXER.  Rest is same as KIFIX
03 0639 12112 011623370000237027541622Z25571456000             ;       ---
03 0640                                                         FIXRN:  D[CONST 60] ROT[30.] ALU[D-Q] DEST[AC AR] PUSHJ[FADSUB] NORM $
03 0641 12113 010730170000060554067623655774160O0                      ;Negative. Finish making constant -0.5 and add it
03 0642                                                         D[CONST 33] DEST[Q] SHORT $
03 0643 12114 010630170000060541405623655774160O0                      ;Start making magic constant
03 0644                                                         D[CONST 2] ROT[6] ALU[DORQ] DEST[Q] SHORT $
03 0645 12115 011321370000000001416150225431456000                     ;Finish making magic constant
03 0646                                                         ALU[0-AC] DEST[AR HOLD] PUSHJ[FIXER] NORM $
03 0647 12116 011223170000227114161620254314S6000                      ;Setup for FIXER
03 0648                                                         ACSEL[AC] ALU[0-Q] DEST[AC] JUMP[MAIN] NORM $
03 0649                                                                 ;Negate it back again and start next instruction
03 0650                                                 ;**** Here's another JUMP[MAIN] that could be faster
03 0651                                                 ;       ---
03m0652
03m0652                                                   .opcode[122]
03 0653 06244 010730170000000001406762025571256000      [xlist
03 0654 06245 010730170000000001406762025571456000        list  ]KIFIX:        DFRQ D[CONST 33] DEST[Q] JUMP[KIFIX1] NORM $
03 0655                                                         D[CONST 33] DEST[Q] JUMP[KIFIX0] NORM $
03m0656                                                                 ;Start making magic constant and start fetch
03m0656                                                   .reloc
03m0656                                                  [.USE[HILOC]
03m0656                                                  [ xlist
```

```
03 0662 12121 010630170000000000140562225571456000     FIXRZ:  D[CONST 2] ROT[6] ALU[DORQ] DEST[Q] PUSHJ[FIXER] NORM $
03 0663                                                        ;Finish making magic constant and do fix.
03 0664 12122 010233170000227114161620254314560000            ACSEL[AC] ALU[Q] DEST[AC] JUMP[MAIN] NORM $
03 0665                                                        ;Put result into AC and we're done
03 0666                                                ;**** Here's another JUMP[MAIN] that could be faster
03 0667                                                ;       ---
03 0668 12123 010630170000060541405623655774160000    KIFIXN: D[CONST 2] ROT[6] ALU[DORQ] DEST[Q] SHORT $
03 0669                                                        ;Finish making magic constant
03 0670 12124 011711370000000001416150225471556000            D[MEM] ALU[0-D] DEST[HOLD AR] PUSHJ[FIXER] NORM $
03 0671                                                        ;ABS to both MEM and AR.  Fix the number
03 0672 12125 011223170000227114161620254314560000            ACSEL[AC] ALU[0-Q] DEST[AC] JUMP[MAIN] NORM $
03 0673                                                        ;Negate it back again and start next instruction
03 0674                                                ;**** Here's another JUMP[MAIN] that could be faster
03 0675
03 0676                                                ;Fix a number (without sign extension) in MEM, AR=ABS(MEM), Q=magic
03 0676                                                 number
03 0677                                                ;(233 for normal fix)
03 0678                                                FIXER:  D[AR] ROT[9] MASK[9] ALU[D-Q] DEST[ROTR Q]
03 0679 12126 011620004000000000ZZZZ44025421457000                     COND[OBUS<0] JUMP[FIXER2] C550 $
03 0680                                                        ;Calculate number of positions to move
03 0681                                                        ;Jump if we'll be shifting right (n<Z+26)
03 0682 12127 010601170000060554066503655714170000            D[CONST 27.] ALU[D+Q] DEST[MASKR] NORM $
03 0683                                                        ;Construct appropriate mask
03 0684 12130 010731170000060554067503654715160000            D[MEM] MASK[27.] DEST[HOLD] NORM $
03 0685                                                        ;Flush exponent, our mask won't reach.
03 0686 12131 011611004000000014023620255614560000            D[CONST 9.] ALU[Q-D] COND[OBUS<0] JUMP[FIXER1] C550 $
03 0687                                                        ;Check for overflow
03 0688                                                ;;; Set overflow here.  What kind?  (We can live without it for KAFIX,
03 0688                                                 since it
03 0689                                                ;;; never did check, but when KIFIX is done, this will have to be
03 0689                                                 corrected.)
03 0690                                                ;;;     D[PC-FLAGS] DEST[Q] NORM $
03 0691                                                ;;;     D[CONST 1] ROT[35.] ALU[DORQ] DEST[PC-FLAGS] NORM $
03 0692                                                ;;;             ;Set overflow flag.
03 0693 12132 011701170041406554107623647314160000            D[MASK 35.] ALU[D+1] SET-PC-FLAGS $
03 0694                                                        ;This causes an AROV trap if the enable bit is on.
03 0695 12133 010731170000060554110503655714170000            D[CONST 44] DEST[MASKR] NORM $
03 0696                                                        ;For those losers who want to see some of the number...
03 0697                                                ;Now that we know how much to shift things, do it and return.
03 0698 12134 010730170000060552121624254715160000    FIXER1: D[MEM] ROT[R] MASK[R] DEST[Q] NORM POPJ $
03 0699                                                        ;Gee, that was fast.
03 0700                                                ;Negative, shift count, the easy case.
03 0701 12135 010601004000000014066500255714570000    FIXER2: D[CONST 27.] ALU[D+Q] DEST[MASKR] COND[OBUS<0] JUMP[FIXER3] $
03 0702                                                        ;Construct appropriate mask.  If !x!<1, return zero
03 0703 12136 010601170000242714110440255714570000            D[CONST 36.] ALU[D+Q] DEST[ROTR] JUMP[FIXER1] NORM $
03 0704                                                        ;Hardware doesn't believe in negative shift counts
03 0705                                                ;Number is fractional, i.e. shift would go off the end.  Return zero
03 0706 12137 010240170000060554160504254374170000    FIXER3: ALU[0] DEST[Q MASKR] SHORT POPJ $
03 0707                                                        ;Just return zero.  Set mask just in case
03 0708
03m0709                                                  .opcode[1Z7]   [xlist
03 0709                                                   list  ];FLTR
03 0710 06256 010730170000000001406762025571256000    FLTR:   DFRQ D[CONST 33] DEST[Q] JUMP[FLTR1] $
03 0711                                                        ;Setup magic constant
03 0712 06257 010331370000000001416162005431456000            ALU[MEMAC] DEST[AR] JUMP[FLTR0] $
03 0713                                                        ;Setup for normalize
03m0714                                                  .reloc
03m0714                                                 [.USE[HILOC]                                    . #
03m0714                                                 [ xlist
03 0715                                                   list ]                                       ]
03 0716 12140 010730170000000001406762025571456000    FLTR0:  D[CONST 33] DEST[Q] JUMP[FLTR2] $
03 0717 12141 010731370000000001416162025471556000    FLTR1:  D[MEM] DEST[AR] JUMP[FLTR2] $
03 0718 12142 610631170000060541405413655714160000    FLTR2:  D[CONST 2] ROT[6] ALU[DORQ] DEST[IR-ADR] NORM $
03 0719                                                        ;Make magic constant for exponent
03 0720                                                ;;;     ALU[0] DEST[Q] PUSHJ[FPNARL] $
03 0721 12143 010240170000237714161622254314560000            ALU[0] DEST[Q] PUSHJ[FPNAR] $
03 0722                                                        ;Clear low order word
03 0723                                                        ;Normalize and round (happens to have same bit on as
03 0723                                                 FxxR)
03 0724                                                        ;Result goes to AC
03 0725 12144 010731170000227114161620254314560000            JUMP[MAIN] NORM $
03 0726                                                        ;Start next instruction fetch
03 0727                                                ;**** Another JUMP[MAIN]
03 0728
03 0729                                                .REPEAT WAITS [
03 0730                                                ;;; Checksum user memory.  It is used as follows (note: PROG=DDB+1):
03 0731                                                ;;;
03 0732                                                ;;;     MOVN DDB,DDB    ;-SIZE OF CORE IMAGE,,0  = AOBJN POINTER TO CORE
03 0732
03 0733                                                ;;;     MOVEI PROG,0
03 0734                                                ;;;     XCTR XBLTR,[CHKINS DDB,]        ;CHECKSUM!
03 0735                                                ;;;
03 0736                                                        .even
03 0737                                                ;CAUTION: Do not add/delete instructions from here to CKINS1!!!
03 0738                                                CHKINS: ALU[AC] DEST[MA] COND[-OBUS<0] JUMP[MAIN] $
03 0739                                                        ;If WCMA exhausted, instruction done
03 0740                                                        DFRQ COND[-MA<20] JUMP[CKINSZ] $
03 0741                                                        ;Fetch first word.  We make sure we get in at least one
03 0741                                                 word
03 0742                                                        ;before taking interrupts
03 0743                                                        ;Jump if not referencing real ACs
03 0744                                                .repeat 1 - waits [
03 0745                                                .error $         ;WARNING: CHKINS doesn't know about ACBAS!!!
03 0746                                                ];.repeat 1 - waits
03 0747                                                        COND[USER] JUMP[CKINSZ] NORM $
03 0748                                                        ;If we're referencing Exec memory, take the AC after
03 0748                                                 all!!!
03 0749                                                ;       \ /
03 0750                                                CKINS1: ACSEL[MA] ALU[AC] DEST[HOLD] JUMP[CKINS2] NORM $
03 0751                                                        ;Copy specified AC
03 0752                                                        D[MEM] ACSEL[AC+1] ALU[D+AC] DEST[AC] NORM $
03 0753                                                        ;Add into checksum
03 0754                                                        D[1,,1] ALU[D+AC] DEST[AC MA] COND[-OBUS<0] JUMP[MAIN] C550 $
03 0755                                                        ;Update WMCA.  Done if exhausted.
03 0756                                                        COND[INTRPT] JUMP[CKINS1] C500 $
03 0757                                                        ;See if there's an interrupt pending
03 0758                                                        COND[MA<20] JUMP[CKINS1] C500 $
03 0759                                                        ;Check to see if we stay in AC checking loop
03 0760                                                ;       ---
03 0761                                                CKINS2: D[MEM] ACSEL[AC+1] ALU[D+AC] DEST[AC] NORM $
```

```
03 0767                                                         ;Wait for next word.  Jump if not interrupting
03 0768                                                 D[PC] ALU[D-1] DEST[PC] JUMP[MAIN] NORM $
03 0769                                                         ;Somebody wants to take a micro-interrupt.
03 0770                                                         ;Backup PC so we can finish later, and do instruction
03 0770                                                  fetch
03 0771                                                         ;ritual just in case we don't interrupt
03m0772                                                 ];[
03m0772                                                 ;;; Checksum user memory.  It is used as follows (note: PROG=DDB+1):
03m0772                                                 ;;;
03m0772                                                 ;;;     MOVN DDB,DDB    ;-SIZE OF CORE IMAGE,,0  = AOBJN POINTER TO CORE
03m0772
03m0772                                                 ;;;     MOVEI PROG,0
03m0772                                                 ;;;     XCTR XBLTR,[CHKINS DDB,]        ;CHECKSUM!
03m0772                                                 ;;;
03m0772                                                         .even
03m0772                                                  [:. \ 2 + .
03m0773                                                    ];CAUTION: Do not add/delete instructions from here to CKINS1!!!
03m0773 12146 710331006000ZZ711416162025431456000       CHKINS: ALU[AC] DEST[MA] COND[-OBUS<0] JUMP[MAIN] $
03m0773                                                         ;If WCMA exhausted, instruction done
03m0773 12147 010731112000000041416Z025431Z56000                DFRQ COND[-MA<20] JUMP[CKINSZ] $
03m0773                                                         ;Fetch first word.  We make sure we get in at least one
03m0773                                                  word
03m0773                                                         ;before taking interrupts
03m0773                                                         ;Jump if not referencing real ACs
03m0773                                                 .repeat 1 - waits [
03m0773                                                 .error $         ;WARNING: CHKINS doesn't know about ACBAS!!!
03m0773                                                 ];.repeat 1 - waits
03m0773 12150 010731IZ400000000141616Z025431456000              COND[USER] JUMP[CKINSZ] NORM $
03m0773                                                         ;If we're referencing Exec memory, take the AC after
03m0773                                                  all!!!
03m0773                                                 ;       \ /
03m0773 12151 010331170000000001416150005431456000      CKINS1: ACSEL[MA] ALU[AC] DEST[HOLD] JUMP[CKINSZ] NORM $
03m0773                                                         ;Copy specified AC
03m0773 12152 010503170000060554161623554715160000             D[MEM] ACSEL[AC+1] ALU[D+AC] DEST[AC] NORM $
03m0773                                                         ;Add into checksum
03m0773 12153 710503006037ZZ711400362025561456000               D[1,,1] ALU[D+AC] DEST[AC MA] COND[-OBUS<0] JUMP[MAIN] C550 $
03m0773                                                         ;Update WMCA.  Done if exhausted.
03m0773 12154 010731104000243Z341616Z025421456000               COND[INTRPT] JUMP[CKINS1] C500 $
03m0773                                                         ;See if there's an interrupt pending
03m0773 12155 010731110000243Z341616Z025421456000               COND[MA<Z0] JUMP[CKINS1] C500 $
03m0773                                                         ;Check to see if we stay in AC checking loop
03m0773                                                 ;       ---
03m0773 12156 010503170000060554161623554715160000     CKINS2: D[MEM] ACSEL[AC+1] ALU[D+AC] DEST[AC] NORM $
03m0773                                                         ;Add into checksum
03m0773 12157 710503006037ZZ711400362025561456000               D[1,,1] ALU[D+AC] DEST[AC MA] COND[-OBUS<0] JUMP[MAIN] C550 $
03m0773                                                         ;Update WMCA.  Done if exhausted.
03m0773                                                 ;       \ /
03m0773 12160 010731106000243354161620254212Z56000     CKINS3: DFRQ COND[-INTRPT] JUMP[CKINSZ] C500 $
03m0773                                                         ;Wait for next word.  Jump if not interrupting
03m0773 12161 610721170000ZZ711416146025631456000               D[PC] ALU[D-1] DEST[PC] JUMP[MAIN] NORM $
03m0773                                                         ;Somebody wants to take a micro-interrupt.
03m0773                                                         ;Backup PC so we can finish later, and do instruction
03m0773                                                  fetch
03m0773                                                         ;ritual just in case we don't interrupt
03 0772                                                 ].REPEAT WAITS
```

```
01 0005                                                 ;Floating Point stuff
01 0006
01 0007                                                 ;These definitions allow selected microcode to assemble for either an F2
01 0007                                                  or
01 0008                                                 ;an F4, depending on F4SW.   See VC.SLO for an example.
01 0009
01 0010                                                 .DEFINE DEFINE-A-MEM[SYM VAL] [
01 0011                                                         SYM = VAL + D%AMEM0
01 0012                                                 ];.DEFINE-A-MEM
01 0013
01 0014                                                 .DEFINE MEMST [] [ SMAC ]
01 0015                                                 .DEFINE FIXM1 [] [ DFRQ ALU[MEMAC] DEST[HOLD] ]
01 0016                                                 .DEFINE FETCH-NEXT-INST [] [ JUMP[MAIN] ]
01 0017                                                 .DEFINE FETCH-NEXT-INST1 [] [ JUMP[MAIN] ]
01 0018
01 0019                                                 .REPEAT 1 + (MUA-ROT > 0) [
01 0020                                                 .DEFINE XDISP [] [ ODISP LONG ]
01 0021                                                 .DEFINE XDISP1 [] [ ODISP LONG ]
01m0022                                                 ];[
01m0022                                                 .DEFINE XDISP [] [ ODISP LONG ]
01m0022                                                 .DEFINE XDISP1 [] [ ODISP LONG ]
01 0022                                                 ].REPEAT 1 + (MUA-ROT > 0)
01 0023
01 0024                                                 .REPEAT 0 - (MUA-ROT > 0) [
01 0025                                                 .DEFINE XDISP [] [ DEST[AR] $ D[AR] ROT[MUA-ROT] ODISP LONG ]
01 0026                                                 .DEFINE XDISP1 [] [ ROT[MUA-ROT] ODISP LONG ]
01 0027                                                 ];.REPEAT 0 - (MUA-ROT > 0)
01 0028
01 0029       4047      ILLINS = ITRAP
01 0030
01 0031                                                 .insert F4IO
```

```
01 0001                                         ;<XF4-MICROCODE>F4IO.SLO.5, 22-Jan-83 13:14:33, Edit by FRENCH
01 0002                                         ;BE SURE MAP OFF FOR RESETING OF TYMNET PTRS
01 0003
01 0004
01 0005
01 0006                                         .REPEAT 1 - STANSW [
01 0007                                         .define ILLIOT[FIRST LAST] [
01 0008                                            .opcode[FIRST]
01 0009                                            .repeat (1 + LAST - FIRST) [
01 0010                                                 JUMP[ITRAP] $
01 0011                                                 JUMP[ITRAP] $
01 0012                                                ]
01 0013                                           ]
01 0014                                         ];.REPEAT 1 - STANSW
01 0015
01 0016                                         .REPEAT STANSW [
01 0017                                         .define ILLIOT[FIRST LAST] [
01 0018                                            .opcode[FIRST]
01 0019                                            .repeat (1 + LAST - FIRST) [
01 0020                                                 COND[-USER] LBJUMP[PANIOT] $
01 0021                                                 COND[-USER] LBJUMP[PANIOT] $
01 0022                                                ]
01 0023                                           ]
01m0024                                         ];[
01m0024                                         .define ILLIOT[FIRST LAST] [
01m0024                                            .opcode[FIRST]
01m0024                                            .repeat (1 + LAST - FIRST) [
01m0024                                                 COND[-USER] LBJUMP[PANIOT] $
01m0024                                                 COND[-USER] LBJUMP[PANIOT] $
01m0024                                                ]
01m0024                                           ]
01 0024                                         ].REPEAT STANSW
01 0025
01 0026
01m0027                                                 ILLIOT[701 701]
01m0027                                          [
01m0027                                            .opcode[701]
01m0027                                          [xlist
01m0028                                            list  ]   .repeat (1 + 701 - 701) [
01m0028                                                 COND[-USER] LBJUMP[PANIOT] $
01m0028                                                 COND[-USER] LBJUMP[PANIOT] $
01m0028                                                ]
01m0028                                          [
01m0028 07602 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0028 07603 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01 0028                                                 ] ]
01m0029                                         .repeat 1 - JSYSDF [ ILLIOT[703 703] ]  [ ILLIOT[703 703]  [
01m0029                                            .opcode[703]
01m0029                                          [xlist
01m0030                                            list  ]   .repeat (1 + 703 - 703) [
01m0030                                                 COND[-USER] LBJUMP[PANIOT] $
01m0030                                                 COND[-USER] LBJUMP[PANIOT] $
01m0030                                                ]
01m0030                                          [
01m0030 07606 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0030 07607 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01 0029                                                 ] ]];JSYS DEVICE IOT
01 0030
01m0031                                                 ILLIOT[704 711]
01m0031                                          [
01m0031                                            .opcode[704]
01m0031                                          [xlist
01m0032                                            list  ]   .repeat (1 + 711 - 704) [
01m0032                                                 COND[-USER] LBJUMP[PANIOT] $
01m0032                                                 COND[-USER] LBJUMP[PANIOT] $
01m0032                                                ]
01m0032                                          [
01m0032 07610 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0032 07611 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0032                                                ][
01m0032 07612 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0032 07613 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0032                                                ][
01m0032 07614 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0032 07615 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0032                                                ][
01m0032 07616 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0032 07617 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0032                                                ][
01m0032 07620 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0032 07621 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0032                                                ][
01m0032 07622 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0032 07623 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01 0032                                                 ] ]      ILLIOT[713 714]  [
01m0033                                            .opcode[713]
01m0033                                          [xlist
01 0034                                            list  ]   .repeat (1 + 714 - 713) [
01 0035                                                 COND[-USER] LBJUMP[PANIOT] $
01 0036                                                 COND[-USER] LBJUMP[PANIOT] $
01m0037                                                ]
01m0037                                          [
01m0037 07626 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0037 07627 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0037                                                ][
01m0037 07630 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0037 07631 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01 0032                                                 ] ]
01 0033                                          .repeat 1 - DLS [
01 0034                                             .opcode[737]
01 0035                                               JUMP[MAIN] $
01 0036                                               JUMP[MAIN] $
01 0037                                                    ];[
01m0038                                             .opcode[737]
01m0038                                          [xlist
01 0039 07676 010731170000227114161620254314560000             list  ]      JUMP[MAIN] $
01 0040 07677 010731170000227114161620254314560000                  JUMP[MAIN] $
01 0037                                                                ].repeat 1 - DLS
01 0038                                         .REPEAT 1 - WAITS [
01 0039                                                 ILLIOT[750 752]
```

```
01 0045                                                 ];.REPEAT 1 - WAITS
01 0046
01 0047                                                 .REPEAT WAITS [
01 0048                                                         ILLIOT[733 736]         ;French's tape IOTs
01 0049                                                 .REPEAT 1 - DLS [
01 0050                                                 ;       ILLIOT[737 737]         ;DLS
01 0051                                                 ];.REPEAT 1 - DLS
01 0052                                                 ;               [740 743]       ;Disk
01 0053                                                 ;       ILLIOT[744 747]         ;Tymnet
01 0054                                                         ILLIOT[750 751]         ;Diagnostic instructions?
01 0055                                                         ILLIOT[752 753]         ;SAMx (devices 520 to 534) at CCRMA
01 0056                                                 .REPEAT VC [            ;Versatec
01 0057                                                         .OPCODE[754]    ;Undefined
01 0058                                                         D[CONST VCDEV] DEST[DEV-ADR] NORM COND[-USER] LBJUMP[VCIOT] $
01 0059                                                         D[CONST VCDEV] DEST[DEV-ADR] NORM COND[-USER] LBJUMP[VCIOT] $
01 0060                                                                 ;Set micro device address for Versatec and check for
01 0061                                                                 ;IOT-User mode
01 0062                                                 ];.REPEAT VC
01 0063                                                 .REPEAT 1 - VC [
01 0064                                                         ILLIOT[754 754]         ;VC
01 0065                                                 ];.REPEAT 1 - VC
01 0066                                                         ILLIOT[755 755]         ;IMP (device 550)
01 0067                                                         ILLIOT[756 757]         ;unused
01 0068                                                 ;               [760 760]       ;TIMER
01 0069                                                         ILLIOT[761 767]         ;unused (formerly extravangent timer
01 0069                                                  iots)
01 0070                                                 ;Note: CCRMA uses DCA = 770
01 0071                                                 .REPEAT 1 - STANSW [
01 0072                                                         ILLIOT[771 771]         ;FOOVision (opcodes 766 thru 773)
01 0073                                                 ];.REPEAT 1 - STANSW
01 0074                                                 .REPEAT STANSW [
01 0075                                                       .opcode[771]
01 0076                                                         D[CONST AS-UDEV] DEST[DEV-ADR] NORM COND[-USER] LBJUMP[ASIOT] $
01 0077                                                         D[CONST AS-UDEV] DEST[DEV-ADR] NORM COND[-USER] LBJUMP[ASIOT] $
01 0078                                                                 ;Set micro device address for audio switch and check for
01 0078
01 0079                                                                 ;IOT-User mode
01 0080                                                 ];.REPEAT STANSW
01 0081                                                 .REPEAT 1 + (DIAGDEV > 0) [
01 0082                                                         ILLIOT[772 773]         ;Diagnostic instruction (for arbitrary
01 0082                                                  device)
01 0083                                                 ];.REPEAT 1 + (DIAGDEV > 0)
01 0084                                                 .REPEAT 1 - STANSW [
01 0085                                                         ILLIOT[774 774]         ;TMPGRN
01 0086                                                 ];.REPEAT 1 - STANSW
01 0087                                                 ;               [775 776]       ;FOONET
01 0088                                                         ILLIOT[777 777]         ;unused
01m0089                                                 ];[
01m0089                                                         ILLIOT[733 736]  [
01m0089                                                   .opcode[733]
01m0089                                                 [xlist
01m0090                                                   list  ]   .repeat (1 + 736 - 733) [
01m0090                                                         COND[-USER] LBJUMP[PANIOT] $
01m0090                                                         COND[-USER] LBJUMP[PANIOT] $
01m0090                                                       ]
01m0090                                                 [
01m0090 07666 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0090 07667 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0090                                                       ][
01m0090 07670 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0090 07671 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0090                                                       ][
01m0090 07672 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0090 07673 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0090                                                       ][
01m0090 07674 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0090 07675 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01 0089                                                       ] ]       ;French's tape IOTs
01 0090                                                 .REPEAT 1 - DLS [
01 0091                                                 ;       ILLIOT[737 737]         ;DLS
01 0092                                                 ];[
01 0093                                                 ;       ILLIOT[737 737]         ;DLS
01 0092                                                 ].REPEAT 1 - DLS
01 0093                                                 ;               [740 743]       ;Disk
01 0094                                                 ;       ILLIOT[744 747]         ;Tymnet
01m0095                                                         ILLIOT[750 751]  [
01m0095                                                   .opcode[750]
01m0095                                                 [xlist
01m0096                                                   list  ]   .repeat (1 + 751 - 750) [
01m0096                                                         COND[-USER] LBJUMP[PANIOT] $
01m0096                                                         COND[-USER] LBJUMP[PANIOT] $
01m0096                                                       ]
01m0096                                                 [
01m0096 07720 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0096 07721 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0096                                                       ][
01m0096 07722 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0096 07723 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01 0095                                                       ] ]       ;Diagnostic instructions?
01 0096                                                         ILLIOT[752 753]  [
01m0097                                                   .opcode[752]
01m0097                                                 [xlist
01 0098                                                   list  ]   .repeat (1 + 753 - 752) [
01 0099                                                         COND[-USER] LBJUMP[PANIOT] $
01 0100                                                         COND[-USER] LBJUMP[PANIOT] $
01m0101                                                       ]
01m0101                                                 [
01m0101 07724 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0101 07725 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0101                                                       ][
01m0101 07726 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01m0101 07727 010731126000000001416162125431456000             COND[-USER] LBJUMP[PANIOT] $
01 0096                                                       ] ]       ;SAMx (devices 520 to 534) at CCRMA
01 0097                                                 .REPEAT VC [            ;Versatec
01 0098                                                         .OPCODE[754]    ;Undefined
01 0099                                                         D[CONST VCDEV] DEST[DEV-ADR] NORM COND[-USER] LBJUMP[VCIOT] $
01 0100                                                         D[CONST VCDEV] DEST[DEV-ADR] NORM COND[-USER] LBJUMP[VCIOT] $
01 0101                                                                 ;Set micro device address for Versatec and check for
01 0102                                                                 ;IOT-User mode
01 0103                                                 ];[             ;Versatec
```

```
01 0108                                                                 ;IOT-User mode
01 0103                                                         ].REPEAT VC
01 0104                                                         .REPEAT 1 - VC [
01 0105                                                                 ILLIOT[754 754]         ;VC
01 0106                                                         ];.REPEAT 1 - VC
01m0107                                                                 ILLIOT[755 755]  [
01m0107                                                           .opcode[755]
01m0107                                                         [xlist
01m0108                                                           list  ]   .repeat (1 + 755 - 755) [
01m0108                                                                 COND[-USER] LBJUMP[PANIOT] $
01m0108                                                                 COND[-USER] LBJUMP[PANIOT] $
01m0108                                                               ]
01m0108                                                         [
01m0108 07732 0107311Z60000000141616Z1Z5431456000                       COND[-USER] LBJUMP[PANIOT] $
01m0108 07733 0107311Z60000000141616Z1Z5431456000                       COND[-USER] LBJUMP[PANIOT] $
01 0107                                                               ] ]       ;IMP (device 550)
01 0108                                                                 ILLIOT[756 757]  [
01m0109                                                           .opcode[756]
01m0109                                                         [xlist
01m0109                                                           list  ]   .repeat (1 + 757 - 756) [
01 0110                                                                 COND[-USER] LBJUMP[PANIOT] $
01 0111                                                                 COND[-USER] LBJUMP[PANIOT] $
01 0112                                                               ]
01m0113                                                         [
01m0113                                                                 COND[-USER] LBJUMP[PANIOT] $
01m0113 07734 0107311Z60000000141616Z1Z5431456000                       COND[-USER] LBJUMP[PANIOT] $
01m0113 07735 0107311Z60000000141616Z1Z5431456000                     ][
01m0113                                                                 COND[-USER] LBJUMP[PANIOT] $
01m0113 07736 0107311Z60000000141616Z1Z5431456000                       COND[-USER] LBJUMP[PANIOT] $
01m0113 07737 0107311Z60000000141616Z1Z5431456000                     ] ]       ;unused
01 0108                                                         ;               [760 760]         ;TIMER
01 0109                                                                 ILLIOT[761 767]  [
01 0110                                                           .opcode[761]
01m0111                                                         [xlist
01m0111                                                           list  ]   .repeat (1 + 767 - 761) [
01 0112                                                                 COND[-USER] LBJUMP[PANIOT] $
01 0113                                                                 COND[-USER] LBJUMP[PANIOT] $
01 0114                                                               ]
01m0115                                                         [
01m0115                                                                 COND[-USER] LBJUMP[PANIOT] $
01m0115 07742 0107311Z60000000141616Z1Z5431456000                       COND[-USER] LBJUMP[PANIOT] $
01m0115 07743 0107311Z60000000141616Z1Z5431456000                     ][
01m0115                                                                 COND[-USER] LBJUMP[PANIOT] $
01m0115 07744 0107311Z60000000141616Z1Z5431456000                       COND[-USER] LBJUMP[PANIOT] $
01m0115 07745 0107311Z60000000141616Z1Z5431456000                     ][
01m0115                                                                 COND[-USER] LBJUMP[PANIOT] $
01m0115 07746 0107311Z60000000141616Z1Z5431456000                       COND[-USER] LBJUMP[PANIOT] $
01m0115 07747 0107311Z60000000141616Z1Z5431456000                     ][
01m0115                                                                 COND[-USER] LBJUMP[PANIOT] $
01m0115 07750 0107311Z60000000141616Z1Z5431456000                       COND[-USER] LBJUMP[PANIOT] $
01m0115 07751 0107311Z60000000141616Z1Z5431456000                     ][
01m0115                                                                 COND[-USER] LBJUMP[PANIOT] $
01m0115 07752 0107311Z60000000141616Z1Z5431456000                       COND[-USER] LBJUMP[PANIOT] $
01m0115 07753 0107311Z60000000141616Z1Z5431456000                     ][
01m0115                                                                 COND[-USER] LBJUMP[PANIOT] $
01m0115 07754 0107311Z60000000141616Z1Z5431456000                       COND[-USER] LBJUMP[PANIOT] $
01m0115 07755 0107311Z60000000141616Z1Z5431456000                     ][
01m0115                                                                 COND[-USER] LBJUMP[PANIOT] $
01m0115 07756 0107311Z60000000141616Z1Z5431456000                       COND[-USER] LBJUMP[PANIOT] $
01m0115 07757 0107311Z60000000141616Z1Z5431456000                     ] ]       ;unused (formerly extravangant timer iots)
01 0110                                                         ;Note: CORMA uses DCA = 770
01 0111                                                         .REPEAT 1 - STANSW [
01 0112                                                                 ILLIOT[771 771]         ;FOOVision (opcodes 766 thru 773)
01 0113                                                         ];.REPEAT 1 - STANSW
01 0114                                                         .REPEAT STANSW [
01 0115                                                               .opcode[771]
01 0116                                                                 D[CONST AS-UDEV] DEST[DEV-ADR] NORM COND[-USER] LBJUMP[ASIOT] $
01 0117                                                                 D[CONST AS-UDEV] DEST[DEV-ADR] NORM COND[-USER] LBJUMP[ASIOT] $
01 0118                                                                         ;Set micro device address for audio switch and check for
01 0119
01 0119
01 0120                                                                         ;IOT-User mode
01 0121                                                         ];[
01m0122                                                                 .opcode[771]
01m0122                                                         [xlist
01 0123 07762 0107311Z60400000140604612Z571457000                         list  ]      D[CONST AS-UDEV] DEST[DEV-ADR] NORM COND[-USER]
01 0123                                                         LBJUMP[ASIOT] $
01 0124 07763 0107311Z60400000140604612Z571457000                       D[CONST AS-UDEV] DEST[DEV-ADR] NORM COND[-USER] LBJUMP[ASIOT] $
01 0125                                                                 ;Set micro device address for audio switch and check for
01 0125
01 0126                                                                 ;IOT-User mode
01 0121                                                         ].REPEAT STANSW
01 0122                                                         .REPEAT 1 + (DIAGDEV > 0) [
01 0123                                                                 ILLIOT[772 773]         ;Diagnostic instruction (for arbitrary
01 0123                                                          device)
01 0124                                                         ];.REPEAT 1 + (DIAGDEV > 0)
01 0125                                                         .REPEAT 1 - STANSW [
01 0126                                                                 ILLIOT[774 774]         ;TMPGRN
01 0127                                                         ];.REPEAT 1 - STANSW
01 0128                                                         ;               [775 776]         ;FOONET
01m0129                                                                 ILLIOT[777 777]  [
01m0129                                                           .opcode[777]
01m0129                                                         [xlist
01m0130                                                           list  ]   .repeat (1 + 777 - 777) [
01m0130                                                                 COND[-USER] LBJUMP[PANIOT] $
01m0130                                                                 COND[-USER] LBJUMP[PANIOT] $
01m0130                                                               ]
01m0130                                                         [
01m0130 07776 0107311Z60000000141616Z1Z5431456000                       COND[-USER] LBJUMP[PANIOT] $
01m0130 07777 0107311Z60000000141616Z1Z5431456000                       COND[-USER] LBJUMP[PANIOT] $
01 0129                                                               ] ]       ;unused
01 0089                                                         ].REPEAT WAITS
01 0090
01m0091                                                           .reloc
01m0091                                                         [.USE[HILOC]
01m0091                                                         [ xlist
01 0092                                                           list ]                                        ]
01 0093                                                         .repeat VID [
01 0094                                                             .insert F4VID
01 0095                                                               ;Get the FOOVISION code.
01 0096                                                           ]
```

```
01 0102
01 0103                                                 .repeat DCA [
01 0104                                                     .insert DCA
01 0105                                                        ;Alternate scanner code (F2 or F4)
01m0106                                                   ]
01m0106                                                 [
01m0106                                                     .insert DCA
```

```
01m0001                                         ;------------------------------------------------------------------------
01m0001                                         ------
01m0001                                         ;
01m0001                                         ;       DCA - DCA simulator for TTY scanner
01m0001                                         ;
01m0001                                         ;------------------------------------------------------------------------
01m0001                                         ------
01m0001                                         ;Caveats:
01m0001                                         ;       Not fully implemented for more than one TTY card.  You will have
                                                 to fix
01m0001                                         ;       code near DCACOZ to do IACK to more than one card, and define
                                                 NDLS to
01m0001                                         ;       be number of TTY cards used.  These are the known problems;
                                                 others may
01m0001                                         ;       exist, especially for newer version of TTY card.
01m0001                                         ;
01m0001                                         ;       A select CONO with bit 28 off will generate an illegal
                                                 instruction
01m0001                                         ;       error.  Also, a DATAI following a non-select CONO that had IACK
                                                 off
01m0001                                         ;       will generate an illegal instruction error as will a DATAO when
                                                 IACK
01m0001                                         ;       has not been cleared by a non-select CONO or either a DATAI or
                                                 DATAO
01m0001                                         ;       when CS is not set.
01m0001
01m0001                                         ;Format for write
01m0001                                         ;       Bit(s)  Meaning
01m0001                                         ;       9:14    TTY number  (9:10 compared with GRP NO switches)
01m0001                                         ;       15      CS
01m0001                                         ;       16      WE
01m0001                                         ;       28:35   Data
01m0001
01m0001                                         ;Format for CSR
01m0001                                         ;       Bit(s)  Meaning
01m0001                                         ;        8      Interrupt Enable
01m0001                                         ;       11      MR
01m0001
01m0001                                         ;A-Mem usage
01m0001                                         DEFINE-A-MEM[DCADSP 0]   [
01m0001         40              DCADSP = 0 + D%AMEM0
01m0001                                         ]       ;IOT dispatch,,interrupt dispatch
01m0001                                         DEFINE-A-MEM[DCASTATE 1]         [
01m0001         41              DCASTATE = 1 + D%AMEM0
01m0001                                         ];Microcode status (most of what CONI returns)
01m0001                                         DEFINE-A-MEM[DCACPR 2]   [
01m0001         42              DCACPR = 2 + D%AMEM0
01m0001                                         ]       ;Card, Port, Register
01m0001                                         DEFINE-A-MEM[DCASEL 3]   [
01m0001         43              DCASEL = 3 + D%AMEM0
01m0001                                         ]       ;Hardware select word
01m0001                                         DEFINE-A-MEM[DCAJMP 4]   [
01m0001         44              DCAJMP = 4 + D%AMEM0
01m0001                                         ]       ;Saved dispatch address for register select
01m0001
01m0001                                         ;MAPF fields
01m0001         0 DLSSEL = 0        ;Select TTY and register number
01m0001         4 DLSIACK = 4       ;Interrupt acknowledge (also, interrupting line)
01m0001         10          DLSCSR = 10     ;Interface Command/Status Register
01m0001         14          DLSNOSEL = 14   ;Don't select anything
01m0001                                         ;Load write data in above format if MAPF not eq DLSCSR
01m0001                                         ;A write effects every card.  A CSR write effects every card.
01m0001                                         ;A MAPF select of IACK effects only the selected card.
01m0001                                         ;A read effects only the selected card.
01m0001
01m0001                                         ;Number of microseconds to wait for IACK
01m0001         10          IACKWT = 10     ;.2*16 = 3.2 + margin
01m0001
01m0001         3 NDLS = 3          ;Number of DLS cards in LLL FZ
01m0001
01m0001                                         .REPEAT F4SW [
01m0001                                                 .opcode[770]
01m0001                                                 D[CONST DCA-UDEV] DEST[DEV-ADR] COND[-USER] LBJUMP[DCAIOT] $
01m0001                                                 D[CONST DCA-UDEV] DEST[DEV-ADR] COND[-USER] LBJUMP[DCAIOT] $
01m0001                                                 .reloc
01m0001                                         ];[
01m0001                                                 .opcode[770]
01m0001                                         [xlist
01m0002 07760 0107311Z60400000140404612557145700        list ]      D[CONST DCA-UDEV] DEST[DEV-ADR] COND[-USER]
01m0002                                         LBJUMP[DCAIOT] $
01m0002 07761 0107311Z60400000140404612557145700        D[CONST DCA-UDEV] DEST[DEV-ADR] COND[-USER] LBJUMP[DCAIOT] $
01m0002                                                 .reloc
01m0002                                         [.USE[HILOC]
01m0002                                         [ xlist
01 0001                                           list ]                                ]].REPEAT F4SW
01 0002
01m0003                                                 .PAIR
01m0003                                         [:. \ 2 + .
01 0004 1Z162 0107310020001021016036Z024611456000         ]DCAIOT:     D[PC-FLAGS] ROT[6 + 1] MASK[1] COND[OBUS=0] JUMP[MUUO]
01 0004                                         LONG $
01 0005                                                 ;Trap if User and not IOT-USER
01 0006 1Z163 0107301700000605434116Z365771416000       D[IR] ROT[12. + 1 + 1] MASK[4] DEST[Q] NORM $
01 0007                                                 ;Extract IOT decode * 2.  Note we can do this because the
01 0008
01 0009                                                 ;machine has already done indexing/indirection and bits
01 0010 1Z164 0106011700000000004440007Z6011456000      ;13:17 are guaranteed zero
01 0011                                         D[DCADSP] ROT[18.] MASK[16.] ALU[D+Q] XDISP  [ QDISP LONG ]$
01 0012                                                 ;Dispatch of type of IOT
01 0013
01 0014                                         ;Allocate space for dispatch table
01 0015                                         DCA-DISPATCH:                                   .#
01 0016                                         :. + 20
01 0017
01 0017                                         ;------------------------------------------------------------------------
01 0018                                         ------
01 0019                                         ;
01 0020                                         ;       CONI K10,
01 0021                                         ;        ------------- ------------          -------- -------------
01 0022                                         ;       !   !   !   !   !   !   !   !   !   !   !            !
```

```
01 0028                                                 ;-----------------------------------------------------------------------------
01 0028                                                 ----
01m0029                                                         .ORG[DCA-DISPATCH + 12]
01m0029                                                 [ xlist
01 0030 12177 0107313100002Z2665416156126070456000       list ]DCACNI: D[DCASTATE] DEST[MEMSTO] MEMST  [ SMAC  [ IFRQ
01 0030                                                 DEST[MEMSTO AR] COND[MA<20] LBJUMP[SEOI] ]]$
01 0031                                                                 ;Simply read back saved state
01 0032 12200 0107311700000605541616236543141600                       NOP $
01 0033 12201 0107301700000605541615236607141600       DCACSZ: D[DCASTATE] DEST[Q] NORM $
01 0034 12202 0107311700000000141616202543145600                       JUMP[XXCONSZ] NORM $
01 0035                                                                         ;Go do generalized CONSZ.
01 0036 12203 0107301700000605541616236607141600       DCACSO: D[DCASTATE] DEST[Q] NORM $
01 0037 12204 0107311700000000141616202543145600                       JUMP[XXCONSO] NORM $
01 0038                                                                         ;Go do generalized CONSO.
01 0039
01m0040                                                         .RELOC
01m0040                                                 [.USE[HILOC]
01m0040                                                 [ xlist
01 0041                                                   list ]                                ]
01 0042                                                 ;-----------------------------------------------------------------------------
01 0042                                                 ------
01 0043                                                 ;
01 0044                                                 ;       CONO K10,
01 0045                                                 ;
01 0046                                                 ;        ---------- ---------- ---------- ----------
01 0047                                                 ;       !    !          !    !    !    !    !    !          !
01 0048                                                 ;       ! 0  !    Unused  !DATA!DATA!TEST! EN !IACK!  PI CHANNEL  !
01 0049                                                 ;       !    !          ! 28 ! 29 !    !    !    !   !    !    !
01 0050                                                 ;        ---------- ---------- ---------- ----------
01 0051                                                 ;       !    !          !    !                   !          !
01 0052                                                 ;       !  1 !  CARD FILE # !  1 !      PORT #       !    REG #     !
01 0053                                                 ;       !    !          !    !                   !    !   !    !
01 0054                                                 ;        ---------- ---------- ---------- ----------
01 0055                                                 ;         24   25   26   27   28   29   30   31   32   33        35
01 0056                                                 ;
01 0057                                                 ;-----------------------------------------------------------------------------
01 0057                                                 ----
01m0058                                                         .ORG[DCA-DISPATCH + 10]
01m0058                                                 [ xlist
01 0059 12175 0107310041000000006161620257614560000     list ]DCACNO: D[IR] ROT[24.] COND[OBUS<0] JUMP[DCASRG] CS50 $
01 0060                                                                 ;Jump if setup CONO
01 0061 12176 0107301700000000140256202507145600               D[DCASTATE] MASK[36. - 26.] DEST[Q] JUMP[DCACO1] NORM $
01 0062                                                                 ;Get old state of device, excluding CS and IACK RTN
01m0063                                                         .RELOC
01m0063                                                 [.USE[HILOC]
01m0063                                                 [ xlist
01 0064 12205 0106501700000605540216236473141600         list ]                                ]DCACO1: D[MASK (36. - 28.)]
01 0064                                                 ALU[-D&Q] DEST[Q] NORM $
01 0065                                                                 ;Remove old state (keep only RPLY and PIR)
01 0066 12206 0106301700000605540210236577141600               D[IR] MASK[36. - 28.] ALU[DORQ] DEST[Q DCASTATE] NORM $
01 0067                                                                 ;Put in relevant bits in Q to construct the status
01 0068                                                                 ;word to be stored back into A-Mem.
01 0069 12207 0107310060000000007761622257614560000             D[IR] ROT[31.] COND[-OBUS<0] PUSHJ[DCARST] CS50 $
01 0070                                                                 ;If enable bit is off, gronk state of scanner
01 0071 12210 0107310060000000010161620257614560000             D[IR] ROT[32.] COND[-OBUS<0] JUMP[DCACO3] CS50 $
01 0072                                                                 ;If not doing IACK, then skip this business
01 0073 12211 0107311700310605462015236557141600               D[CONST 0] ROT[35. - 10.] DEST[IOD] SPEC[IOB-OUT] NORM $
01 0074                                                                 ;Select card 0 but don't run the ASTRA chips
01 0075 12212 0107311714300605541616236542141600               MAPF[DLSNOSEL] SPEC[IOB-IN] CS00 $
01 0076                                                                 ;Read - INTR bit from card 0
01 0077 12213 0107310060000000043616202571145600               CYLEN[IOB-IN] D[IOD] ROT[17.] COND[-OBUS<0] JUMP[DCACO5] $
01 0078                                                                 ;Jump if card 0 has INTR on
01 0079 12214 0107311700310605462035236557141600               D[CONST 1] ROT[35. - 10.] DEST[IOD] SPEC[IOB-OUT] NORM $
01 0080                                                                 ;Select card 1 but don't run the ASTRA chips
01 0081 12215 0107311714300605541616236542141600               MAPF[DLSNOSEL] SPEC[IOB-IN] CS00 $
01 0082                                                                 ;Read - INTR bit from card 1
01 0083 12216 0107310060000000043616202571145600               CYLEN[IOB-IN] D[IOD] ROT[17.] COND[-OBUS<0] JUMP[DCACO6] $
01 0084                                                                 ;Jump if card 1 has INTR on
01 0085 12217 0107311700310605462055236557141600               D[CONST 2] ROT[35. - 10.] DEST[IOD] SPEC[IOB-OUT] NORM $
01 0086                                                                 ;Select card 2 but don't run the ASTRA chips
01 0087 12220 0107311714300605541616236542141600               MAPF[DLSNOSEL] SPEC[IOB-IN] CS00 $
01 0088                                                                 ;Read - INTR bit from card 2
01 0089 12221 0107310060000000043616202571145600               CYLEN[IOB-IN] D[IOD] ROT[17.] COND[-OBUS<0] JUMP[DCACO7] $
01 0090                                                                 ;Jump if card 2 has INTR on
01 0091
01 0092                                                 ;No one responded.  Set IACK RTN, clear RPLY, set NO ADDR
01 0093 12222 0106301700000605424030236557141600       DCACO2: D[CONST 1] ROT[35. - 25.] ALU[DORQ] DEST[Q DCASTATE] NORM $
01 0094                                                                 ;Turn on IACK RTN
01 0095 12223 0107311700000000026030402557145600               D[CONST 1] ROT[35. - 24.] DEST[DCACPR] JUMP[DCACO3] NORM $
01 0096                                                                 ;Set NO ADDR
01 0097
01 0098 12224 0117011700310605450035236557141600       DCACO5: D[CONST 1] ROT[35. - 15.] ALU[D+1] DEST[IOD] SPEC[IOB-OUT] NORM $
01 0098
01 0099                                                                 ;Assert CS and set card file for card 0
01 0100                                                                 ;with invalid address (low bit) to make sure no one
01 0100                                                 responds
01 0101 12225 0107311714100605540320036555141700               MAPF[DLSNOSEL] CYLEN[IOB-OUT] D[CONST ((2 * IACKWT) - 3)] LLOAD $
01 0101
01 0102                                                                 ;Set max. time for ASTRA to respond
01 0103                                                                 ;*** I think DLSNOSEL allows setup time for IACK on
01 0104                                                                 ;*** ASTRA's low order data bus bit.
01 0105 12226 0102410704002445541615202542145600               MAPF[DLSIACK] ALU[0] DEST[IOD] LOOP[.] CYLEN[CS00] $
01 0106                                                                 ;Wait for slowest.  Load up IOD with CS off for use
01 0106                                                 later.
01 0107 12227 0107301704300605540216236606141600               MAPF[DLSIACK] D[DCASTATE] MASK[36. - 28.] DEST[Q] SPEC[IOB-IN]
01 0107                                                 CS00 $
01 0108                                                                 ;Start read and restore state word,
01 0109                                                                 ;clearing CS, IACK RTN, RPLY and PIR
01 0110 12230 0107313704310605541616236571141600               MAPF[DLSIACK] CYLEN[IOB-IN] D[IOD] DEST[AR] SPEC[IOB-OUT] $
01 0111                                                                 ;Finish read. Start clearing CS
01 0112                                                         MAPF[DLSNOSEL] CYLEN[IOB-OUT]           ,#
01 0113 12231 0107311714300605541616236541141600       .REPEAT F4SW [ $ ]      [ $ ];Bug:  Can't do A-MEM store and IOT at same
01 0113                                                 time!
01 0114 12232 0106311700000605422030236557141600               D[CONST 1] ROT[35. - 26.] ALU[DORQ] DEST[DCASTATE] $
01 0115                                                                 ;Finish clearing CS, set RPLY
01 0116 12233 0107301700000605540216236543741600               D[AR] MASK[8] DEST[Q] SHORT $
01 0117                                                                 ;Extract PORT #
01 0118 12234 0107313700000000140016202557145600               D[CONST 0] ALU[D] DEST[AR] JUMP[DCACOJ] NORM $
01 0119                                                                 ;card file 0
01 0120
```

```
01 0124 1Z236 010731171410060554032003655514170000             MAPF[DLSNOSEL] CYLEN[IOB-OUT] D[CONST ((2 * IACKWT) - 3)] LLOAD $
01 0124
01 0125                                                                 ;Set max. time for ASTRA to respond
01 0126 1Z237 010241070400244774161520254214560000             MAPF[DLSIACK] ALU[0] DEST[IOD] LOOP[.] CYLEN[CS00] $
01 0127                                                                 ;Wait for slowest.  Load up IOD with CS off for use
01 0127                                                          later.
01 0128 1Z240 010730170430060554021623660614160000             MAPF[DLSIACK] D[DCASTATE] MASK[36. - 28.] DEST[Q] SPEC[IOB-IN]
01 0128                                                          CS00 $
01 0129                                                                 ;Start read and restore state word,
01 0130                                                                 ;clearing CS, IACK RTN, RPLY and PIR
01 0131 1Z241 010731370431060554161623657114160000             MAPF[DLSIACK] CYLEN[IOB-IN] D[IOD] DEST[AR] SPEC[IOB-OUT] $
01 0132                                                                 ;Finish read. Start clearing CS
01 0133                                                         MAPF[DLSNOSEL] CYLEN[IOB-OUT]
01 0134 1Z242 010731171400060554161623654114160000            .REPEAT F4SW [ $ ]      [ $ ];Bug:  Can't do A-MEM store and IOT at same
01 0134                                                          time!
01 0135 1Z243 010631170000060542203023655714160000             D[CONST 1] ROT[35. - 26.] ALU[DORQ] DEST[DCASTATE] $
01 0136                                                                 ;Finish clearing CS, set RPLY
01 0137 1Z244 010730170000060554021623654374160000             D[AR] MASK[8] DEST[Q] SHORT $
01 0138                                                                 ;Extract PORT #
01 0139 1Z245 010731370000000014003620255714560000             D[CONST 1] ALU[D] DEST[AR] JUMP[DCACOJ] NORM $
01 0140                                                                 ;card file 1
01 0141
01 0142 1Z246 010730170000060546205623655714160000            DCACO7: D[CONST 2] ROT[35. - 10.] ALU[D] DEST[Q] NORM $
01 0143                                                                 ;Set card number to 2
01 0144 1Z247 011601170031060545003523655714160000             D[CONST 1] ROT[35. - 15.] ALU[D+Q+1] DEST[IOD] SPEC[IOB-OUT] NORM
01 0144                                                          $
01 0145                                                                 ;Assert CS and set card file for card 2
01 0146                                                                 ;with invalid address (low bit) to make sure no one
01 0146                                                          responds
01 0147 1Z250 010731171410060554032003655514170000             MAPF[DLSNOSEL] CYLEN[IOB-OUT] D[CONST ((2 * IACKWT) - 3)] LLOAD $
01 0147
01 0148                                                                 ;Set max. time for ASTRA to respond
01 0149 1Z251 010241070400245234161520254214560000             MAPF[DLSIACK] ALU[0] DEST[IOD] LOOP[.] CYLEN[CS00] $
01 0150                                                                 ;Wait for slowest.  Load up IOD with CS off for use
01 0150                                                          later.
01 0151 1Z252 010730170430060554021623660614160000             MAPF[DLSIACK] D[DCASTATE] MASK[36. - 28.] DEST[Q] SPEC[IOB-IN]
01 0151                                                          CS00 $
01 0152                                                                 ;Start read and restore state word,
01 0153                                                                 ;clearing CS, IACK RTN, RPLY and PIR
01 0154 1Z253 010731370431060554161623657114160000             MAPF[DLSIACK] CYLEN[IOB-IN] D[IOD] DEST[AR] SPEC[IOB-OUT] $
01 0155                                                                 ;Finish read. Start clearing CS
01 0156                                                         MAPF[DLSNOSEL] CYLEN[IOB-OUT]
01 0157 1Z254 010731171400060554161623654114160000            .REPEAT F4SW [ $ ]      [ $ ];Bug:  Can't do A-MEM store and IOT at same
01 0157                                                          time!
01 0158 1Z255 010631170000060542203023655714160000             D[CONST 1] ROT[35. - 26.] ALU[DORQ] DEST[DCASTATE] $
01 0159                                                                 ;Finish clearing CS, set RPLY
01 0160 1Z256 010730170000060554021623654374160000             D[AR] MASK[8] DEST[Q] SHORT $
01 0161                                                                 ;Extract PORT #
01 0162 1Z257 010731370000000014005620255714560000             D[CONST 2] ALU[D] DEST[AR] JUMP[DCACOJ] NORM $
01 0163                                                                 ;card file 2
01 0164
01 0165 1Z260 010630170000060541603623655714160000            DCACOJ: D[CONST 1] ROT[35. - 28.] ALU[DORQ] DEST[Q] NORM $
01 0166                                                                 ;Set stray bit
01 0167 1Z261 010631170000060542161043654314160000             D[AR] ROT[35. - 27.] ALU[DORQ] DEST[DCACPR] NORM $
01 0168                                                                 ;Fill in CARD FILE #
01 0169 1Z262 010730170000060554161623660714160000             D[DCASTATE] DEST[Q] NORM $
01 0170                                                                 ;Restore DCASTATE in Q, do rest of COND
01 0171 1Z263 010731006000227107761620257614560000            DCACO3: D[IR] ROT[31.] COND[-OBUS<0] FETCH-NEXT-INST  [ JUMP[MAIN] ]C550
01 0171                                                          $
01 0172                                                                 ;If EN is not on, then don't enable micro-interrupts!
01 0173 1Z264 010731004000227106761620260614560000             D[DCASTATE] ROT[27.] COND[OBUS<0] FETCH-NEXT-INST  [ JUMP[MAIN]
01 0173                                                          ]C550 $
01 0174                                                                 ;If we've already planted an interrupt request, wait
01 0174                                                          until
01 0175                                                                 ;it is acknowledged by COND K10,K10IAK (Usually COND
01 0175                                                          K10,3x)
01 0176 1Z265 010731170031060546603523655714160000             D[CONST 1] ROT[35. - 8.] DEST[IOD] SPEC[IOB-OUT] NORM $
01 0177                                                                 ;Set micro-interrupt enable.
01 0178 1Z266 010731171000227114161620254114560000             MAPF[DLSCSR] CYLEN[IOB-OUT] FETCH-NEXT-INST  [ JUMP[MAIN] ]$
01 0179                                                                 ;And we're done.
01 0180                                                        ;       ---
01 0181                                                        ;Select a chip
01 0182 1Z267 010731006000101167161620257614560000            DCASRG: D[IR] ROT[28.] COND[-OBUS<0] JUMP[ILLINS] C550 $
01 0183                                                                 ;If bit 28 is not set, then complain bitterly.
01 0184 1Z270 010730170000060554027623657714160000             D[IR] MASK[36. - 25.] DEST[Q] NORM $
01 0185                                                                 ;Remember register selection.
01 0186 1Z271 010651170000060554003043655714160000             D[CONST 1] ALU[-D&Q] DEST[DCACPR] NORM $
01 0187                                                                 ;After offending register select bit.
01 0188 1Z272 010731370000060550211623657714160000             D[IR] ROT[1 + 32.] MASK[4] DEST[AR] NORM $
01 0189                                                                 ;Extract PORT #, which here is the chip within a board.
01 0190 1Z273 010730170000060545361623654314160000             D[AR] ROT[35. - 14.] DEST[Q] NORM $
01 0191                                                                 ;Put it in format that the hardware understands.
01 0192 1Z274 010731370000060547007623657714160000             D[IR] ROT[1 + 27.] MASK[3] DEST[AR] NORM $
01 0193                                                                 ;Extract CARD FILE #, which here is the board
01 0194 1Z275 010630170000060546361063654314160000             D[AR] ROT[35. - 14. + 4.] ALU[DORQ] DEST[Q DCASEL] NORM $
01 0195                                                                 ;Construct full TTY number, in left half, where hardware
01 0195
01 0196                                                                 ;expects it.
01 0197                                                        .REPEAT OLDDLS (
01 0198                                                                 D[CONST 1] ROT[35. - 9.] ALU[DORQ] DEST[Q IOD] SPEC[IOB-OUT] NORM
01 0198                                                          $
01 0199                                                                 ;Turn on bit that sets TTY number and make sure we
01 0199                                                          satisfy
01 0200                                                                 ;setup time in the hardware
01 0201                                                                 MAPF[DLSNOSEL] CYLEN[IOB-OUT] ALU[Q] DEST[Q DCASEL] $
01 0202                                                                 ;Turn on bit that sets TTY number
01 0203                                                                 D[DCADSP] MASK[16.] DEST[Q] NORM $
01 0204                                                                 ;Offsets from interrupt dispatch are negative at start at
01 0204
01 0205                                                                 ;the first location preceding the interrupt.
01 0206                                                                 D[DCACPR] ROT[1 + 34.] MASK[2] ALU[D+1] DEST[AR] NORM $
01 0207                                                                 ;Calculate dispatch address while we have Q available
01 0208                                                                 D[AR] ROT[1] ALU[Q-D] DEST[DCAJMP] NORM $
01 0209                                                                 ;Multiply by two to get offset for address to select
01 0209                                                          registers
01 0210                                                        );.REPEAT OLDDLS
01 0211                                                        .REPEAT 1 - OLDDLS (
01 0212                                                                 D[CONST NDLS] ROT[35. - 14. + 4.] ALU[Q-D]
01 0213                                                                         COND[-OBUS<0] JUMP[DCASG2] C550 $
01 0214                                                                 ;Check to make sure the port really exists
01 0215                                                                 D[CONST 1] ROT[35. - 8.] ALU[DORQ] DEST[Q DCASEL] NORM $
```

```
01 0218                                                 ;;;             ;(i think)
01 0219                                                         D[DCADSP] MASK[16.] DEST[Q] NORM $
01 0220                                                                 ;Offsets from interrupt dispatch are negative at start at
01 0220
01 0221                                                                 ;the first location preceding the interrupt.
01 0222                                                         D[DCACPR] ROT[1 + 34.] MASK[2] ALU[D+1] DEST[AR] NORM $
01 0223                                                                 ;Calculate dispatch address while we have Q available
01 0224                                                         D[AR] ROT[1] ALU[Q-D] DEST[DCAJMP] NORM $
01 0225                                                                 ;Multiply by two to get offset for address to select
01 0225                                                  registers
01m0226                                                 ];[
01m0226                                                         D[CONST NDLS] ROT[35. - 14. + 4.] ALU[Q-D]
01m0226 12276 011611006000000000620762025561456000                             COND[-OBUS<0] JUMP[DCASG2] CS50 $
01m0226                                                                 ;Check to make sure the port really exists
01m0226                                                 ;;;     D[CONST 1] ROT[35. - 9.] ALU[DORQ] DEST[Q DCASEL] NORM $
01m0226                                                 ;;;             ;Turn on bit that sets TTY number and save select word
01m0226                                                  until
01m0226                                                 ;;;             ;DATAI/DATAO is done.  We'll have enough setup time at
01m0226                                                  DCASLR
01m0226                                                 ;;;             ;(i think)
01m0226 12277 010730170000060554041623660314160000             D[DCADSP] MASK[16.] DEST[Q] NORM $
01m0226                                                                 ;Offsets from interrupt dispatch are negative at start at
01m0226
01m0226                                                                 ;the first location preceding the interrupt.
01m0226 12300 011701370000060550605623661314160000             D[DCACPR] ROT[1 + 34.] MASK[2] ALU[D+1] DEST[AR] NORM $
01m0226                                                                 ;Calculate dispatch address while we have Q available
01m0226 12301 011611170000060540361103654314160000             D[AR] ROT[1] ALU[Q-D] DEST[DCAJMP] NORM $
01m0226                                                                 ;Multiply by two to get offset for address to select
01m0226                                                  registers
01 0226                                                 ].REPEAT 1 - OLDDLS
01 0227 12302 010730170000060554023623660714160000             D[DCASTATE] MASK[36. - 27.] DEST[Q] NORM $
01 0228                                                                 ;Get current CONI word, turning off CS, IACK RTN, RPLY
01 0229 12303 010631170000227102213020255714560000             D[CONST 5] ROT[35. - 26.] ALU[DORQ] DEST[DCASTATE] JUMP[MAIN]
01 0229                                                  NORM $
01 0230                                                                 ;Turn on CS and RPLY in CONI word
01 0231                                                                 ;Done with CONO
01 0232                                                 ;       ---
01 0233
01 0234                                                 ;Requested port doesn't exist.
01 0235 12304 010730170000060554023623660714160000     DCASG2: D[DCASTATE] MASK[36. - 27.] DEST[Q] NORM $
01 0236                                                                 ;Get current CONI word, turning off CS, IACK RTN, RPLY
01 0237 12305 010631170000060542211023655714160000             D[CONST 4] ROT[35. - 26.] ALU[DORQ] DEST[DCASTATE] NORM $
01 0238                                                                 ;Turn on CS, but not RPLY
01 0239 12306 010730170000060554161623661314160000             D[DCACPR] DEST[Q] NORM $
01 0240                                                                 ;Setup to indicate lack of card to respond
01 0241 12307 010631170000227102603040255714560000             D[CONST 1] ROT[35. - 24.] ALU[DORQ] DEST[DCACPR] JUMP[MAIN] NORM
01 0241                                                  $
01 0242                                                                 ;Turn on NO ADDR
01 0243                                                                 ;Done with CONO
01 0244
01 0245                                                 ;-----------------------------------------------------------------------
01 0245                                                 ----
01 0246                                                 ;
01 0247                                                 ;       DATAI K10,
01 0248                                                 ;
01 0249                                                 ; After select CONO
01 0250                                                 ;
01 0251                                                 ;        -------------- -------------- -------------- --------------
01 0252                                                 ;       ! NO !                 !                                      !
01 0253                                                 ;       !ADDR!  CARD FILE # !                  DATA                  !
01 0254                                                 ;       !    !                 !                                      !
01 0255                                                 ;        -------------- -------------- -------------- --------------
01 0256                                                 ;         24   25   26   27   28   29   30        32   33        35
01 0257                                                 ;
01 0258                                                 ; After IACK
01 0259                                                 ;
01 0260                                                 ;        -------------- -------------- -------------- --------------
01 0261                                                 ;       ! NO !                 !    !                 !              !
01 0262                                                 ;       !ADDR!  CARD FILE # !  1 !      PORT #       !    FLAGS     !
01 0263                                                 ;       !    !                 !    !                 !    !    !    !
01 0264                                                 ;        -------------- -------------- -------------- --------------
01 0265                                                 ;         24   25   26   27   28   29   30        32   33        35
01 0266                                                 ;
01 0267                                                 ;-----------------------------------------------------------------------
01 0267                                                 ------
01m0268                                                         .ORG[DCA-DISPATCH + 2]
01m0268                                                  [ xlist
01 0269 12167 010731004000000010161620260614560000       list ]DCADTI: D[DCASTATE] ROT[32.] COND[OBUS<0] JUMP[DCADIA] CS50 $
01 0270                                                                 ;Jump if last CONO did an IACK
01 0271 12170 010730170000000014027620260714560000             D[DCASTATE] MASK[36. - 25.] DEST[Q] JUMP[DCADI1] NORM $
01 0272                                                                 ;Clear CS
01m0273                                                         .RELOC
01m0273                                                  [.USE[HILOC]
01m0273                                                  [ xlist
01 0274 12310 010731006000101166161620260614560000       list ]                                ]DCADI1: D[DCASTATE] ROT[24.]
01 0274                                                  COND[-OBUS<0] JUMP[ILLINS] CS50 $
01 0275                                                                 ;Make sure CS was set.  If not, complain
01 0276 12311 010231170000060554161023654314160000             ALU[Q] DEST[DCASTATE] NORM $
01 0277                                                                 ;Set status with CS clear now that we know it was set.
01 0278 12312 010730170000000014161622261714560000             D[DCASEL] DEST[Q] PUSHJ[DCASLR] NORM $
01 0279                                                                 ;Get ready to set CS
01 0280 12313 010730170000060554161623661314160000             MAPF[DLSSEL] D[DCACPR] DEST[Q] NORM $
01 0281                                                                 ;Get CARD FILE#
01 0282 12314 010650170030060554021623647314160000             MAPF[DLSSEL] D[MASK 8.] ALU[-D&Q] DEST[Q] SPEC[IOB-IN] NORM $
01 0283                                                                 ;Make space for data and start reading it.
01 0284                                                         MAPF[DLSSEL] CYLEN[IOB-IN]
01 0285 12315 010630170031060554021623657114160000                             D[IOD] MASK[8] ALU[DORQ] DEST[Q] SPEC[IOB-OUT] $
01 0285
01 0286                                                                 ;Return port and data.
01 0287                                                                 ;Start turning off CS in hardware.
01 0288 12316 010231310000226654161561254104560000             MAPF[DLSSEL] CYLEN[IOB-OUT] ALU[Q] SMAC  [ IFRQ DEST[MEMSTO AR]
01 0288                                                  COND[MA<20] LBJUMP[SEOI] ]$
01 0289                                                 ;       ---
01 0290                                                 ;Last operation was an IACK.  Return result of IACK, which was saved in
01 0290                                                  A-MEM
01 0291 12317 010731310000226654161561261304560000     DCADIA: D[DCACPR] DEST[MEMSTO] MEMST  [ SMAC  [ IFRQ DEST[MEMSTO AR]
01 0291                                                  COND[MA<20] LBJUMP[SEOI] ]]$
01 0292                                                                 ;Just return saved thing.
01 0293
01 0294                                                 ;-----------------------------------------------------------------------
```

```
01m0299                                                 .ORG[DCA-DISPATCH + 6]
01m0299                                                [ xlist
01 0300 12173 010731004000010117016162025061456000       list ]DCADTO: D[DCASTATE] ROT[32.] COND[DBUS<0] JUMP[ILLINS] C550 $
01 0301                                                                 ;Complain if last COND did an IACK
01 0302                                                .REPEAT 1 - F4SW[
01 0303                                                         FIXM1 JUMP[DCAD01] $
01 0304                                                                 ;Take map fault if necessary
01 0305                                                ];.REPEAT 1 - F4SW
01 0306                                                .REPEAT F4SW [
01 0307                                                         DFRQ ALU[MEMAC] DEST[HOLD] JUMP[DCAD01] $
01 0308                                                                 ;Get C(E) from either memory or ACs.
01m0309                                                ];[
01m0309 12174 010331170000000001416150005431256000              DFRQ ALU[MEMAC] DEST[HOLD] JUMP[DCAD01] $
01m0309                                                                 ;Get C(E) from either memory or ACs.
01 0309                                                ].REPEAT F4SW
01m0310                                                         .RELOC
01m0310                                                [.USE[HILOC]
01m0310                                                [ xlist
01 0311 12320 010731006000010116616162025061456000       list ]                                        ]DCAD01: D[DCASTATE] ROT[24.]
01 0311                                                COND[-DBUS<0] JUMP[ILLINS] C550 $
01 0312                                                                 ;Make sure CS was set.  If not, complain
01 0313                                                .REPEAT OLDDLS [
01 0314                                                         D[DCASEL] DEST[Q] PUSHJ[DCASLR] NORM $
01 0315                                                                 ;Get selection bits, go do select
01 0316                                                         MAPF[DLSSEL] D[CONST 41] ROT[35. - 15.] ALU[DORQ]
01 0317                                                                         DEST[IOD] SPEC[IOB-OUT] $
01 0318                                                                 ;Set CS & WE and start write.
01 0319                                                         MAPF[DLSSEL] CYLEN[IOB-OUT] ALU[Q] DEST[IOD] SPEC[IOB-OUT] $
01 0320                                                                 ;Finish write, clear WE
01 0321                                                ];.REPEAT OLDDLS
01 0322                                                .REPEAT 1 - OLDDLS [
01 0323                                                         D[DCASEL] DEST[Q] NORM PUSHJ[DCASLR] $
01 0324                                                                 ;Get selection bits
01 0325                                                         MAPF[DLSSEL] D[CONST 1] ROT[35. - 16.] ALU[DORQ] DEST[Q IOD]
01 0326                                                                         SPEC[IOB-OUT] NORM $
01 0327                                                                 ;Sets WE and clears CS
01 0328                                                         MAPF[DLSSEL] CYLEN[IOB-OUT] D[CONST 1] ROT[35. - 16.] ALU[Q-D]
01 0328                                                 DEST[IOD]
01 0329                                                                         SPEC[IOB-OUT] $
01 0330                                                                 ;Clear WE to finish write.
01m0331                                                ];[
01m0331 12321 010730170000000001416162226171456000              D[DCASEL] DEST[Q] NORM PUSHJ[DCASLR] $
01m0331                                                                 ;Get selection bits
01m0331                                                         MAPF[DLSSEL] D[CONST 1] ROT[35. - 16.] ALU[DORQ] DEST[Q IOD]
01m0331 12322 010630170031060544603523655714116000                     SPEC[IOB-OUT] NORM $
01m0331                                                                 ;Sets WE and clears CS
01m0331                                                         MAPF[DLSSEL] CYLEN[IOB-OUT] D[CONST 1] ROT[35. - 16.] ALU[Q-D]
01m0331                                                 DEST[IOD]
01m0331 12323 011611170031060544603523655514116000                     SPEC[IOB-OUT] $
01m0331                                                                 ;Clear WE to finish write.
01 0331                                                ].REPEAT 1 - OLDDLS
01 0332 12324 010730171400060554027623660514116000              MAPF[DLSNOSEL] CYLEN[IOB-OUT] D[DCASTATE] MASK[36. - 25.]
01 0332                                                 DEST[Q]$
01 0333                                                                 ;Finish clear. Clear CS in microcode.
01 0334 12325 010231171400227114161020254314566000              MAPF[DLSNOSEL] ALU[Q] DEST[DCASTATE] JUMP[MAIN] NORM $
01 0335                                                                 ;We have to keep the MAPF field at NOSEL to keep data up!
01 0336
01 0336                                                ;       ---
01 0337
01 0338                                                ;IOTs not supported (BLKI and BLKO)
01m0339                                                         .ORG[DCA-DISPATCH]
01m0339                                                 [ xlist
01 0340 12165 010731170000101174161620254314566000       list ]        JUMP[ILLINS] NORM $
01 0341
01m0342                                                         .ORG[DCA-DISPATCH + 4]
01m0342                                                 [ xlist
01 0343 12171 010731170000101174161620254314566000       list ]        JUMP[ILLINS] NORM $
01m0344                                                         .RELOC
01m0344                                                 [.USE[HILOC]
01m0344                                                 [ xlist
01 0345                                                   list ]                                        ]
01 0346                                                ;Set CS, then dispatch according to register number
01 0347 12326 010631170000060545003523655714116000     DCASLR: D[CONST 1] ROT[35. - 15.] ALU[DORQ] DEST[IOD] NORM $
01 0348                                                                 ;Set CS then dispatch
01 0349 12327 010731170031000014160007262114566000             D[DCAJMP] SPEC[IOB-OUT] XDISP1  [ ODISP LONG ]$
01 0350                                                ;       ---
01 0351
01 0352                                                ;Following locations must be in order, up to and including DCAINT, as
01 0352                                                 they are
01 0353                                                ;dispatched to based on the register number.
01 0354 12330 010630170300060554021623654515160000             MAPF[DLSSEL + 3] CYLEN[IOB-OUT] D[MEM] MASK[8] ALU[DORQ] DEST[Q]
01 0354                                                 $
01 0355                                                                 ;Finish setting CS in hardware
01 0356                                                                 ;Combine data and selection stuff in case we're writing.
01 0356
01 0357 12331 010731170300060554161524261714160000             MAPF[DLSSEL + 3] D[DCASEL] DEST[IOD] POPJ NORM $
01 0358                                                                 ;One more cycle for register select (sigh...)
01 0359                                                                 ;Set IOD to turn off CS in case we're reading.
01 0360 12332 010630170200060554021623654515160000             MAPF[DLSSEL + 2] CYLEN[IOB-OUT] D[MEM] MASK[8] ALU[DORQ] DEST[Q]
01 0360                                                 $
01 0361 12333 010731170200060554161524261714160000             MAPF[DLSSEL + 2] D[DCASEL] DEST[IOD] POPJ NORM $
01 0362 12334 010630170100060554021623654515160000             MAPF[DLSSEL + 1] CYLEN[IOB-OUT] D[MEM] MASK[8] ALU[DORQ] DEST[Q]
01 0362                                                 $
01 0363 12335 010731170100060554161524261714160000             MAPF[DLSSEL + 1] D[DCASEL] DEST[IOD] POPJ NORM $
01 0364 12336 010630170000060554021623654515160000             MAPF[DLSSEL + 0] CYLEN[IOB-OUT] D[MEM] MASK[8] ALU[DORQ] DEST[Q]
01 0364                                                 $
01 0365 12337 010731170000060554161524261714160000             MAPF[DLSSEL + 0] D[DCASEL] DEST[IOD] POPJ NORM $
01 0366                                                ;Preceding locations must be in order, and must immediately preceed
01 0366                                                 DCAINT
01 0367                                                ;-------------------------------------------------------------------------
01 0367                                                ------                                                  .
01 0368                                                ;
01 0369                                                ;       DCA micro-interrupt code
01 0370                                                ;
01 0371                                                ;-------------------------------------------------------------------------
01 0371                                                ------
01 0372 12340 010241170031060554161523654314160000     DCAINT: ALU[0] DEST[IOD] SPEC[IOB-OUT] NORM $
01 0373                                                                 ;Disable further micro-interrupts until this interrupt is
01 0373                                                 acted
01 0374                                                                 ;upon by the macrocode
```

```
01 0380 12343 010731200000203434007620260614560000             D[DCASTATE] MASK[3] DEST[AR] COND[-OBUS=0] JUMP[PIGEN] C550 $
01 0381                                                                 ;Take macro-interrupt if enabled for such
01 0382 12344 010731170040227114161620254314560000             CLR-DEV-FROM-INTR FETCH-NEXT-INST  [ JUMP[MAIN] ]NORM $
01 0383                                                                 ;Start fetch and execute next instruction
01 0384
01 0385
01 0386                                                 ;-------------------------------------------------------------------------
01 0387                                                 ;
01 0388                                                 ;       Reset DCA scanner, Set interrupt and dispatch addresses
01 0389                                                 ;
01 0390                                                 ;-------------------------------------------------------------------------
01 0391 12345 010731170040247154040460255714570000     DCARST: D[CONST DCA-UDEV] DEST[DEV-ADR] NORM JUMP[. + 1] $
01 0392                                                                 ;Start IOB RESET happening.
01 0393 12346 010731170031060546003523655714160000             D[CONST 1] ROT[24.] DEST[IOD] SPEC[IOB-OUT] NORM $
01 0394                                                                 ;"Hit MASTER RESET on all ASTRAs"
01 0395 12347 010241171000060554161523654114160000             MAPF[DLSCSR] CYLEN[LONG] ALU[0] DEST[IOD] $
01 0396                                                                 ;Setup to clear it later
01 0397                                                 .REPEAT 1 - F4SW [
01 0398                                                         D[CONST (DCA-DISPATCH / 100)] ROT[30] DEST[Q] $
01 0399                                                                 ;Construct dispatch address: high 6 bits.  Finish IOB
01 0399                                                  RESET
01 0400                                                         D[CONST (DCA-DISPATCH \ 100)] ROT[22] ALU[DORQ] DEST[Q] NORM $
01 0401                                                                 ;Low 6 bits
01 0402                                                         D[CONST (DCAINT / 100)] ROT[6] ALU[DORQ] DEST[Q] NORM $
01 0403                                                                 ;Construct interrupt address: high 6 bits
01 0404                                                         D[CONST (DCAINT \ 100)] ALU[DORQ] DEST[DCADSP] NORM $
01 0405                                                                 ;Finish contructing interrupt address and store away.
01 0406                                                 ];.REPEAT 1 - F4SW
01 0407                                                 .REPEAT F4SW [
01 0408                                                         D[LIT ((DCA-DISPATCH * 1000000) + DCAINT)] DEST[DCADSP] NORM $
01m0409                                                 ];[
01m0409 12350 010731170021352024701003655314160000             D[LIT ((DCA-DISPATCH * 1000000) + DCAINT)] DEST[DCADSP] NORM $
01 0409                                                 ].REPEAT F4SW
01 0410 12351 010731170031060554001023655714160000             D[CONST 0] DEST[DCASTATE] SPEC[IOB-OUT] NORM $
01 0411                                                                 ;Initial state
01 0412 12352 010731171000060554001044255514160000             MAPF[DLSCSR] CYLEN[LONG] D[CONST 0] DEST[DCACPR] POPJ $
01 0413                                                                 ;Nothing referenced yet.
```

```
01 0106                                                         ;Alternate scanner code (F2 or F4)
01 0105                                                    ]
01 0107                                                   .repeat IMP [
01 0108                                                       .insert F4IMPi
01 0109                                                          ;Get the IMP code.
01 0110                                                    ]
01 0111
01 0112                                                   .repeat VC [
01 0113                                                       .insert VC
01 0114                                                          ;Get the Versatec code
01m0115                                                    ]
01m0115                                                  [
01m0115                                                       .insert VC
```

```
01m0001       0 VCDEB = 0       ; Nonzero for debugging microcode              ;! VCDSP
01m0001                                                                        VCORG VCINT VCINT2 VCINT3 VCIOT VCODSP VCCMD VCRST VCONO VCONO0 VCONO1
01m0001                                                                        VCONO2 VCSETZ VCSTAT VCONSZ VCONSO VCGST VCOUNT VCOUT VCGO VCBRST VCLP
01m0001                                                                        VCDONE VCLST VCWRD VCWRD1 MA VCOLP VCOUT1 VCODLY
01m0001
01m0001       1 VCNEWWIRING = WAITS             ;Isn't WAITS specific, but only CCRMA
01m0001                                                                        doesn't
01m0001                                                                                                  ;have long-line (differential) Versatec
01m0001                                                                        and
01m0001                                                                                                  ;must use new backplane wiring
01m0001       11344     VCBAD = MAIN
01m0001
01m0001                                                                        ; Versatec now has its own device code 540, with 544 reserved for a
01m0001                                                                        second VC.
01m0001                                                                        VCDSP:
01m0001                                                                        :. + 20          ;Reserve space for dispatch table
01m0001                                                                        ;Let's save this for a second VC
01m0001                                                                        .REPEAT 10 [     ILGIOT $
01m0001                                                                                         NOP $
01m0001 12373 540731170000060554161623254204160000                            ];[      ILGIOT $
01m0001 12374 010731170000060554161623654314160000                                     NOP $
01m0001 12375 540731170000060554161623254204160000                            ][       ILGIOT $
01m0001 12376 010731170000060554161623654314160000                                     NOP $
01m0001 12377 540731170000060554161623254204160000                            ][       ILGIOT $
01m0001 12400 010731170000060554161623654314160000                                     NOP $
01m0001 12401 540731170000060554161623254204160000                            ][       ILGIOT $
01m0001 12402 010731170000060554161623654314160000                                     NOP $
01m0001 12403 540731170000060554161623254204160000                            ][       ILGIOT $
01m0001 12404 010731170000060554161623654314160000                                     NOP $
01m0001 12405 540731170000060554161623254204160000                            ][       ILGIOT $
01m0001 12406 010731170000060554161623654314160000                                     NOP $
01m0001 12407 540731170000060554161623254204160000                            ][       ILGIOT $
01m0001 12410 010731170000060554161623654314160000                                     NOP $
01m0001 12411 540731170000060554161623254204160000                            ][       ILGIOT $
01m0001 12412 010731170000060554161623654314160000                                     NOP $
01 0001                                                                        ].REPEAT 10
01 0002
01 0003                                                                        VCORG:           ; Origin of Versatec code
01 0004
01 0005                                                                        ; Versatec driver definitions and register usage
01 0006                                                                        ;
01 0007                                                                        ; CONO - command out.  Takes (E) as
01 0008                                                                        ;               Bits 0-15: Command to Versatec
01 0009                                                                        ;               Bits 16-28: Unused
01 0010                                                                        ;               Bit 29: Load PI and enable only
01 0011                                                                        ;               Bit 30: Enable interrupt on READY
01 0012                                                                        ;               Bits 31-32: Mode 00 - unused (same as mode 01)
01 0013                                                                        ;                                   01 - 7-bit bytes, 5 per word, left
01 0013                                                                        justified
01 0014                                                                        ;                                   10 - 8-bit bytes, 4 per word, left
01 0014                                                                        justified
01 0015                                                                        ;                                   11 - 8-bit bytes, 9 per two words
01 0016                                                                        ;               Bits 33-35: PI channel
01 0017                                                                        ;
01 0018                                                                        ; CONI - status in.  Puts in E
01 0019                                                                        ;               Bits 0-17: Zero, except
01 0020                                                                        ;                       3-4, 10-11: State of corresponding
01 0021                                                                        ;                                   command out bits.
01 0022                                                                        ;               Bits 18-23: Status bits from Versatec
01 0023                                                                        ;               Bit 24: Reserved
01 0024                                                                        ;               Bit 25: Versatec READY bit
01 0025                                                                        ;               Bit 26: Data ready bit
01 0026                                                                        ;               Bits 27-29: zero
01 0027                                                                        ;               Bit 30: ENB-RDY-INT
01 0028                                                                        ;               Bits 31-35: Mode and PI from last command out
01 0029                                                                        ;
01 0030                                                                        ; CONSZ, CONSO - Test right half of above CONI bits.
01 0031                                                                        ;
01 0032                                                                        ; DATAO - Start output.  Takes (E) as
01 0033                                                                        ;               Bits 0-17: 2's complement byte count (0 means 256K bytes)
01 0033
01 0034                                                                        ;               Bits 18-35: Starting memory address
01 0035                                                                        ;       This operation sets Busy, clears Done, and starts the output.
01 0036                                                                        ;       When output finishes, Busy clears and Done sets.
01 0037                                                                        ;
01 0038                                                                        ; The following are debugging aids
01 0039                                                                        ;
01 0040                                                                        ; BLKI - Enters the interrupt routine.
01 0041                                                                        ; DATAI - Returns the remaining byte count/MA.
01 0042                                                                        ; BLKO - Unused.
01 0043                                                                        ;
01 0044                                                                        ; A-MEM Usage:
01m0045                                                                        DEFINE-A-MEM[VC-DISP 0]  [
01m0045       40                  VC-DISP = 0 + D%AMEM0
01 0045                                                                        ]       ;0 - Interrupt Dispatch
01m0046                                                                        DEFINE-A-MEM[VC-MA 1]    [
01m0046       41                  VC-MA = 1 + D%AMEM0
01 0046                                                                        ]       ;1 - MA saved between bursts
01m0047                                                                        DEFINE-A-MEM[VC-BC 2]    [
01m0047       42                  VC-BC = 2 + D%AMEM0
01 0047                                                                        ]       ;2 - byte count saved between bursts
01m0048                                                                        DEFINE-A-MEM[VC-MODE 3]  [
01m0048       43                  VC-MODE = 3 + D%AMEM0
01 0048                                                                        ]       ;3 - Constants depending on Mode
01 0049                                                                                                 ;       Bit 35: 0 for modes 1&2, 1 for
01 0049                                                                         mode 3
01 0050                                                                                                 ;       Bits 32-34: Bytes/Word - 1
01 0051                                                                                                 ;       Bits 25-31: Bytes/burst
01m0052                                                                        DEFINE-A-MEM[VC-STAT 4]  [
01m0052       44                  VC-STAT = 4 + D%AMEM0
01 0052                                                                        ]       ;4 - Command/Status Register
01m0053                                                                        DEFINE-A-MEM[VC-CMD-MASK 5]      [
01m0053       45                  VC-CMD-MASK = 5 + D%AMEM0
01 0053                                                                        ];5 - Mask for command bits that are levels.
01m0054                                                                        DEFINE-A-MEM[VC-AC-SAVE 6]       [
01m0054       46                  VC-AC-SAVE = 6 + D%AMEM0
01 0054                                                                        ];6 - Saved AC during output burst
01m0055                                                                        DEFINE-A-MEM[VC-PC-SAVE 7]       [
01m0055       47                  VC-PC-SAVE = 7 + D%AMEM0
01 0055                                                                        ];7 - Saved PC during output burst (F2 only)
```

```
01 0061        1 VC-LD-CTRL = 1
01 0062                                                   ;        Bit 0        -CLEAR
01 0063                                                   ;        Bit 1        -RESET
01 0064                                                   ;        Bit 2        -REMCUT
01 0065                                                   ;        Bit 3        -RCUTDIS
01 0066                                                   ;        Bit 4        -AFFDIS
01 0067                                                   ;        Bit 5        -ALTER
01 0068                                                   ;        Bit 6        -AFFED
01 0069                                                   ;        Bit 7        -REOTR
01 0070                                                   ;        Bit 9        -RTTON
01 0071                                                   ;        Bit 10       PRINT
01 0072                                                   ;        Bit 11       -SPP
01 0073        2 VC-RD-STATUS = 2
01 0074                                                   ;        Bit 0        -ONLIN
01 0075                                                   ;        Bit 1        LOSUP
01 0076                                                   ;        Bit 2        -BUFUL
01 0077                                                   ;        Bit 3        NOPUP
01 0078                                                   ;        Bit 4        -TNERON
01 0079                                                   ;        Bit 5        -DXTER
01 0080                                                   ;        Bit 6        -READY
01 0081        3 VC-LD-DATA = 3
01 0082                                                   ;        Bit 1        FIFO IN LAST
01 0083                                                   ;        Bits 28:35   Data
01 0084        4 VC-LD-CTL = 4
01 0085                                                   ;        Bit 32       EMPTY ENB
01 0086                                                   ;        Bit 33       ATTENTION ENB
01 0087        5 VC-CLR-INT = 5
01 0088                                                   ;        Clears MAKE INT RQ
01 0089                                                   ;
01 0090                                                   ; Other registers:
01 0091                                                   ;              PC - Addresses memory during output
01 0092                                                   ;              AC - Counts bytes during output
01 0093                                                   ;              Loop Count - counts bytes in a word
01 0094                                                   ;              Q, AR, HOLD - temporary
01 0095                                                   ;
01 0096                                                   : VCDSP + 0
01 0097                                                   .REPEAT 1 - VCDEB [      ; Make debugging ops illegal (NOP)
01 0098                                                   .repeat 3 [     ILGIOT $ NOP $
01 0099                                                   ] ; .REPEAT 3
01m0100                                                   ] [     ; Make debugging ops illegal (NOP)
01m0100                                                   .repeat 3 [     ILGIOT $ NOP $
01m0100 12353 540731170000060554161623254204l6000         ] [     ILGIOT $ NOP $
01m0100 12354 010731170000060554161623654314l6000
01m0100 12355 540731170000060554161623254204l6000         ][      ILGIOT $ NOP $
01m0100 12356 010731170000060554161623654314l6000
01m0100 12357 540731170000060554161623254204l6000         ][      ILGIOT $ NOP $
01m0100 12360 010731170000060554161623654314l6000
01 0100                                                   ]; .REPEAT 3
01 0100                                                   ]; .REPEAT 1 - VCDEB
01 0101
01 0102                                                   .REPEAT VCDEB [          ; BLKI - act like the Versatec interrupted
01 0103
01 0104                                                          JUMP[.] NORM $
01 0105                                                          D[CONST VCDEV] DEST[DEV-ADR] JUMP[VCINT] NORM $
01 0106                                                   ]
01 0107
01 0108                                                   : VCORG + 2      ; Dispatch base and interrupt entry
01 0109
01 0110                                                   ; Dispatch for modes 1-3 relative to the vector (2 words/entry)
01 0111                                                   ;              Set burst count for the mode
01 0112 12415 010730170000060541437623655714l6000                D[CONST 17] ROT[6] DEST[Q] NORM $ ;Mode 1 - 60 bytes/burst, 5
01 0112                                                    bytes/word
01 0113 12416 010631170000000140210602557145600 0                D[CONST 10] ALU[DORQ] DEST[VC-MODE] JUMP[VCSET2] NORM $
01 0114                                                   ;
01 0115 12417 010730170000060541441623655714l6000                D[CONST 20] ROT[6] DEST[Q] NORM $ ;Mode 2 - 64 bytes/burst, 4
01 0115                                                    bytes/word
01 0116 12420 01063117000000014015060255714560000                D[CONST 06] ALU[DORQ] DEST[VC-MODE] JUMP[VCSET2] NORM $
01 0117                                                   ;
01 0118 12421 010730170000060541437623655714l6000                D[CONST 17] ROT[6] DEST[Q] NORM $ ;Mode 3 - 63 bytes/burst, 4
01 0118                                                    bytes/word
01 0119 12422 01063117000000014157060255714560000                D[CONST 67] ALU[DORQ] DEST[VC-MODE] JUMP[VCSET2] NORM $
01 0120                                                   ;
01 0121                                                   VCINT:          ;Interrupt routine
01 0122
01 0123                                                   : VCORG
01 0124                                                   .REPEAT VCDEB [
01 0125                                                          JUMP[.] NORM $
01 0126                                                          JUMP[VCINT] NORM $
01 0127                                                   : VCINT
01 0128                                                   ]
01 0129 12413 010241170031060554161523654314l6000                ALU[0] DEST[IOD] SPEC[IOB-OUT] NORM $ ;Disable interrupts
01 0130                                                          MAPF[VC-LD-CTL] CYLEN[IOB-OUT] SPEC[IOB-OUT]
01 0131                                                   .REPEAT VCDEB [ $ ]
01 0132                                                   .REPEAT 1 - VCDEB [ JUMP[VCINT] $
01 0133                                                   : VCINT
01m0134                                                   ]
01m0134 12414 010731170431250474161620254114560000        [ JUMP[VCINT] $
01m0134                                                   : VCINT
01 0134 12423 010731170031060554161623654114l6000         ]      MAPF[VC-CLR] SPEC[IOB-OUT] CYLEN[IOB-OUT] $ ; Clear last-byte bit
01 0134
01 0135                                                          MAPF[VC-CLR-INT] CYLEN[IOB-OUT] ; Clear request
01 0136 12424 010731000500000141616202611145600 0                       D[VC-BC] COND[-OBUS=0] JUMP[VCGO] $ ;Output if count/= 0
01 0136
01 0137
01 0138 12425 010730170000060554161623662314l6000                D[VC-STAT] DEST[Q] NORM $ ; Fetch status register
01 0139 12426 01073100400000007561620262214560000                D[VC-STAT] ROT[30.] COND[OBUS<0] JUMP[VCINT2] C550 $
01 0140
01 0141                                                   ; End of data transfer.  Set data ready bit.
01 0142 12427 01063117000000002203100255714560000                D[CONST 1] ROT[9.] ALU[DORQ] DEST[VC-STAT] JUMP[VCINT3] NORM $
01 0143
01 0144                                                   ; READY interrupt.  Clear ready int enable bit.
01 0145 12430 010651170000060541203103655714l5000         VCINT2: D[CONST 1] ROT[5] ALU[-D&Q] DEST[VC-STAT] NORM $
01 0146
01 0147                                                   VCINT3: D[VC-STAT] MASK[3] DEST[AR Q] CLR-DEV-FROM-INTR
01 0148 12431 01073020004020343400762026221456000                        COND[-OBUS=0] JUMP[PIGEN] C550 $ ; Macro int if enabled
01 0149 12432 01073117060022711416162025431456000                JUMP[MAIN] NORM $
01 0150
01 0151                                                   ;IOT dispatch
01m0152                                                            .PAIR
01m0152                                                    [:. \ 2 + .
01 0153 12434
```

```
01 0157                                                         ;machine has already done indexing/indirection and bits
01 0158                                                         ;13:17 are guaranteed zero
01 0159                                                 .REPEAT 1 - F4SW [
01 0160                                                         D[VC-DISP] ROT[18.] MASK[16.] ALU[D+Q] SDISP CYLEN[DISP] $
01 0161                                                         ;Dispatch of type of IOT
01 0162                                                 ];.REPEAT 1 - F4SW
01 0163                                                 .REPEAT F4SW [
01 0164                                                         D[VC-DISP] ROT[18.] MASK[16.] ALU[D+Q] DEST[AR] $
01 0165                                                 VCODSP: D[AR] ROT[MUA-ROT] CDISP LONG $
01m0166                                                 ];[
01m0166 12436 010601370000060544441623660316160000             D[VC-DISP] ROT[18.] MASK[16.] ALU[D+Q] DEST[AR] $
01m0166 12437 010731170000000000160007251145600 0      VCODSP: D[AR] ROT[MUA-ROT] CDISP LONG $
01 0166                                                 ].REPEAT F4SW
01 0167
01 0168                                                 ;
01 0169                                                 ; Command out to Versatec
01 0170
01 0171                                                 VCCMD:
01 0172                                                 : VCDSP + 10 ; CONO dispatch
01 0173                                                 .REPEAT VCDEB [
01 0174                                                         JUMP[.] NORM $
01 0175                                                         JUMP[VCCMD] NORM $
01 0176                                                 : VCCMD
01 0177                                                 ]
01 0178 12363 010731170041000001403046225571457000             D[CONST VCDEV] DEST[DEV-ADR] PUSHJ[VCONO] NORM $
01 0179 12364 010731170000227114161620254314560 0              JUMP[MAIN] $
01 0180                                                 .REPEAT 1 - VCDEB [
01 0181                                                 : VCCMD
01m0182                                                 ]
01m0182                                                 [
01m0182                                                 : VCCMD
01 0182                                                 ]; Reset.
01 0183
01 0184 12440 010730170000060547415623655714160000     VCRST:  D[CONST 6] ROT[30.] DEST[Q] NORM $ ; mask for VC mode bits
01 0185 12441 010631170000060546007123655714160000             D[CONST 3] ROT[24.] ALU[DORQ] DEST[VC-CMD-MASK] NORM $
01 0186
01 0187                                                 .REPEAT 1 - F4SW [
01 0188                                                         D[CONST VCDSP / 100] ROT[6 + 18.] DEST[Q] $     ;Setup dispatch
01 0189                                                         D[CONST VCDSP \ 100] ROT[18.] ALU[DORQ] DEST[Q] $
01 0190                                                         D[CONST VCORG / 100] ROT[6] ALU[DORQ] DEST[Q] $ ;Interrupt base
01 0190                                                  address
01 0191                                                         D[CONST VCORG \ 100] ALU[DORQ] DEST[VC-DISP] NORM $
01 0192                                                 ];.REPEAT 1 - F4SW
01 0193                                                 .REPEAT F4SW [
01 0194                                                         D[LIT ((VCDSP * 1000000) + VCORG)] DEST[VC-DISP] NORM $
01m0195                                                 ];[
01m0195 12442 010731170024726025027003655314160000             D[LIT ((VCDSP * 1000000) + VCORG)] DEST[VC-DISP] NORM $
01 0195                                                 ].REPEAT F4SW
01 0196
01 0197 12443 010731170000000014021500255714560000             D[CONST 10] DEST[HOLD] JUMP[VCONO0] NORM $ ; Invent CONO 124,[10]
01 0197
01 0198
01 0199                                                 ; CONO routine
01 0200
01 0201 12444 010331170000060554161503454312160000     VCONO: FIXM1  [ DFRQ ALU[MEMAC] DEST[HOLD] ]$
01 0202
01 0203 12445 010731370000060547603623654715160000     VCONO0: D[MEM] ROT[31.] MASK[1] DEST[AR] NORM $ ; Save int-on-rdy bit
01 0204                                                         SPEC[IOB-OUT]
01 0205 12446 010731004031000007361620254615560000             D[MEM] ROT[29.] COND[OBUS<0] JUMP[VCONO4] C550 $
01 0206                                                                 ;If just setting enables, don't touch FIFO!!
01 0207
01 0208                                                 .REPEAT 1 - VCNEWWIRING [
01 0209                                                         MAPF[VC-CLR] SPEC[IOB-OUT] CYLEN[IOB-OUT] ; Clear the FIFO
01 0210                                                         D[MEM] DEST[IOD Q] $ ;Set up the control bits
01 0211
01 0212                                                         MAPF[VC-LD-CTRL] SPEC[IOB-OUT] LONG ;Output control bits
01 0213                                                         D[VC-CMD-MASK] ALU[D&Q] DEST[IOD Q] $ ;Clear the ones
01 0213                                                  that pulse.
01 0214
01 0215                                                 .REPEAT F4SW [
01 0216                                                         MAPF[VC-CLR-INT] LONG $         ;Wait some, the F4 is faster.
01 0217                                                 ];.REPEAT F4SW
01 0218
01 0219                                                         MAPF[VC-CLR-INT] SPEC[IOB-OUT] LONG ; Clear int req
01 0220                                                         D[MEM] MASK[6] ALU[DORQ] DEST[Q] $ ; Save the mode
01 0221                                                 ];.REPEAT 1 - VCNEWWIRING
01 0222                                                 .REPEAT VCNEWWIRING [
01 0223                                                         MAPF[VC-CLR] CYLEN[IOB-OUT]
01 0224                                                                 D[MASK 12.] ROT[35. - 12.] DEST[Q] $
01 0225                                                                                         ;Clear the FIFO
01 0226                                                                                         ;Invert the screwed up levels
01 0227
01 0228                                                         D[MEM] ALU[D#Q] DEST[Q IOD] SPEC[IOB-OUT] NORM $ ;Set up the
01 0228                                                  control bits
01 0229
01 0230                                                         MAPF[VC-LD-CTRL] SPEC[IOB-OUT] LONG     ;Output control bits
01 0231                                                 .REPEAT WAITS * (1 - F4SW) [ $ NORM ]   ;Bug:  Insufficient hold time
01 0232                                                         D[MASK 10.] ROT[35. - 8.] ALU[DORQ] DEST[IOD] $
01 0233                                                                                         ;"Clear" the ones that
01 0233                                                  pulse.
01 0234
01 0235                                                         SPEC[IOB-OUT] D[MEM] DEST[Q] NORM $     ;Setup to save stuff.
01 0236
01 0237                                                         MAPF[VC-CLR-INT] LONG                   ; Clear int req
01 0238                                                                 D[VC-CMD-MASK] ALU[D&Q] DEST[Q] $ ;Bits to save for CONI
01 0238
01 0239
01 0240                                                         MAPF[VC-CLR-INT] SPEC[IOB-OUT] LONG
01 0241                                                                 D[MEM] MASK[6] ALU[DORQ] DEST[Q] $       ;Save the mode
01m0242                                                 ];[
01m0242                                                         MAPF[VC-CLR] CYLEN[IOB-OUT]
01m0242 12447 010730170000060545631623647114160000             D[MASK 12.] ROT[35. - 12.] DEST[Q] $
01m0242                                                                                 ;Clear the FIFO
01m0242                                                                                 ;Invert the screwed up levels
01m0242
01m0242 12450 010660170031060554161523654715160000             D[MEM] ALU[D#Q] DEST[Q IOD] SPEC[IOB-OUT] NORM $ ;Set up the
01m0242                                                  control bits
01m0242
01m0242                                                         MAPF[VC-LD-CTRL] SPEC[IOB-OUT] LONG     ;Output control bits
01m0242                                                 .REPEAT WAITS * (1 - F4SW) [ $ NORM ]   ;Bug:  Insufficient hold time
01m0242 12451 010631170131060546625523647114160000             D[MASK 10.] ROT[35. - 8.] ALU[DORQ] DEST[IOD] $
```

```
01m0242
01m0242                                                 MAPF[VC-CLR-INT] LONG                   ; Clear int req
01m0242 12453 010640170500060554161623662514116000             D[VC-CMD-MASK] ALU[D&Q] DEST[Q] $ ;Bits to save for CONI
01m0242
01m0242
01m0242                                                 MAPF[VC-CLR-INT] SPEC[IOB-OUT] LONG
01m0242 12454 010630170531060554015623654515116000             D[MEM] MASK[6] ALU[DORQ] DEST[Q] $       ;Save the mode
01 0242                                                 ].REPEAT VCNEWWIRING
01 0243
01 0244                                                 MAPF[VC-LD-CTRL] SPEC[IOB-OUT] LONG ; Clear the pulsed bits
01 0245 12455 010731170131060540561523654114116000             D[AR] ROT[2] DEST[IOD] $ ;Set up the int-on-rdy bit
01 0246
01 0247                                                 MAPF[VC-LD-CTL] CYLEN[IOB-OUT] ; Set interrupt enablings
01 0248 12456 010731170400060554161623654114116000     .REPEAT F4SW [ $ ]      [ $ ];Bug:  F4 can't do A-MEM write during
01 0248                                                  IOB-OUT cycle
01 0249 12457 010631170000060542203103655714116000             D[CONST 1] ROT[9.] ALU[DORQ] DEST[VC-STAT] $ ;Set data ready bit
01 0249
01 0250
01 0251                                                 D[MEM] ROT[33.] MASK[2] DEST[AR] ;Extract the mode
01 0252 12460 010731202000000010205620254615560000             COND[OBUS=0] JUMP[VCONO1] CS50 $
01 0253 12461 010730170000000000361620254314560000     D[AR] ROT[1] DEST[Q] JUMP[VCONO2] NORM $ ; Double mode to Q
01 0254
01 0255 12462 010730170000060540203623655714116000     VCONO1: D[CONST 1] ROT[1] DEST[Q] NORM $ ;Change mode 0 to 1 and double
01 0256
01 0257                                                 VCONO2:
01 0258                                                 .REPEAT 1 - F4SW [
01 0259                                                         D[VC-DISP] ALU[D+Q] SDISP CS50 $ ;Set up constants by mode.
01 0260                                                 ];.REPEAT 1 - F4SW
01 0261                                                 .REPEAT F4SW [
01 0262                                                         D[VC-DISP] MASK[16.] ALU[D+Q] DEST[AR] JUMP[VCODSP] $
01 0263                                                                                 ;Set up constants by mode.
01m0264                                                 ];[
01m0264 12463 010601370000250774041620260314560000             D[VC-DISP] MASK[16.] ALU[D+Q] DEST[AR] JUMP[VCODSP] $
01m0264                                                                                 ;Set up constants by mode.
01 0264                                                 ].REPEAT F4SW
01 0265                                                 ;Set only PI channel and ready interrupt enabling.
01 0266                                                 VCONO4: MAPF[VC-CLR-INT] CYLEN[IOB-OUT]
01 0267 12464 010731170531060540561523654114116000                     D[AR] ROT[2] DEST[IOD] SPEC[IOB-OUT] $
01 0268                                                         ;Clear previous interrupt perhaps
01 0269                                                         ;Set up the int-on-rdy bit
01 0270                                                 MAPF[VC-LD-CTL] CYLEN[IOB-OUT]
01 0271 12465 010770170400060554007623647114116000                     D[MASK 3] ALU[NOTD] DEST[Q] $
01 0272                                                         ;Get status, minus PI
01 0273 12466 010640170000060554161623662314116000     D[VC-STAT] ALU[D&Q] DEST[Q] NORM $
01 0274                                                         ;Remove from saved status
01 0275 12467 010630170000060554007623654715116000     D[MEM] MASK[3] ALU[DORQ] DEST[Q] NORM $
01 0276                                                         ;Save new PI status
01 0277 12470 010631170000060541361104254314116000     D[AR] ROT[35. - 30.] ALU[DORQ] DEST[VC-STAT] POPJ NORM $
01 0278                                                         ;Save new status
01 0279                                                 ;       ---
01 0280                                                 ;
01 0281                                                 ; Back from setup of constants.  Clear byte count.
01 0282                                                 ;
01 0283 12471 010241170000060554161044254314116000     VCSETZ: ALU[0] DEST[VC-BC] NORM POPJ $
01 0284                                                 ;
01 0285                                                 ; Read status/control bits
01 0286
01 0287                                                 VCSTAT:
01 0288                                                 : VCDSP + 12 ; CONI dispatch
01 0289
01 0290                                                 .REPEAT VCDEB [
01 0291                                                         JUMP[.] NORM $
01 0292                                                         JUMP[VCSTAT] NORM $
01 0293                                                 : VCSTAT
01 0294                                                 ]
01 0295 12365 010731170030000014030462255714570000             D[CONST VCDEV] DEST[DEV-ADDRESS] SPEC[IOB-IN] PUSHJ[VCGST] NORM $
01 0295
01 0296                                                 ;               Fetch device status into AR.
01 0297 12366 010731310000226654161561254304560000             D[AR] DEST[MEMSTO] MEMST  [ SMAC  [ IFRQ DEST[MEMSTO AR]
01 0297                                                  COND[MA<20] LBJUMP[SEOI] ]]$
01 0298                                                 .REPEAT 1 - VCDEB [
01 0299                                                 : VCSTAT
01m0300                                                 ]
01m0300                                                 [
01m0300                                                 : VCSTAT
01 0300                                                 ]VCONSZ:
01 0301                                                 : VCDSP + 14 ; CONSZ dispatch
01 0302                                                 .REPEAT VCDEB [
01 0303                                                         JUMP[.] NORM $
01 0304                                                         JUMP[VCONSZ] NORM $
01 0305                                                 : VCONSZ
01 0306                                                 ] ; VCDEB
01 0307 12367 010731170030000014030462255714570000             D[CONST VCDEV] DEST[DEV-ADDRESS] SPEC[IOB-IN] PUSHJ[VCGST] NORM $
01 0307
01 0308 12370 010730170000000001404562025771456000             D[IR] MASK[18.] DEST[Q] JUMP[CTYCZ] NORM $
01 0309                                                 .REPEAT 1 - VCDEB [
01 0310                                                 : VCONSZ
01m0311                                                 ]
01m0311                                                 [
01m0311                                                 : VCONSZ
01 0311                                                 ]VCONSO:
01 0312                                                 : VCDSP + 16 ; CONSO dispatch
01 0313                                                 .REPEAT VCDEB [
01 0314                                                         JUMP[.] NORM $
01 0315                                                         JUMP[VCONSO] NORM $
01 0316                                                 : VCONSO
01 0317                                                 ] ; VCDEB
01 0318 12371 010731170030000014030462255714570000             D[CONST VCDEV] DEST[DEV-ADDRESS] SPEC[IOB-IN] PUSHJ[VCGST] NORM $
01 0318
01 0319 12372 010730170000000001404562025771456000             D[IR] MASK[18.] DEST[Q] JUMP[CTYCS] NORM $
01 0320                                                 .REPEAT 1 - VCDEB [
01 0321                                                 : VCONSO
01m0322                                                 ]
01m0322                                                 [
01m0322                                                 : VCONSO
01 0322                                                 ]
01 0323 12472 010731370200060542021623655711416000     VCGST:  MAPF[VC-RD-STATUS] D[IOD] ROT[8.] MASK[8.] DEST[AR] CYLEN[IOB-IN]
01 0323                                                  $
01 0324 12473 010730170000060542561623654314116000             D[AR] ROT[18.] DEST[Q] NORM $ ;Merge status with command.
01 0325 12474 010631370000060554161624262314116000             D[VC-STAT] ALU[DORQ] DEST[AR] POPJ NORM $
```

```
01 0331                                                         D[CONST VCDEV] DEST[DEV-ADR] JUMP[VCOUNT] NORM $
01 0332                                                       : VCOUNT
01 0333                                                         D[VC-BC] ROT[18.] MASK[0] SPEC[LEFT] DEST[Q] NORM $
01 0334                                                         D[VC-MA] MASK[18.] ALU[DORQ] DEST[MEMSTD] MEMST $
01 0335                                                       ];.REPEAT VCDEB
01 0336                                                       VCOUT:
01 0337                                                       : VCDSP + 6 ; DATAO dispatch
01 0338
01 0339                                                       .REPEAT VCDEB [
01 0340                                                         JUMP[.] NORM $
01 0341                                                         JUMP[VCOUT] NORM $
01 0342                                                       : VCOUT
01 0343                                                         FIXM1 $
01 0344                                                         D[CONST VCDEV] DEST[DEV-ADR] NORM $
01 0345                                                       ]
01 0346                                                       .REPEAT 1 - VCDEB [
01 0347                                                         FIXM1 $
01 0348                                                         D[CONST VCDEV] DEST[DEV-ADR] JUMP[VCOUT] NORM $
01 0349                                                       : VCOUT
01m0350                                                       ]
01m0350                                                       [
01m0350 12361 010331170000060554161503454312160000             FIXM1  [ DFRQ ALU[MEMAC] DEST[HOLD] ]$
01m0350 12362 010731170040251734030460255714570000             D[CONST VCDEV] DEST[DEV-ADR] JUMP[VCOUT] NORM $
01m0350                                                       : VCOUT
01 0350                                                       ];              Illegal op if BUSY is on.
01 0351 12475 010731002000227106603620262214560000           D[VC-STAT] ROT[27.] MASK[1] COND[OBUS=0] JUMP[VCBAD] CYLEN[C550]
01 0351                                                        $
01 0352 12476 010730170000060554161623662314160000           D[VC-STAT] DEST[Q] NORM $
01 0353 12477 010651170000060542203103655714160000           D[CONST 1] ROT[9.] ALU[-D&Q] DEST[VC-STAT] NORM $ ; Clear DONE
01 0354                                                       ;D[LEFT] can be made to work by moving MODIFY-FIELD D to after MAPF
01 0354                                                        definitions
01 0355                                                       ;and adding LEFT'EXT = LEFT 2
01 0356 12500 010730170000060554141623647314160000           D[LEFT] DEST[Q] NORM $ ;Pad byte count with ones in left.
01 0357 12501 010631170000060544445043654715160000           D[MEM] ROT[18.] MASK[18.] ALU[DORQ] DEST[VC-BC] NORM $
01 0358 12502 010731170000060554045023654715160000           D[MEM] MASK[18.] DEST[VC-MA] NORM $
01 0359                                                       ;
01 0360                                                       VCGO:           ;Q gets 0 for 7-bit bytes, 1 for 8-bit.
01 0361                                                       .REPEAT 1 - F4SW [
01 0362                                                         ALU[-1] DEST[MAP-DISABLE] NORM $
01 0363                                                       ];.REPEAT 1 - F4SW
01 0364                                                       .REPEAT F4SW [
01 0365                                                         PUSHJ[MEMADR-23BIT-ABS] NORM $
01 0366                                                                 ;*** Shouldn't we be saving the old state?
01m0367                                                       ];[
01m0367 12503 010731170000227414161622254314560000             PUSHJ[MEMADR-23BIT-ABS] NORM $
01m0367                                                                 ;*** Shouldn't we be saving the old state?
01 0367                                                       ].REPEAT F4SW
01 0368 12504 010730170000060550003623662314160000             D[VC-STAT] ROT[32.] MASK[1] DEST[Q] NORM $
01 0369 12505 010601170000060554016443655714170000             D[CONST 7] ALU[D+Q] DEST[ROTR] NORM $ ;byte length to rotate and
01 0369
01 0370 12506 010601170000060554016503655714170000             D[CONST 7] ALU[D+Q] DEST[MASKR] NORM $ ;mask registers.
01 0371 12507 010730170000060554001623655714160000             D[CONST 0] DEST[Q] NORM $ ;Q contains last-byte flag, initially
01 0371                                                        0.
01 0372 12510 010331170000060554161143654314160000             ALU[AC] DEST[VC-AC-SAVE] NORM $ ;Save AC and PC for scratching.
01 0373                                                       .REPEAT 1 - F4SW [
01 0374                                                         D[PC] DEST[VC-PC-SAVE] NORM $
01 0375                                                         D[VC-MA] DEST[PC] NORM $ ;PC gets current MA.
01 0376                                                       ];.REPEAT 1 - F4SW
01 0377                                                       .REPEAT F4SW [
01 0378                                                         D[VC-MA] DEST[MA] NORM $ ;Set MA for transfer
01m0379                                                       ];[
01m0379 12511 710731170000060554161623660714160000             D[VC-MA] DEST[MA] NORM $ ;Set MA for transfer
01 0379                                                       ].REPEAT F4SW
01 0380 12512 010733170000060554161623661314160000             D[VC-BC] DEST[AC] NORM $ ;AC gets (negative) byte count.
01 0381 12513 010503170000060550017623661714160000             D[VC-MODE] ROT[32.] MASK[7] ALU[D+AC] DEST[AC] NORM $ ;Add burst
01 0381                                                        count,
01 0382                                                               ALU[AC] DEST[VC-BC]
01 0383 12514 010331004000000014161040254214560000                   COND[OBUS<0] JUMP[VCBRST] CYLEN[C550] $ ;check for
01 0383                                                        overflow.
01 0384 12515 011513170000060550017623661714160000             D[VC-MODE] ROT[32.] MASK[7] ALU[AC-D] DEST[AC] NORM $
01 0385                                                       ;               Ov.  AC gets bytes remaining.
01 0386 12516 010731170000000014001040255714560000           D[CONST 0] DEST[VC-BC] JUMP[VCLP] NORM $ ;Clear byte count.
01 0387
01 0388 12517 011713170000060550017623661714160000           VCBRST: D[VC-MODE] ROT[32.] MASK[7] ALU[0-D] DEST[AC] NORM $ ;No. ov.
01 0388                                                        AC+ burst ct.
01 0389
01 0390 12520 010731170000000014161622254314560000           VCLP:   PUSHJ[VCWRD] NORM $ ;Output a word.  If mode 1 or 2 and not last
01 0390                                                        byte,
01 0391                                                       ;               loop on these two instructions.
01 0392                                                               MAPF[VC-LD-DATA] D[VC-MODE] MASK[1] ALU[DORQ] COND[OBUS=0]
01 0392                                                        JUMP[VCLP]
01 0393 12521 010631002300252414003620261514560000                   LONG $
01 0394                                                       ;               go to VCDONE if last byte.
01 0395 12522 010231000000000014161620254214560000                   ALU[Q] COND[-OBUS=0] JUMP[VCDONE] CYLEN[C550] $
01 0396                                                       ;
01 0397                                                       ;               Mode 3.  Pull high 4 bits of the odd byte.
01 0398 12523 010731170000060541011503654715160000                   D[MEM] ROT[4] MASK[4] DEST[HOLD] NORM $
01 0399                                                       ;               Shift them to proper place in Q and fetch next word.
01 0400                                                       .REPEAT 1 - F4SW [
01 0401                                                               SPEC[MA+PC] D[MEM] ROT[4] DEST[MA Q] NORM $
01 0402                                                               FIXM1 $
01 0403                                                       ];.REPEAT 1 - F4SW
01 0404                                                       .REPEAT F4SW [
01 0405                                                         DFRQ D[MEM] ROT[4] DEST[Q] NORM $
01m0406                                                       ];[
01m0406 12524 010730170000060541161623654713160000             DFRQ D[MEM] ROT[4] DEST[Q] NORM $
01 0406                                                       ].REPEAT F4SW
01 0407 12525 010731170000060541161503654715160000             D[MEM] ROT[4] DEST[HOLD] NORM $ ;Low bits of odd byte to low end.
01 0407
01 0408                                                       ;               Check whether this is the last byte of the burst.
01 0409 12526 011303006000000014161620254214560000           ALU[AC+1] DEST[AC] COND[-OBUS<0] JUMP[VCLST] CYLEN[C550] $
01 0410                                                       ;               No.  Put it out.
01 0411 12527 010631170031060554011523654715160000           SPEC[IOB-OUT] D[MEM] MASK[4] ALU[DORQ] DEST[IOD] NORM $
01 0412                                                       ;               Set up byte count for second word.
01 0413                                                       .REPEAT 1 - F4SW [
01 0414                                                               MAPF[VC-LD-DATA] D[VC-MODE] ROT[35.] MASK[3] LLOAD CYLEN[IOB-OUT]
01 0414                                                        $
01 0415                                                       ];.REPEAT 1 - F4SW
01 0416                                                       .REPEAT F4SW [
01 0417                                                               MAPF[VC-LD-DATA] CYLEN[IOB-OUT] D[VC-MODE] ROT[35.] MASK[3]
01 0417                                                        $
```

```
01m0419 12S31 010731170010060540160003654314170000             D[AR] ROT[MUA-ROT] LLOAD $
01 0419                                                       ].REPEAT F4SW
01 0420                                                       ;               Clear last-byte flag, put out second word.
01 0421 12S32 010730170000000001400162225571456000           D[CONST 0] DEST[Q] PUSHJ[VCWRD1] NORM $
01 0422                                                       ;               Loop for next double word if not end of burst.
01 0423 12S33 010231002300252414161620254114560000           MAPF[VC-LD-DATA] ALU[Q] COND[OBUS=0] JUMP[VCLP] LONG $
01 0424                                                       ;               End of burst.  Save updated MA and restore AC and PC.
01 0425                                                       VCDONE:
01 0426                                                       .REPEAT F4SW [
01 0427                                                             MAPF[VC-LD-DATA] $      ;Bug:  F4 can't do A-MEM write during
                                                              IOB-OUT cycle
01 0427                                                             D[MA] DEST[VC-MA] NORM $
01 0428                                                             SPEC[IOB-OUT] D[CONST 10] DEST[IOD] $
01 0429                                                             MAPF[VC-LD-CTL] CYLEN[IOB-OUT] PUSHJ[MEMADR-18BIT] $
01 0430                                                                   ;Enable interrupt and reset map to normal
01 0431                                                                   ;*** Shouldn't we be restoring the old map state?
01 0432                                                       ];[
01m0433                                                             MAPF[VC-LD-DATA] $      ;Bug:  F4 can't do A-MEM write during
01m0433 12S34 010731170300060554161623654314160000          IOB-OUT cycle
01m0433                                                             D[MA] DEST[VC-MA] NORM $
01m0433 12S35 010731170000060554161023656714160000                 SPEC[IOB-OUT] D[CONST 10] DEST[IOD] $
01m0433 12S36 010731170031060554021523655714160000                 MAPF[VC-LD-CTL] CYLEN[IOB-OUT] PUSHJ[MEMADR-18BIT] $
01m0433 12S37 010731170400227374161622254114560000                       ;Enable interrupt and reset map to normal
01m0433                                                                   ;*** Shouldn't we be restoring the old map state?
01m0433                                                       ].REPEAT F4SW
01 0433                                                       .REPEAT 1 - F4SW [
01 0434                                                             MAPF[VC-LD-DATA] D[PC] DEST[VC-MA] NORM $
01 0435                                                             SPEC[IOB-OUT] D[CONST 10] DEST[IOD] NORM $ ;Enable interrupt
01 0436                                                             MAPF[VC-LD-CTL] D[VC-PC-SAVE] DEST[PC] CYLEN[IOB-OUT] $
01 0437                                                             ALU[0] DEST[MAP-DISABLE] NORM $
01 0438                                                       ];.REPEAT 1 - F4SW
01 0439                                                             D[VC-AC-SAVE] DEST[AC] CLR-DEV-FROM-INTR JUMP[MAIN] NORM $
01 0440 12S40 010733170040227114161620263314560000          ;               Odd byte is the last.  Put it out with the last-byte bit
01 0441                                                       on.
01 0441                                                       VCLST:  D[CONST 1] ROT[34.] ALU[DORQ] DEST Q NORM $
01 0442 12S41 010630170000060550403623655714160000                 SPEC[IOB-OUT] D[MEM] MASK[4] ALU[DORQ] DEST[IOD] JUMP[VCDONE]
01 0443 12S42 010631170031252714011520254715560000           NORM $
01 0443                                                       
01 0444                                                       
01 0445                                                       ; Subroutine to output one word.  Instruction following call must have
01 0446                                                       ; MAPF[VC-LD-DATA] CYLEN[IOB-OUT].
01 0447                                                       ;
01 0448                                                       ;               Entry for modes 1 and 2, and first word of a pair for
01 0449                                                       ;               mode 3.  Set loop count to bytes/word.
01 0450                                                       VCWRD:
01 0451                                                       .REPEAT 1 - F4SW [
01 0452                                                             SPEC[MA+PC] DEST[MA] D[VC-MODE] ROT[35.] MASK[3] LLOAD NORM $
01 0453                                                             FIXM1 $
01 0454                                                       ];.REPEAT 1 - F4SW
01 0455                                                       .REPEAT F4SW [
01 0456                                                             D[VC-MODE] ROT[35.] MASK[3] DEST[AR] NORM $
01 0457                                                             DFRQ D[AR] ROT[MUA-ROT] LLOAD $
01m0458                                                       ];[
01m0458 12S43 010731370000060550607623661714160000                 D[VC-MODE] ROT[35.] MASK[3] DEST[AR] NORM $
01m0458 12S44 010731170010060540160003654312170000                 DFRQ D[AR] ROT[MUA-ROT] LLOAD $
01 0458                                                       ].REPEAT F4SW
01 0459                                                       ;               Second entry for other word in mode 3 (it's been
01 0459                                                        fetched).
01 0460                                                       ;               Add bytes/word to bytes remaining.
01 0461                                                       VCWRD1:
01 0462                                                       .REPEAT 1 - F4SW [
01 0463                                                             SPEC[PC+1] D[VC-MODE] ROT[35.] MASK[3] ALU[D+AC] CARRY DEST[AC]
01 0464                                                                   COND[OBUS<0] JUMP[VCOUT1] CYLEN[C550] $
01 0465                                                       ;             Overflow.  Change loop count to bytes remaining.
01 0466                                                             D[VC-MODE] ROT[35.] MASK[3] ALU[D-AC] DEST[AC] LLOAD NORM $
01 0467                                                       ];.REPEAT 1 - F4SW
01 0468                                                       .REPEAT F4SW [
01 0469                                                             MA+MA+1 D[VC-MODE] ROT[35.] MASK[3] ALU[D+AC] CARRY DEST[AC]
01 0470                                                                   COND[OBUS<0] JUMP[VCOUT1] CYLEN[C550] $
01 0471                                                       ;             Overflow.  Change loop count to bytes remaining.
01 0472                                                             D[VC-MODE] ROT[35.] MASK[3] ALU[D-AC] DEST[AC AR] NORM $
01 0473                                                             D[AR] ROT[MUA-ROT] LLOAD $
01m0474                                                       ];[
01m0474                                                             MA+MA+1 D[VC-MODE] ROT[35.] MASK[3] ALU[D+AC] CARRY DEST[AC]
01m0474 12S45 111503004000000001060762026161456000                 COND[OBUS<0] JUMP[VCOUT1] CYLEN[C550] $
01m0474                                                       ;             Overflow.  Change loop count to bytes remaining.
01m0474 12S46 011523370000060550607623661714160000                 D[VC-MODE] ROT[35.] MASK[3] ALU[D-AC] DEST[AC AR] NORM $
01m0474 12S47 010731170010060540160003654314170000                 D[AR] ROT[MUA-ROT] LLOAD $
01 0474                                                       ].REPEAT F4SW
01 0475                                                       ;               Set last-byte flag in Q.
01 0476 12S50 010730170000000001040362025571456000                 D[CONST 1] ROT[34.] DEST[Q] JUMP[VCOUT1] NORM $
01 0477                                                       ;               This two-instruction loop puts out all but the last byte.
01 0477                                                       
01 0478 12S51 010731170000000014121522254515560000          VCOLP:  D[MEM] MASK[R] DEST[IOD] PUSHJ[VCODLY] LONG $
01 0479 12S52 010731070300252323216150025451556000          VCOUT1: MAPF[VC-LD-DATA] D[MEM] ROT[R] DEST[HOLD] LOOP[VCOLP] LONG $
01 0480                                                       ;               Last byte here, with the flag bit set if end of burst.
01 0481 12S53 010631170000060554121523654515160000                 D[MEM] MASK[R] ALU[DORQ] DEST[IOD] LONG $
01 0482                                                       VCODLY:
01 0483                                                       .REPEAT 1 - F4SW [
01 0484                                                             NOP NORM $              ;Let the disk get thru (was C1000)
01 0485                                                             NOP NORM $
01 0486                                                             NOP NORM $
01 0487                                                       ];.REPEAT 1 - F4SW
01 0488                                                       .REPEAT F4SW [  ;LONG isn't nearly as long!
01 0489                                                             NOP LONG $
01 0490                                                             NOP LONG $
01 0491                                                             NOP LONG $
01m0492                                                       ];[     ;LONG isn't nearly as long!
01m0492 12S54 010731170000060554161623654114160000                 NOP LONG $
01m0492 12S55 010731170000060554161623654114160000                 NOP LONG $
01m0492 12S56 010731170000060554161623654114160000                 NOP LONG $
01 0492                                                       ].REPEAT F4SW
01 0493 12S57 010731170031060554161624254114160000                 SPEC[IOB-OUT] POPJ LONG $
01 0494
```

```
01 0115                                                         ;Get the Versatec code
01 0115                                                       ]
01 0116                                                     .repeat WAITS [ ;For now, WAITS only.
01 0117                                                      .repeat 0 - (diagdev > 0) [
01 0118                                                          .insert DACDEB
01 0119                                                             ;Read/write pre-assembled device according at MAPF in AC field
01 0120
01 0121                                                      ];.repeat 0 - (diagdev > 0)
01m0122                                                     ];[     ;For now, WAITS only.
01m0122                                                      .repeat 0 - (diagdev > 0) [
01m0122                                                          .insert DACDEB
01m0122                                                             ;Read/write pre-assembled device according at MAPF in AC field
01m0122
01m0122                                                      ];[
01m0122                                                          .insert DACDEB
```

```
01m0001                                          COMMENT !   VALID 00002 PAGES
01m0001                                          C REC  PAGE   DESCRIPTION
01m0001                                          C00001 00001
01m0001                                          C00002 00002            .opcode[772]
01m0001                                          C00004 ENDMK
01m0001                                          C!;
01m0001
```

```
SLOE  March 23, 1984 21:13:30  file DSK:DACDEB.SLO  --  of -- F41NNF

02m0001                                                         .opcode[772]
02m0001                                                  [xlist
02m0002 07764 010731170000000001416162025431456000        list  ]DIAGI:        JUMP[DIAGI2] NORM $
02m0002 07765 010731170000000001416162025431456000             JUMP[DIAGI2] NORM $
02m0002                                                                 ;Too bad we can't do anything useful here...
02m0002                                                         .reloc
02m0002                                                  [.USE[HILOC]
02m0002                                                  [ xlist
02m0002                                                    list ]                                      ]
02m0003 12560 010730170000000003211622257714560000     DIAGI2: D[IR] ROT[12. + 1] MASK[4] DEST[Q] PUSHJ[DIAGSB] NORM $
02m0003                                                                 ;Set device address and setup for MAPF dispatch
02m0003 12561 010731170000000001416162225431456000             PUSHJ[DIAGI3] NORM $
02m0003                                                                 ;Go read from device
02m0003 12562 010231310000226654161561254304560000             ALU[Q] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<Z0] LBJUMP[SEQI] ]$
02m0003                                                                 ;Return data
02m0003
02m0003 12563 010731170030000000160007254314560000     DIAGI3: SPEC[IOB-IN] D[AR] ROT[MUA-ROT] QDISP $
02m0003                                                                 ;Start reading device and dispatch
02m0003
02m0003                                                         .opcode[773]
02m0003                                                  [xlist
02m0004 07766 010731170000000001416162025431256000        list  ]DIAGO:        DFRQ JUMP[DIAGO2] $
02m0004 07767 010331170000000001416150005431456000             ALU[MEMAC] DEST[HOLD] JUMP[DIAGO2] $
02m0004                                                                 ;Setup thing to store
02m0004                                                         .reloc
02m0004                                                  [.USE[HILOC]
02m0004                                                  [ xlist
02m0005                                                    list ]                                      ]
02m0005 12564 010730170000000003211622257714560000     DIAGO2: D[IR] ROT[12. + 1] MASK[4] DEST[Q] PUSHJ[DIAGSB] NORM $
02m0005                                                                 ;Set device address and setup for MAPF dispatch
02m0005 12565 010731170000000001416152225471556000             D[MEM] DEST[IOD] PUSHJ[DIAGO3] NORM $
02m0005                                                                 ;Set thing to store and dispatch
02m0005 12566 010731170000227114161620254314560000             JUMP[MAIN] $
02m0005                                                 ;       ---
02m0005
02m0005 12567 010731170031000000160007254314560000     DIAGO3: SPEC[IOB-OUT] D[AR] ROT[MUA-ROT] QDISP $
02m0005
02m0005
02m0005                                                 ;Table of MAPF's for diagnostic instruction
02m0005                                                 MAPFTB:
02m0005       0         MAPFNO = 0
02m0005                                                 .REPEAT Z0 [
02m0005                                                         MAPF[MAPFNO] D[IOD] DEST[Q] POPJ $
02m0005                                                         MAPFNO = MAPFNO + 1
02m0005                                                 ]
02m0005                                                 [
02m0005 12570 010730170000060554161624257314160000             MAPF[MAPFNO] D[IOD] DEST[Q] POPJ $
02m0005       1         MAPFNO = MAPFNO + 1
02m0005                                                 ][
02m0005 12571 010730170100060554161624257314160000             MAPF[MAPFNO] D[IOD] DEST[Q] POPJ $
02m0005       2         MAPFNO = MAPFNO + 1
02m0005                                                 ][
02m0005 12572 010730170200060554161624257314160000             MAPF[MAPFNO] D[IOD] DEST[Q] POPJ $
02m0005       3         MAPFNO = MAPFNO + 1
02m0005                                                 ][
02m0005 12573 010730170300060554161624257314160000             MAPF[MAPFNO] D[IOD] DEST[Q] POPJ $
02m0005       4         MAPFNO = MAPFNO + 1
02m0005                                                 ][
02m0005 12574 010730170400060554161624257314160000             MAPF[MAPFNO] D[IOD] DEST[Q] POPJ $
02m0005       5         MAPFNO = MAPFNO + 1
02m0005                                                 ][
02m0005 12575 010730170500060554161624257314160000             MAPF[MAPFNO] D[IOD] DEST[Q] POPJ $
02m0005       6         MAPFNO = MAPFNO + 1
02m0005                                                 ][
02m0005 12576 010730170600060554161624257314160000             MAPF[MAPFNO] D[IOD] DEST[Q] POPJ $
02m0005       7         MAPFNO = MAPFNO + 1
02m0005                                                 ][
02m0005 12577 010730170700060554161624257314160000             MAPF[MAPFNO] D[IOD] DEST[Q] POPJ $
02m0005       10                MAPFNO = MAPFNO + 1
02m0005                                                 ][
02m0005 12600 010730171000060554161624257314160000             MAPF[MAPFNO] D[IOD] DEST[Q] POPJ $
02m0005       11                MAPFNO = MAPFNO + 1
02m0005                                                 ][
02m0005 12601 010730171100060554161624257314160000             MAPF[MAPFNO] D[IOD] DEST[Q] POPJ $
02m0005       12                MAPFNO = MAPFNO + 1
02m0005                                                 ][
02m0005 12602 010730171200060554161624257314160000             MAPF[MAPFNO] D[IOD] DEST[Q] POPJ $
02m0005       13                MAPFNO = MAPFNO + 1
02m0005                                                 ][
02m0005 12603 010730171300060554161624257314160000             MAPF[MAPFNO] D[IOD] DEST[Q] POPJ $
02m0005       14                MAPFNO = MAPFNO + 1
02m0005                                                 ][
02m0005 12604 010730171400060554161624257314160000             MAPF[MAPFNO] D[IOD] DEST[Q] POPJ $
02m0005       15                MAPFNO = MAPFNO + 1
02m0005                                                 ][
02m0005 12605 010730171500060554161624257314160000             MAPF[MAPFNO] D[IOD] DEST[Q] POPJ $
02m0005       16                MAPFNO = MAPFNO + 1
02m0005                                                 ][
02m0005 12606 010730171600060554161624257314160000             MAPF[MAPFNO] D[IOD] DEST[Q] POPJ $
02m0005       17                MAPFNO = MAPFNO + 1
02m0005                                                 ][
02m0005 12607 010730171700060554161624257314160000             MAPF[MAPFNO] D[IOD] DEST[Q] POPJ $
02m0005       20                MAPFNO = MAPFNO + 1
02 0005                                                 ]
02 0006 12610 010601370000000002536162365531416000     DIAGSB: D[LABEL MAPFTB] ALU[D+Q] DEST[AR] NORM $
02 0007                                                                 ;Construct pointer into MAPF table
02 0008 12611 010731170040060554062464257571417000             D[CONST DIAGDEV] DEST[DEV-ADR] NORM POPJ $
02 0009                                                                 ;Set device number
02 0010                                                 ;       ---
02 0011
```

```
01 0122                                                 ;Read/write pre-assembled device according at MAPF in AC field
01 0123
01 0122                                                ].repeat 0 - (diagdev > 0)
01 0122                                                !.repeat WAITS
01 0123
01 0124                                                .repeat STANSW [
01 0125                                                    .insert PAN
01 0126                                                    .insert TMPGRN
01 0127                                                    .insert AUDSWX
01m0128                                                  ]
01m0128                                                [
01m0128                                                    .insert PAN
```

```
01m0001                                         ;-----------------------------------------------------------------------;
01m0001
01m0001                                         ;                                                                       ;
01m0001
01m0001                                         ;   PAN - Panofsky Interface for Systems Concept Digital Synthesizer   ;
01m0001
01m0001                                         ;                                                                       ;
01m0001
01m0001                                         ;-----------------------------------------------------------------------;
01m0001
01m0001       30        PAN-UDEV        = 30    ;uDevice address
01m0001
01m0001                                         ;MAPF fields
01m0001       1 PAN-STATUS      = 1     ;Read status register
01m0001                                         ;       Bit 0   -INT REQ 0
01m0001                                         ;       Bit 1   -INT REQ 1
01m0001                                         ;       Bit 2   MY PAR ERR
01m0001                                         ;       Bit 3   HIS PAR ERR
01m0001                                         ;       Bit 4   INT ENB 0
01m0001                                         ;       Bit 5   INT ENB 1
01m0001       2 PAN-EN-IOB      = 2     ;IOB->IOD
01m0001       0 PAN-IOB-RESET   = 0     ;Start PAN Bus Reset
01m0001       1 PAN-CONTROL     = 1     ;Set PAN device and control
01m0001       2 PAN-EN-IOB      = 2     ;IOD->IOB (same mnemonic is other direction)
01m0001       3 PAN-INT-ENB     = 3     ;Interrupt enable
01m0001
01m0001                                         ;Opcodes for CONTROL field
01m0001       27        PAN-CONTROL-ROT = 35. - 12.
01m0001       40        PAN-R/-W        = 40
01m0001       20        PAN-CON/-DAT    = 20
01m0001       10        PAN-MUNG        = 10    ;This SHOULD be two.  SET-MUNG and CLEAR-MUNG
01m0001       70        PAN-CONI-OP     = PAN-MUNG + PAN-R/-W + PAN-CON/-DAT
01m0001       30        PAN-CONO-OP     = PAN-MUNG + PAN-CON/-DAT
01m0001       50        PAN-DATAI-OP    = PAN-MUNG + PAN-R/-W
01m0001       10        PAN-DATAO-OP    = PAN-MUNG + 0
01m0001                                                 ;Use MUNGed (Pete's) format for now...  Need CONO to set change
01m0001                                          it
01m0001                                                 ;and besides, there should be SET and CLEAR of MUNG
01m0001
01m0001                                         ;A-MEM usage
01m0001                                         DEFINE-A-MEM[PAN-DISP 0]        [
01m0001       40                PAN-DISP = 0 + D%AMEM0
01m0001                                         ];PAN dispatches
01m0001                                         ;
01m0001                                         ;Also see TMPGRN.SLO
01m0001
01m0001                                                 .PAIR
01m0001                                          [:. \ 2 + .
01m0002 12612 01073100200010210160362024631456000        ]PANIOT:     D[PC-FLAGS] ROT[6 + 1] MASK[1] COND[OBUS=0] JUMP[MUUO] $
01m0002
01m0002                                                 ;Trap if User and not IOT-USER
01m0002 12613 01073117004006055406046365571417000              D[CONST PAN-UDEV] DEST[DEV-ADR] NORM $
01m0002                                                 ;Set common micro device address
01m0002                                         ;;;     D[IR] ROT[11.] MASK[7] DEST[AR] NORM $
01m0002                                         ;;;             ;Get macro device address
01m0002 12614 01077137000006054256162365771416000              d[ir] rot[9. + 1] alu[notd] dest[ar] norm $
01m0002 12615 01073137000006055401762365431416000              d[ar] mask[7] dest[ar] norm $
01m0002                                                 ;Special fudge for hardware bug.  Get device address
01m0002 12616 01073017000006054341162365771416000              D[IR] ROT[12. + 1 + 1] MASK[4] DEST[Q] NORM $
01m0002                                                 ;Extract IOT decode * 2.' Note we can do this because the
01m0002
01m0002                                                 ;machine has already done indexing/indirection and bits
01m0002                                                 ;13:17 are guaranteed zero
01m0002 12617 01060117000000000041400072601145600              D[PAN-DISP] ROT[18.] MASK[16.] ALU[D+Q] XDISP  [ ODISP LONG ]$
01m0002                                                 ;Dispatch of type of IOT
01m0002
01m0002                                                 ;Normal IOT dispatch for PAN
01m0002                                         PAN-DISPATCH:
01m0002                                         ;BLKI XXX, - Not implemented
01m0002 12620 01073117000010211416162025431456000              JUMP[MUUO] $
01m0002 12621 01073117000006055416162365431416000              NOP $
01m0002                                         ;DATAI XXX,
01m0002                                                 D[CONST PAN-DATAI-OP] ROT[PAN-CONTROL-ROT] DEST[Q]
01m0002                                          PUSHJ[PAN-IOB-READ]
01m0002 12622 01073017000000000057216222557145600                      NORM $
01m0002                                                 ;Use DATAx, In
01m0002 12623 01077131020022665416156125710456000              MAPF[PAN-EN-IOB] CYLEN[LONG] D[IOD] ALU[NOTD] SMAC  [ IFRQ
01m0002                                          DEST[MEMSTO AR] COND[MA<20] LBJUMP[SEOI] ]$
01m0002                                                 ;Finish reading data from PAN bus.  Start store and let
01m0002                                                 ;MSMAIN finish it.  (LONG is TVR being paranoid)
01m0002                                         ;BLKO XXX, - Not implemented
01m0002 12624 01073117000010211416162025431456000              JUMP[MUUO] $
01m0002 12625 01073117000006055416162365431416000              NOP $
01m0002                                         ;DATAO XXX,
01m0002 12626 01033117000006055416150345431216000              FIXM1  [ DFRQ ALU[MEMAC] DEST[HOLD] ]$
01m0002                                                 D[CONST PAN-DATAO-OP] ROT[PAN-CONTROL-ROT] DEST[Q]
01m0002                                          JUMP[PAN-IOB-WRITE]
01m0002 12627 01073017000000000056216202557145600                      NORM $
01m0002                                                 ;Use DATAx, Out
01m0002                                                 ;Send data to drive and go to MAIN
01m0002                                         ;CONO XXX,
01m0002 12630 01073117000006055404550365771416000              D[IR] MASK[22] DEST[HOLD] NORM $
01m0002                                                 ;Use immediate form, i.e. Instruction as data.
01m0002                                                 D[CONST PAN-CONO-OP] ROT[PAN-CONTROL-ROT] DEST[Q]
01m0002                                          JUMP[PAN-IOB-WRITE]
01m0002 12631 01073017000000000056616202557145600                      NORM $
01m0002                                                 ;Use CONx, Out
01m0002                                                 ;Send data to drive and go to MAIN
01m0002                                         ;CONI XXX,
01m0002                                                 D[CONST PAN-CONI-OP] ROT[PAN-CONTROL-ROT] DEST[Q]
01m0002                                          PUSHJ[PAN-IOB-READ]
01m0002 12632 01073017000000000057616222557145600                      NORM $
01m0002                                                 ;Use CONx, In
01m0002 12633 01077131020022665416156125710456000              MAPF[PAN-EN-IOB] CYLEN[LONG] D[IOD] ALU[NOTD] SMAC  [ IFRQ
01m0002                                          DEST[MEMSTO AR] COND[MA<20] LBJUMP[SEOI] ]$
01m0002                                                 ;Finish reading data from PAN bus.  Start store and let
01m0002                                                 ;MSMAIN finish it.  (LONG is TVR being paranoid)
01m0002                                         ;CONSZ XXX,
01m0002                                                 D[CONST PAN-CONI-OP] ROT[PAN-CONTROL-ROT] DEST[Q]
01m0002                                          PUSHJ[PAN-IOB-READ]
```

```
01m0002                                                         ;CONSZ. (LONG is TVR being paranoid)
01m0002                                                 ;CONSO XXX,
01m0002                                                         D[CONST PAN-CONI-OP] ROT[PAN-CONTROL-ROT] DEST[Q]
01m0002                                                  PUSHJ[PAN-IOB-READ]
01m0002 12636 010730170000000005761622255714560000                             NORM $
01m0002                                                         ;Use CONx, In
01m0002 12637 010770170200000014161620257114560000             MAPF[PAN-EN-IOB] CYLEN[LONG] D[IOD] ALU[NOTD] DEST[Q]
01m0002                                                  JUMP[XXCONSO] $
01m0002                                                         ;Finish reading data from PAN and go do generalized
01m0002                                                         ;CONSO. (LONG is TVR being paranoid)
01m0002
01m0002                                                 ;Read from PAN's IOB.  Return to caller to finish read.
01m0002                                                 PAN-IOB-READ:
01m0002 12640 010631170031060547361523654314160000             D[AR] ROT[36. - 7] ALU[DORQ] DEST[IOD] SPEC[IOB-OUT] NORM $
01m0002                                                         ;Set device selection and mode
01m0002 12641 010731170100060554161623654114160000             MAPF[PAN-CONTROL] CYLEN[IOB-OUT] $
01m0002                                                         ;Finish setting mode and device
01m0002                                                 .REPEAT F4SW [
01m0002                                                 ;;;;;; What is settling time supposed to be for PAN interface?
01m0002                                                         MAPF[PAN-CONTROL] C1000 $
01m0002                                                         ;C1000 lies on the F4.  Give more settling time.
01m0002                                                 ];[
01m0002                                                 ;;;;;; What is settling time supposed to be for PAN interface?
?ERROR - set location with : must be followed by a valid expression
01m0002 12642 010731170100060554161623654114160000             MAPF[PAN-CONTROL] C1000 $
01m0002                                                         ;C1000 lies on the F4.  Give more settling time.
01 0002                                                 ].REPEAT F4SW
01 0003 12643 010731170030060554161624254114160000             SPEC[IOB-IN] POPJ C1000 $
01 0004                                                         ;Allow 1 usec settling time for device address on PAN bus
01 0004
01 0005                                                         ;Start reading from device
01 0006
01 0007                                                 ;Write to PAN's IOB.  Jump to MAIN on completion
01 0008                                                 PAN-IOB-WRITE:
01 0009                                                 .REPEAT F4SW [
01 0010                                                         D[MEM] NORM $
01 0011                                                         ;Take map trap cycles here (*SIGH*)
01m0012                                                 ];[
01m0012 12644 010731170000060554161623654715160000             D[MEM] NORM $
01m0012                                                         ;Take map trap cycles here (*SIGH*)
01 0012                                                 ].REPEAT F4SW
01 0013 12645 010631170031060547361523654314160000             D[AR] ROT[36. - 7] ALU[DORQ] DEST[IOD] SPEC[IOB-OUT] NORM $
01 0014                                                         ;Set device selection and mode
01 0015 12646 010731170131060554161523654515160000             MAPF[PAN-CONTROL] CYLEN[IOB-OUT] D[MEM] DEST[IOD] SPEC[IOB-OUT] $
01 0015
01 0016                                                         ;Finish setting mode.  Start write to device
01 0017 12647 010731170200227114161620254114560000             MAPF[PAN-EN-IOB] CYLEN[IOB-OUT] FETCH-NEXT-INST  [ JUMP[MAIN] ]$
01 0017
01 0018                                                         ;Finish writing data.  Start next instruction
01 0019
01 0020                                                 ;General CONSZ/CONSO termination
01 0021                                                 XXCONSO:
01 0022                                                 .REPEAT 1 - F4SW [
01 0023                                                         D[IR] MASK[ZZ] ALU[D&Q] COND[OBUS=0] JUMP[MAIN] C550 $
01 0024                                                         DOSKIP $
01 0025                                                 ];.REPEAT 1 - F4SW
01 0026                                                 .REPEAT F4SW [
01 0027                                                         D[IR] MASK[ZZ] ALU[D&Q] CONDSKP[-ALU=0] $
01m0028                                                 ];[
01m0028 12650 540641002000073554045623257604160000             D[IR] MASK[ZZ] ALU[D&Q] CONDSKP[-ALU=0] $
01 0028                                                 ].REPEAT F4SW
01 0029
01 0030                                                 XXCONSZ:
01 0031                                                 .REPEAT 1 - F4SW [
01 0032                                                         D[IR] MASK[ZZ] ALU[D&Q] COND[-OBUS=0] JUMP[MAIN] C550 $
01 0033                                                         DOSKIP $
01 0034                                                 ];.REPEAT 1 - F4SW
01 0035                                                 .REPEAT F4SW [
01 0036                                                         D[IR] MASK[ZZ] ALU[D&Q] CONDSKP[ALU=0] $
01m0037                                                 ];[
01m0037 12651 540641000000073554045623257604160000             D[IR] MASK[ZZ] ALU[D&Q] CONDSKP[ALU=0] $
01 0037                                                 ].REPEAT F4SW
01 0038
01 0039                                                 ;-----------------------------------------------------------------------
01 0039                                                 ------
01 0040                                                 ;       PAN Interrupts
01 0041                                                 ;
01 0042                                                 ;The PAN microinterrupt enable is turned off when the interrupt for that
01 0042
01 0043                                                 ;channel is recieved and turned on by PI-CHECK-RQS by setting all of the
01 0043
01 0044                                                 ;microinterrupt enables for the PAN to the complement of PI IN PROGRESS
01 0044                                                  and
01 0045                                                 ;PI REQUEST.  This means that the PAN can only request interrupts
01 0045                                                  whenever
01 0046                                                 ;something else isn't using those channels.  This implies that any PAN
01 0047                                                 ;devices must be on the end of CONSZ chains, or else they will steal
01 0048                                                 ;interrupts from other, interrupt counting devices.
01 0049                                                 ;-----------------------------------------------------------------------
01 0049                                                 ------
01 0050                                                 PAN-INTR:
01 0051 12652 010731170030060554161623654314160000             SPEC[IOB-IN] NORM $
01 0052                                                         ;Read status from device
01 0053                                                         ;Display PI-WAITING,,IN-PROGRESS on OBUS
01 0054                                                 MAPF[PAN-STATUS] CYLEN[IOB-IN]
01 0055 12653 010770370100060540561503657114160000                             D[IOD] ROT[1 + 1] ALU[NOTD] DEST[AR Q HOLD] $
01 0056                                                         ;Get enabled and interrupting channels
01 0057                                                         ;*** Stuff into HOLD for debugging.
01 0058 12654 010650370000060541161623654314160000             D[AR] ROT[5 - 1.] ALU[-D&Q] DEST[AR Q] NORM $
01 0059                                                         ;And interrupting and enabled channels.  Still have
01 0059                                                  garbage
01 0060                                                         ;in other bits
01 0061 12655 010731202000000010603620254214560000             D[AR] ROT[36. - 1] MASK[1] DEST[AR] COND[OBUS=0] JUMP[PANNC1]
01 0061                                                  C550 $
01 0062                                                         ;Jump if not channel 1.
01 0063                                                         ;If it is channel 1, set PI channel in AR
01 0064 12656 010770170000000007603620255714560000             D[CONST 1] ROT[35. - 4.] ALU[NOTD] DEST[Q] JUMP[PANIDS] NORM $
01 0065                                                         ;Turn off channel 1 and interrupt
01 0066 12657 010641002000000014003620255614560000     PANNC1: D[CONST 1] ALU[D&Q] COND[OBUS=0] JUMP[GRNINT] C550 $
01 0067                                                         ;Jump if not channel 2.
```

```
01 0073                                         ;Micro-interrupt disable.  Used to disable further micro-interrupts until
01 0073
01 0074                                         ;macro-interrupt is taken for that channel.
01 0075                                         PANIDS: D[AMEM-ABS PI-STATUS] ROT[35. - 10.] ALU[D&Q] DEST[Q]
700 01 0076 12662 01064017000000001063616236703141600    .REPEAT F4SW [ $ ]    [ $ ]   ;Watch field conflict on F4
01 0077 12663 01073117000000001416162225431456000                                PUSHJ[PANIST] NORM $
01 0078                                                             ;Remove interrupting channel from channels ready for
01 0078                                          interrupts
01 0079                                                             ;and set the hardware register to that value.
01 0080 12664 01073117030020343416162025411456000                   MAPF[PAN-INT-ENB] CYLEN[IOB-OUT] JUMP[PIGEN] $
01 0081                                                             ;Finish setting hardware interrupt enables and take
01 0082                                                             ;macro-interrupt
01 0083
01 0084                                         ;Compute micro-interrupt enabling from state of firmware PI system, and
01 0084                                          set
01 0085                                         ;hardware register in PAN.  Called with Q containing PI channels that are
01 0085                                          on.
01 0086                                         ;Clobbers dev-address and must be followed by MAPF[PAN-INT-ENB]
3 01 0087 12665 01065017000000001263616236703741600    PANIST: D[AMEM-ABS PI-IN-PROGRESS] ROT[35. - 10.] ALU[-D&Q] DEST[Q] SHORT
01 0087                                          $
01 0088                                                             ;Remove channels in progress
01 0089 12666 01065017000000001217615236703741600          D[AMEM-ABS PI-IN-PROGRESS] ROT[17. - 10.] ALU[-D&Q] DEST[Q IOD]
01 0089                                          SHORT $
01 0090                                                             ;Remove channels which are already waiting.
01 0091 12667 01073117003106055406046425571417000          D[CONST PAN-UDEV] DEST[DEV-ADDRESS] SPEC[IOB-OUT] POPJ NORM $
01 0092                                                             ;Select PAN interface and turn its micro interrupts on.
01 0093                                                             ;There is no way in Hell you can count interrupts a PAN
01 0093                                          IO bus,
01 0094                                                             ;or even know when an interrupt condition has been
01 0094                                          satisfied.
01 0095
01 0096                                         ;Reset PAN, Set interrupt and dispatch addresses
01 0097 12670 01073117003125563406046025571457000      PANRST: D[CONST PAN-UDEV] DEST[DEV-ADDRESS] SPEC[IOB-OUT] NORM JUMP[. +
01 0097                                          1] $
01 0098                                                             ;Start IOB RESET happening.
01 0099                                                 MAPF[PAN-IOB-RESET] CYLEN[IOB-OUT]
01 0100                                         .REPEAT 1 - F4SW [
01 0101                                                                     D[CONST (PAN-DISPATCH / 100)] ROT[30] DEST[Q] $
01 0102                                                             ;Construct dispatch address: high 6 bits.  Finish IOB
01 0102                                          RESET
01 0103                                                 D[CONST (PAN-DISPATCH \ 100)] ROT[22] ALU[DORQ] DEST[Q] NORM $
01 0104                                                             ;Low 6 bits
01 0105                                                 D[CONST (PAN-INTR / 100)] ROT[6] ALU[DORQ] DEST[Q] NORM $
01 0106                                                             ;Construct interrupt address: high 6 bits
01 0107                                                 D[CONST (PAN-INTR \ 100)] ALU[DORQ] DEST[PAN-DISP] POPJ NORM$
01 0108                                                             ;Finish contructing interrupt address and store away.
01 0109                                                             ;We're done.
01 0110
01 0111                                         ];.REPEAT 1 - F4SW
01 0112                                         .REPEAT F4SW [
01 0113                                                 D[LIT ((PAN-DISPATCH * 1000000) + PAN-INTR)] DEST[PAN-DISP] POPJ
01 0113                                          $
01m0114                                         ];[
01m0114 12671 01073117002544002552500425511416000      D[LIT ((PAN-DISPATCH * 1000000) + PAN-INTR)] DEST[PAN-DISP] POPJ
01m0114                                          $
01 0114                                         ].REPEAT F4SW
```

01 0128 .insert TMPGRN

```
01 0001                                         ;-----------------------------------------------------------------------;
01 0001
01 0002                                         ;                                                                       ;
01 0002
01 0003                                         ;   GRN - Temporary Grinnel Interface                                   ;
01 0003
01 0004                                         ;                                                                       ;
01 0004
01 0005                                         ;-----------------------------------------------------------------------;
01 0005
01 0006       30        GRN-UDEV        = PAN-UDEV      ;uDevice address
01 0007
01 0008                                         ;MAPF fields
01 0009       5 GRN-STATUS      = 5     ;Read status register
01 0010                                                 ;200    G INT RQ        (= G INT ENB  G INT REQ)
01 0011                                                 ;100    FIFO BSY        FIFO input register ready
01 0012                                                 ;40     GWBSY           Grinnell is busy
01 0013                                                 ;20     G INT REQ       FIFO seems empty
01 0014                                                 ;10     G INT ENB       Micro interrupt enable
01 0015       5 GRN-CONTROL     = 5     ;Set control
01 0016                                                 ;10     G INT ENB       Micro interrupt enable
01 0017       6 GRN-DATA        = 6     ;Send Data
01 0018
01 0019                                         ;A-MEM usage
01m0020                                         DEFINE-A-MEM[GRN-DISP 4]        [
01m0020       44                GRN-DISP = 4 + D%AMEM0
01 0020                                         ];IOT dispatch in LH, RH unused
01m0021                                         DEFINE-A-MEM[GRN-PICHN 5]       [
01m0021       45                GRN-PICHN = 5 + D%AMEM0
01 0021                                         ];PI channel for Grinnell
01 0022
01 0023                                         .REPEAT F4SW [
01 0024                                                 .OPCODE[774]
01 0025                                                 D[CONST GRN-UDEV] DEST[DEV-ADR] NORM COND[-USER] LBJUMP[GRNIOT] $
01 0025
01 0026                                                 D[CONST GRN-UDEV] DEST[DEV-ADR] NORM COND[-USER] LBJUMP[GRNIOT] $
01 0026
01 0027                                                         ;Set micro device address for Grinnell and check for
01 0028                                                         ;IOT-User mode
01m0029                                         ];[
01m0029                                                 .OPCODE[774]
01m0029                                         [xlist
01m0030 07770 010731126040000014060461255714570000       list ]      D[CONST GRN-UDEV] DEST[DEV-ADR] NORM COND[-USER]
01m0030                                         LBJUMP[GRNIOT] $
01m0030 07771 010731126040000014060461255714570000             D[CONST GRN-UDEV] DEST[DEV-ADR] NORM COND[-USER] LBJUMP[GRNIOT] $
01m0030
01m0030                                                         ;Set micro device address for Grinnell and check for
01m0030                                                         ;IOT-User mode
01 0029                                         ].REPEAT F4SW
01 0030
01m0031                                                 .RELOC
01m0031                                         [.USE[HILOC]
01m0031                                         [ xlist
01 0032                                           list ]                                ]
01m0033                                                 .PAIR
01m0033                                         [:. \ 2 + .
01 0034 12672 010731002000102101603620246114560000       ]GRNIOT:     D[PC-FLAGS] ROT[6 + 1] MASK[1] COND[OBUS=0] JUMP[MUUO]
01 0034                                         LONG $
01 0035                                                         ;Trap if User and not IOT-USER
01 0036 12673 010730170000060543411623657714160000             D[IR] ROT[12. + 1 + 1] MASK[4] DEST[Q] NORM $
01 0037                                                         ;Extract IOT decode * 2.  Note we can do this because the
01 0038
01 0039                                                         ;machine has already done indexing/indirection and bits
01 0040 12674 010601170000000000444000726211456000             ;13:17 are guaranteed zero
01 0041                                                 D[GRN-DISP] ROT[18.] MASK[16.] ALU[D+Q] XDISP  [ ODISP LONG ]$
01 0042
01 0043                                                         ;Dispatch of type of IOT
01 0044                                                 ;Normal IOT dispatch for Grinnell
01 0045                                         GRN-DISPATCH:
01 0046 12675 010731170000102114161620254314560000     ;BLKI XXX, - Not implemented
01 0047 12676 010731170000060554161623654314160000             JUMP[MUUO] $
01 0048                                                 NOP $
01 0049 12677 010731170000102114161620254314560000     ;DATAI XXX,
01 0050 12700 010731170000060554161623654314160000             JUMP[MUUO] $
01 0051                                                 NOP $
01 0052 12701 010731170000102114161620254314560000     ;BLKO XXX, - Not implemented
01 0053 12702 010731170000060554161623654314160000             JUMP[MUUO] $
01 0054                                                 NOP $
01 0055 12703 010331170000060554161503454314216000     ;DATAO XXX,
01 0056 12704 010731170003100001416152025471556000             FIXM1  [ DFRQ ALU[MEMAC] DEST[HOLD] ]$
01 0057                                                 D[MEM] DEST[IOD] SPEC[IOB-OUT] JUMP[GRNDTO] $
01 0058                                                         ;Send data blindly to Grinnell
01 0059                                                         ;(Continued below, two micro(n$tructions' worth)
01 0060 12705 010731170031060554161523657714160000     ;CONO XXX,
01 0061                                                 D[IR] DEST[IOD] SPEC[IOB-OUT] NORM $
01 0061                                                         ;Remember PI channel and start setting micro interrupt
01 0062                                                 enable
01 0063                                                         ;bit.
01 0064 12706 010731170500227114011120257514560000             MAPF[GRN-CONTROL] CYLEN[IOB-OUT]
01 0065                                                                 D[IR] MASK[4] DEST[GRN-PICHN] JUMP[MAIN] $
01 0065                                                         ;Finish setting micro-interrupt enable bit and also save
01 0066                                                 it
01 0067                                                         ;away
01 0068 12707 010731170030000014161622254314560000     ;CONI XXX,
01 0069                                                 SPEC[IOB-IN] PUSHJ[GRNSTS] NORM $
01 0070 12710 010231310000226654161561254304560000             ;Start reading status
01 0070                                                 ALU[Q] DEST[MEMSTO] MEMST  [ SMAC  [ IFRQ DEST[MEMSTO AR]
01 0071                                                 COND[MA<20] LBJUMP[SEOI] ]]$
01 0072                                                         ;Finish reading status from Grinell interface.
01 0073                                                         ;Start store and let MSMAIN finish it.
01 0074 12711 010731170030000014161622254314560000     ;CONSZ XXX,
01 0075                                                 SPEC[IO3-IN] PUSHJ[GRNSTS] NORM $
01 0076 12712 010731170000255234161620254314560000             ;Start reading status
01 0077                                                 JUMP[XXCONSZ] NORM $
01 0078                                                         ;Go do generalized CONSZ.
01 0079 12713 010731170030000014161622254314560000     ;CONSO XXX,
01 0080                                                 SPEC[IOB-IN] PUSHJ[GRNSTS] NORM $
01 0081 12714 010731170000255214161620254314560000             ;Start reading status
01 0082                                                 JUMP[XXCONSO] NORM $
                                                                ;Go do generalized CONSO.
```

```
01 0088                                                         ;Remove PI channel and interrupt enable
01 0089 12717 010630170000060554161624262714160000             D[GRN-PICHN] ALU[DORQ] DEST[Q] POPJ NORM $
01 0090                                                         ;Put in PI channel and firmware interrupt enable
01 0091
01 0092                                                 ;Finish writing to Grinnell and start next instruction.
01 0093 12720 010731170600227114161620254114560000     GRNDTO: MAPF[GRN-DATA] CYLEN[IOB-OUT] FETCH-NEXT-INST  [ JUMP[MAIN] ]$
01 0094
01 0095                                                 ;Grinnell interrupt. Pass to system, if enabled.  Otherwise, ignore
01 0096 12721 010731170030060554161623654374160000     GRNINT: SPEC[IOB-IN] SHORT $
01 0097                                                         ;Fetch status
01 0098 12722 010730170500060547161623657114160000             MAPF[GRN-STATUS] CYLEN[IOB-IN] D[IOD] ROT[35. - 7] DEST[Q] $
01 0099                                                         ;Get interrupt request bit
01 0100 12723 010231002000256474161620254314560000             ALU[Q] COND[OBUS=0] JUMP[.] NORM $
01 0101                                                         ;Interrupt without a cause.
01 0102 12724 010241170031060554161523654374160000             ALU[0] DEST[IOD] SPEC[IOB-OUT] SHORT $
01 0103                                                         ;Clear micro-interrupt enable
01 0104                                                         MAPF[GRN-CONTROL] CYLEN[IOB-OUT]
01 0105 12725 010731200500203434007620262514560000               D[GRN-PICHN] MASK[3] DEST[AR] COND[-OBUS=0] JUMP[PIGEN] $
01 0105
01 0106                                                         ;Take interrupt, if enabled.
01 0107 12726 010731170041022711416162025431456000             CLR-DEV-FROM-INTR FETCH-NEXT-INST  [ JUMP[MAIN] ]$
01 0108                                                         ;Dismiss interrupt
01 0109
01 0110                                                 ;Reset Grinnell, Set dispatch addresses
01 0111 12727 010241170000060554161123654374160000     GRNRST: ALU[0] DEST[GRN-PICHN] SHORT $
01 0112                                                         ;No PI channel yet.
01 0113                                                 .REPEAT 1 - F4SW [
01 0114                                                         D[CONST (GRN-DISPATCH / 100)] ROT[30] DEST[Q] NORM $
01 0115                                                         ;Construct dispatch address: high 6 bits.  Finish IOB
01 0115                                                  RESET
01 0116                                                         D[CONST (GRN-DISPATCH \ 100)] ROT[22] ALU[DORQ]
01 0117                                                                                       DEST[GRN-DISP] NORM POPJ $
01 0118                                                         ;Finish contructing dispatch address and store away.
01 0119                                                         ;We're done.
01 0120                                                 ];.REPEAT 1 - F4SW
01 0121                                                 .REPEAT F4SW [
01 0122                                                         D[LIT ((GRN-DISPATCH * 1000000) + GRNINT)] DEST[GRN-DISP] NORM
01 0122                                                  POPJ $
01 0123                                                         ;Contruct dispatch address and store away.
01 0124                                                         ;Note:  Right half not used.
01 0125                                                         ;We're done.
01m0126                                                 ];[
01m0126 12730 010731170025572025643104255314160000             D[LIT ((GRN-DISPATCH * 1000000) + GRNINT)] DEST[GRN-DISP] NORM
01m0126                                                  POPJ $
01m0126                                                         ;Contruct dispatch address and store away.
01m0126                                                         ;Note:  Right half not used.
01m0126                                                         ;We're done.
01 0126                                                 ].REPEAT F4SW
```

01 0120 .insert AUDSWX

```
01 0001                                                 ;-----------------------------------------------------------------------;
01 0001
01 0002                                                 ;                                                                       ;
01 0002
01 0003                                                 ;   AUDSW - Audio Switch Microcode                                     ;
01 0003
01 0004                                                 ;                                                                       ;
01 0004
01 0005                                                 ;-----------------------------------------------------------------------;
01 0005
01 0006         30      AS-UDEV = PAN-UDEV      ;uDevice address
01 0007
01 0008                                                 ;MAPF fields
01 0009         4 AS-DATA = 4       ;MAPF field
01 0010                                                 ;       Bit 0           Strobe
01 0011                                                 ;       Bit 1           Set/Clear (set=1)
01 0012                                                 ;       Bit 2           Reset
01 0013                                                 ;       Bits 12:17      Output channel
01 0014                                                 ;       Bits 30:35      Input channel
01 0015
01 0016                                                 ;A-MEM usage (defined in PAN.SLO)
01m0017                                                 DEFINE-A-MEM[ASDSP 6]   [
01m0017         46              ASDSP = 6 + D%AMEM0
01 0017                                                 ];IOT dispatch in LH
01m0018                                                 DEFINE-A-MEM[ASBASE 6]  [
01m0018         45              ASBASE = 6 + D%AMEM0
01 0018                                                 ];Address of table (RH)
01m0019                                                 DEFINE-A-MEM[ASTEMP 7]  [
01m0019         47              ASTEMP = 7 + D%AMEM0
01 0019                                                 ];Address of table
01 0020
01m0021                                                         .PAIR
01m0021                                                  [:. \ 2 + .
01 0022 12732 010731002000102101603620256314560000       ]ASIOT:       D[PC] ROT[6 + 1] MASK[1] COND[DBUS=0] JUMP[MUUO] $
01 0023                                                                 ;Trap if User and not IOT-USER
01 0024 12733 010730170000060543411623657714160000             D[IR] ROT[12. + 1 + 1] MASK[4] DEST[Q] NORM $
01 0025                                                                 ;Extract IOT decode * 2.  Note we can do this because the
01 0025
01 0026                                                                 ;machine has already done indexing/indirection and bits
01 0027                                                                 ;13:17 are guaranteed zero
01 0028 12734 010601170000000000444000726311456000             D[ASDSP] ROT[18.] MASK[16.] ALU[D+Q] XDISP  [ ODISP LONG ]$
01 0029                                                                 ;Dispatch of type of IOT
01 0030
01 0031                                                         ;Normal IOT dispatch for Grinnell
01 0032                                                 AS-DISPATCH:
01 0033                                                 ;BLKI XXX, - Not implemented
01 0034 12735 010731170000102114161620254314560000             JUMP[MUUO] $
01 0035 12736 010731170000060554161623654314160000             NOP $
01 0036                                                 ;DATAI XXX,
01 0037 12737 010731170000102114161620254314560000             JUMP[MUUO] $
01 0038 12740 010731170000060554161623654314160000             NOP $
01 0039                                                 ;BLKO XXX, - Not implemented
01 0040                                                 ;;      JUMP[MUUO] $
01 0041                                                 ;;      NOP $
01 0042 12741 010331170000060554161503454312160000             fixm1  [ DFRQ ALU[MEMAC] DEST[HOLD] ]$
01 0043 12742 010731170031000014161520254715560000             d[mem] dest[iod] spec[iob-out] jump[asblko] norm $
01 0044                                                 ;DATAO XXX,
01 0045 12743 010331170000060554161503454312160000             FIXM1  [ DFRQ ALU[MEMAC] DEST[HOLD] ]$
01 0046 12744 010731370000000014161620254715560000             D[MEM] DEST[AR] JUMP[ASDTO] $
01 0047                                                                 ;Fetch device
01 0048                                                 ;CONO XXX,
01 0049 12745 010730170000060554141623663314160000             D[ASBASE] LEFT DEST[Q] NORM $
01 0050 12746 010631170000227114045140257714560000             D[IR] MASK[18.] ALU[DORQ] DEST[ASBASE] JUMP[MAIN] $
01 0051
01 0052                                                 ;CONI XXX,
01 0053 12747 010241310000226654161561254304560000             ALU[0] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<20] LBJUMP[SEOI] ]$
01 0054 12750 010731170000060554161623654314160000             NOP $
01 0055                                                                 ;Fake 0 data, store and let MSMAIN finish it.
01 0056                                                 ;CONSZ XXX,
01 0057 12751 010240170000060554161623654314160000             ALU[0] DEST[Q] NORM $
01 0058                                                                 ;Fake CONI data
01 0059 12752 010731170000255234161620254314560000             JUMP[XXCONSZ] NORM $
01 0060                                                                 ;Go do generalized CONSZ.
01 0061                                                 ;CONSO XXX,
01 0062 12753 010240170000060554161623654314160000             ALU[0] DEST[Q] NORM $
01 0063                                                                 ;Fake CONI data
01 0064 12754 010731170000255214161620254314560000             JUMP[XXCONSO] NORM $
01 0065                                                                 ;Go do generalized CONSO.
01 0066
01 0067                                                 ;Finish BLKO
01 0068 12755 010731170400227114161620254114560000     ASBLKO: MAPF[AS-DATA] CYLEN[IOB-OUT] FETCH-NEXT-INST  [ JUMP[MAIN] ]$
01 0069
01 0070                                                 ;Finish DATAO
01 0071                                                 ;CAUTION:  This code can get map fill cycles, since the map is left on.
01 0071                                                  However,
01 0072                                                 ;          repeating an operation isn't harmful.
01 0073 12756 010730002000227114045620263214560000     ASDTO:  D[ASBASE] MASK[18.] DEST[Q] COND[OBUS=0] JUMP[MAIN] C550 $
01 0074                                                                 ;Make sure we're setup properly
01 0075 12757 610731170000060554161403654715160000             D[MEM] DEST[IR-ALL] NORM $
01 0076                                                                 ;Use entire IR to save new mapping
01 0077 12760 710601170000060544415623657714160000             D[IR] ROT[18.] MASK[6] ALU[D+Q] DEST[MA] NORM $
01 0078                                                                 ;Calculate table entry
01 0079 12761 010331170000060554161503454312160000             FIXM1  [ DFRQ ALU[MEMAC] DEST[HOLD] ]$
01 0080                                                                 ;Contemplate bizarre case of AC, take map traps
01 0081                                                 ;IR - new mapping
01 0082                                                 ;MEM - old mapping
01 0083 12762 010600170000000000752162225571456000             D[CONST 50] ROT[30.] ALU[D+Q] DEST[Q] PUSHJ[ASMAP] NORM $
01 0084                                                                 ;Clear mode
01 0085 12763 010730170000060554045623663314160000             D[ASBASE] MASK[18.] DEST[Q] NORM $
01 0086                                                                 ;Setup again.
01 0087 12764 710601170000060544415623657714160000             D[IR] ROT[18.] MASK[6] ALU[D+Q] DEST[MA] NORM $
01 0088                                                                 ;Calculate table entry
01 0089                                                 .repeat f4sw [
01 0090                                                 ;;;;;; Here's a JUMP[.] to fix.
01m0091                                                 ];[
01m0091                                                 ;;;;;; Here's a JUMP[.] to fix.
?ERROR - set location with : must be followed by a valid expression
01 0091                                                 ].repeat f4sw
01 0092 12765 010731110000257534161560257714560000             D[IR] DEST[MEMSTO] COND[MA<20] JUMP[.]$
01 0093                                                                 ;Now, store in memory, and check for AC reference.
```

```
SLOE  March 23, 1984 21:14:19  file DSK:AUDSWX.SLO  --  of -- F41NNF

01 0098 12767 010731170000ZZ7114161620Z5431456000              FETCH-NEXT-INST  [ JUMP[MAIN] ]$
01 0099
01 0100                                                       ;Do software mapping of audio
01 0101                                                       ;MEM - destination,,source
01 0102 12770 010600170000060541403623655714160000           ASMAP:  D[CONST 1] ROT[6] ALU[D+Q] DEST[Q] NORM $
01 0103                                                                       ;Point to mapping table
01 0104 12771 010601170000060554045163654715160000                   D[MEM] MASK[18.] ALU[D+Q] DEST[ASTEMP] NORM $
01 0105                                                                       ;Setup to fetch source mapping
01 0106 12772 010601370000000004445623254715560000                   D[MEM] ROT[18.] MASK[18.] ALU[D+Q] DEST[AR] PUSHJ[ASMAPS] NORM $
01 0106
01 0107                                                                       ;Setup to fetch destination mapping
01 0108                                                                       ;Go off and fetch both.
01 0109 12773 010731370000060544561623654314160000                   D[AR] ROT[18.] DEST[AR] NORM $
01 0110                                                                       ;Destination mapping is in the left half of the table.
01 0111                                                       ;       \ /
01 0112 12774 010640170000000007561622255714560000           ASMAP1: D[CONST 70] ROT[30.] ALU[D&Q] DEST[Q] PUSHJ[ASMAPS] NORM $
01 0113                                                                       ;Just some mode bits, please
01 0114                                                                       ;Get next pair
01 0115 12775 010630170000060554141623654314160000                   D[AR] LEFT ALU[DORQ] DEST[Q] NORM $
01 0116                                                                       ;Mapped destination
01 0117 12776 010630170031060544445523663714160000                   D[ASTEMP] ROT[18.] MASK[18.] ALU[DORQ] DEST[Q IOD] SPEC[IOB-OUT]
 NORM $
01 0118                                                                       ;Mapped source, leading edge
01 0119                                                                       MAPF[AS-DATA] CYLEN[IOB-OUT]
01 0120 12777 010651170431060547501523655514160000                                   D[CONST 40] ROT[30.] ALU[-D&Q] DEST[IOD]
 SPEC[IOB-OUT] $
01 0121                                                                       ;Leading edge
01 0122                                                                       MAPF[AS-DATA] CYLEN[IOB-OUT]
01 0123 13000 010631170431060547501523655514160000                                   D[CONST 40] ROT[30.] ALU[DORQ] DEST[IOD]
 SPEC[IOB-OUT] $
01 0124                                                                       ;Trailing edge
01 0125 13001 010731170400257714161620Z5411456000                    MAPF[AS-DATA] CYLEN[IOB-OUT] JUMP[ASMAP1] $
01 0126                                                       ;       --
01 0127
01 0128                                                       ;Advance to next entry in list.
01 0129                                                       ASMAPS:
01 0130                                                       .REPEAT 1 - F4SW [
01 0131                                                                       D[AR] DEST[MA] COND[OBUS=0] JPOP[ASMAP3] C550 $
01 0132                                                                       ;Fetch a destination
01 0133                                                       ];.REPEAT 1 - F4SW
01 0134                                                       .REPEAT F4SW [
01 0135                                                                       D[AR] DEST[MA] COND[OBUS=0] JUMP[ASMAP4] C550 $
01 0136                                                                       ;Fetch a destination
01m0137                                                       ];[
01m0137 13002 710731002000000041616Z02542145600O              D[AR] DEST[MA] COND[OBUS=0] JUMP[ASMAP4] C550 $
01m0137                                                                      ;Fetch a destination
01 0137                                                       ].REPEAT F4SW
01 0138 13003 010331170000060554161503454312160000                   FIXM1  [ DFRQ ALU[MEMAC] DEST[HOLD] ]$
01 0139 13004 010731370000060554161623654715160000                   D[MEM] DEST[AR] NORM $
01 0140                                                                       ;Save destination mapping
01 0141                                                       .REPEAT 1 - F4SW [
01 0142                                                                       D[ASTEMP] MASK[18.] DEST[MA] COND[GBUS=0] JPOP[ASMAP3] NORM $
01 0143                                                                       ;Fetch a source
01 0144                                                       ];.REPEAT 1 - F4SW
01 0145                                                       .REPEAT F4SW [
01 0146                                                                       D[ASTEMP] MASK[18.] DEST[MA] COND[OBUS=0] JUMP[ASMAP4] NORM $
01 0147                                                                       ;Fetch a source
01m0148                                                       ];[
01m0148 13005 710731002000000014045620263714560000                   D[ASTEMP] MASK[18.] DEST[MA] COND[OBUS=0] JUMP[ASMAP4] NORM $
01m0148                                                                      ;Fetch a source
01 0148                                                       ].REPEAT F4SW
01 0149 13006 010331170000060554161503454312160000                   FIXM1  [ DFRQ ALU[MEMAC] DEST[HOLD] ]$
01 0150 13007 010731170000060554161164254615160000           ASMAP3: D[MEM] DEST[ASTEMP] POPJ C550 $
01 0151                                                                       ;Save source mapping
01 0152
01 0153                                                       .REPEAT F4SW [
01 0154                                                       ASMAP4: SPEC[MU-POP] JUMP[ASMAP3] $
01m0155                                                       ];[
01m0155 13010 010731170003260174161620Z5431456000            ASMAP4: SPEC[MU-POP] JUMP[ASMAP3] $
01 0155                                                       ].REPEAT F4SW
01 0156
01 0157                                                       ;Reset Grinnell, Set dispatch addresses
01 0158 13011 010731170031060546021523655714160000           ASRST:  D[CONST 10] ROT[30] DEST[IOD] SPEC[IOB-OUT] NORM $
01 0159                                                                       ;Assert RESET
01 0160 13012 010241170431060554161523654114160000                   MAPF[AS-DATA] CYLEN[IOB-OUT] ALU[0] DEST[IOD] SPEC[IOB-OUT] $
01 0161                                                                       ;Clear RESET
01 0162 13013 010241170400060554161143654114160000                   MAPF[AS-DATA] CYLEN[IOB-OUT] ALU[0] DEST[ASBASE] $
01 0163                                                                       ;No table yet.
01 0164                                                       .REPEAT 1 - F4SW [
01 0165                                                                       D[CONST (AS-DISPATCH / 100)] ROT[30] DEST[Q] NORM $
01 0166                                                                       ;Construct dispatch address: high 6 bits.  Finish IOB
01 0166                                                        RESET
01 0167                                                                       D[CONST (AS-DISPATCH \ 100)] ROT[22] ALU[DORQ]
01 0168                                                                              DEST[ASDSP] NORM POPJ $
01 0169                                                       ];.REPEAT 1 - F4SW
01 0170                                                       .REPEAT F4SW [
01 0171                                                                       D[LIT (AS-DISPATCH * 1000000)] DEST[ASBSP] POPJ $
01m0172                                                       ];[
01m0172 13014 010731170025672000001144255314160000                   D[LIT (AS-DISPATCH * 1000000)] DEST[ASDSP] POPJ $
01 0172                                                       ].REPEAT F4SW
01 0173                                                                       ;Finish contructing dispatch address and store away.
01 0174                                                                       ;We're done.
```

01 0128]
01 0129

```
02 0130                                                 .REPEAT 1 - WAITS [
02 0131                                                   .opcode[753]
02 0132                                                 ];.REPEAT 1 - WAITS
02 0133                                                 .REPEAT WAITS [
02 0134                                                   .opcode[775]
02 0135                                                 ];[
02m0136                                                   .opcode[775]
02m0136                                                  [xlist
02m0136                                                    list  ]].REPEAT WAITS
02 0136
02 0137
02 0138                                                 ;FNTWRT
02 0139 07772 0307311700000000014161620254713560000            DF/WT D[MEM] JUMP[FNTWRT] $
02 0140 07773 0103311700000000014161500054314560000            ALU[MEMAC] DEST[HOLD] JUMP[FNTWRT] $
02 0141
02 0142                                                 ;FNTRD
02 0143 07774 0107311700400000014024460255714570000            D[CONST FNTDEV] DEST[DEV-ADR] JUMP[FNTRD] $
02 0144 07775 0107311700400000014024460255714570000            D[CONST FNTDEV] DEST[DEV-ADR] JUMP[FNTRD] $
02 0145
02m0146                                                   .reloc
02m0146                                                  [.USE[HILOC]
02m0146                                                  [ xlist
02 0147                                                    list ]                                  ]
02 0148
02 0149                                                 FNTWRT: D[CONST FNTDEV] DEST[DEV-ADR]
02 0150                                                 .repeat waitsfix [
02 0151                                                           COND[USER] PUSHJ[CKIOTU]
02m0152                                                 ];[
02m0152                                                           COND[USER] PUSHJ[CKIOTU]
02 0152                                                 ].repeat waitsfix
02 0153 13015 0107311240400000014024462255714570000    $         ;Setup device code.  Make sure we're EXEC or IOT-USER mode
02 0154 13016 0107303700000605432116236577141600       0       D[IR] ROT[12. + 1] MASK[4] DEST[AR Q] $
02 0155                                                           ;Extract AC field.
02 0156 13017 0106610000000000140356202556145600       0       D[CONST 16] ALU[D#Q] CS50 COND[-ZERO] JUMP[FNTWRA] $
02 0157                                                           ;Is it a Select Interface command ?
02 0158 13020 0107310020000000014161620254615560000            D[MEM] CS50 COND[ZERO] JUMP[FNTWRB] $
02 0159                                                           ;Yes. Jump if selecting interface 0.
02 0160 13021 0107311700002271140452002557145600       0       D[CONST FNTDEV2] DEST[D%AMEM10] NORM JUMP[MAIN] $
02 0161                                                           ;Select interface 1.
02 0162 13022 0107311700002271140252002557145600       0     FNTWRB: D[CONST FNTDEV] DEST[D%AMEM10] NORM JUMP[MAIN] $
02 0163 13023 0107311700400605541604636643141700       0     FNTWRA: D[D%AMEM10] DEST[DEV-ADR] NORM $
02 0164                                                           ;Get device adr of selected interface.
02 0165 13024 0106013700000000000001623655314160000            D[LABEL FNTDSP] ALU[D+Q] DEST[AR] $
02 0166                                                           ;Calc disp addr
02 0167 13025 0107311700000000141615222547155600       0       D[MEM] DEST[IOD] PUSHJ[FNTWR1] $
02 0168 13026 0102310000002271141616202542145600       0       ALU[Q] CS50 COND[-OBUS=0] JUMP[MAIN] $
02 0169                                                           ;Output done.  Was it to CMD reg ?
02 0170 13027 0107311700002271141610402547155600       0       D[MEM] DEST[D%AMEM2] JUMP[MAIN] $
02 0171                                                           ;Yes.  Remember the data.
02 0172 13030 0107311700310000001600072541145600       0     FNTWR1: START-OUT D[AR] ROT[MUA-ROT] LONG ODISP $
02 0173                                                           ;Start on FBUS output, and dispatch to proper sub-select.
02 0174
02 0175                                                 FNTRD:
02 0176 13031 0107311700400605541604636643141700       0       D[D%AMEM10] DEST[DEV-ADR] NORM $
02 0177                                                 .repeat waitsfix [
02 0178                                                           COND[USER] PUSHJ[CKIOTU]
02m0179                                                 ];[
02m0179                                                           COND[USER] PUSHJ[CKIOTU]
02 0179                                                 ].repeat waitsfix
02 0180 13032 0107311240000000141616222543145600       0       $         ;Get device adr of selected interface.
02 0181                                                           ;Make sure we're EXEC or IOT-USER mode
02 0182 13033 0107303700000605432116236577141600       0       D[IR] ROT[12. + 1] MASK[4] DEST[AR Q] $
02 0183                                                           ;Extract AC field.
02 0184 13034 0106013700000000000001623655314160000            D[LABEL FNTDSP] ALU[D+Q] DEST[AR] $
02 0185                                                           ;Calc disp addr
02 0186 13035 0107311700000000141616222543145600       0       PUSHJ[FNTRD1] $
02 0187 13036 0107313100002266541615612543045600       0       D[AR] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<20] LBJUMP[SEQI] ]$
02 0188 13037 0107311700300000001600072541145600       0     FNTRD1: START-IN  D[AR] ROT[MUA-ROT] LONG ODISP $
02 0189                                                           ;Start an input operation, dispatch to get right sub-sel
02 0190
02 0191       0 FNTDSP:         xx = 0
02 0192                                                   .repeat 17
02 0193                                                     [   MAPF[xx] D[IOD] DEST[AR] LONG POPJ $
02m0194                                                             xx = xx + 1 ]
02m0194
02m0194 13040 0107313700000605541616242571141600       0     [   MAPF[xx] D[IOD] DEST[AR] LONG POPJ $
02m0194       1                   xx = xx + 1 ]
02m0194 13041 0107313701000605541616242571141600       0     [   MAPF[xx] D[IOD] DEST[AR] LONG POPJ $
02m0194       2                   xx = xx + 1 ]
02m0194 13042 0107313702000605541616242571141600       0     [   MAPF[xx] D[IOD] DEST[AR] LONG POPJ $
02m0194       3                   xx = xx + 1 ]
02m0194 13043 0107313703000605541616242571141600       0     [   MAPF[xx] D[IOD] DEST[AR] LONG POPJ $
02m0194       4                   xx = xx + 1 ]
02m0194 13044 0107313704000605541616242571141600       0     [   MAPF[xx] D[IOD] DEST[AR] LONG POPJ $
02m0194       5                   xx = xx + 1 ]
02m0194 13045 0107313705000605541616242571141600       0     [   MAPF[xx] D[IOD] DEST[AR] LONG POPJ $
02m0194       6                   xx = xx + 1 ]
02m0194 13046 0107313706000605541616242571141600       0     [   MAPF[xx] D[IOD] DEST[AR] LONG POPJ $
02m0194       7                   xx = xx + 1 ]
02m0194 13047 0107313707000605541616242571141600       0     [   MAPF[xx] D[IOD] DEST[AR] LONG POPJ $
02m0194       10                        xx = xx + 1 ]
02m0194 13050 0107313710000605541616242571141600       0     [   MAPF[xx] D[IOD] DEST[AR] LONG POPJ $
02m0194       11                        xx = xx + 1 ]
02m0194 13051 0107313711000605541616242571141600       0     [   MAPF[xx] D[IOD] DEST[AR] LONG POPJ $
02m0194       12                        xx = xx + 1 ]
02m0194 13052 0107313712000605541616242571141600       0     [   MAPF[xx] D[IOD] DEST[AR] LONG POPJ $
02m0194       13                        xx = xx + 1 ]
02m0194 13053 0107313713000605541616242571141600       0     [   MAPF[xx] D[IOD] DEST[AR] LONG POPJ $
02m0194       14                        xx = xx + 1 ]
02m0194 13054 0107313714000605541616242571141600       0     [   MAPF[xx] D[IOD] DEST[AR] LONG POPJ $
02m0194       15                        xx = xx + 1 ]
02m0194 13055 0107313715000605541616242571141600       0     [   MAPF[xx] D[IOD] DEST[AR] LONG POPJ $
02m0194       16                        xx = xx + 1 ]
02m0194 13056 0107313716000605541616242571141600       0     [   MAPF[xx] D[IOD] DEST[AR] LONG POPJ $
02 0194       17                        xx = xx + 1 ]           MAPF[17] D[IR] MASK[3]
02 0194 13057 0107311717000605540070242571141600       0      DEST[D%AMEM1] POPJ $
02 0195                                                           ;Sub-sel 17... set the PI channel assignment.
02 0196
02 0197                                                         ;Reset the interface.
02 0198 13060 0102411700310605541615236543141600       0     FNTRST: START-OUT ALU[0] DEST[IOD] $
```

```
02 0203                                                         ;Init the interrupt addr.
02 0204 13063 010731170031060554044463655571417000             START-OUT D[CONST FNTDEV2] DEST[DEV-ADDRESS] $
02 0205                                                         ;Select secondary interface and set its CMD reg. to 0
02 0206 13064 010731170000000000000100425531416000             MAPF[0] D[LABEL FNTINT] DEST[D%AMEM0] POPJ $
02 0207                                                         ;Init the interrupt addr.
02 0208
02 0209                                                       ;Interrupts come here.
02 0210 13065 010730170000060554141623661314160000     FNTINT: D[D%AMEM2] MASK[LEFT] DEST[Q] $
02 0211                                                         ;Get copy of contents of CMD reg.
02 0212 13066 010651170031060550203523655714160000             START-OUT D[CONST 1] ROT[35. - 2] ALU[-D&Q] DEST[IOD] $
02 0213                                                         ;Clear the interrupt enable bit.
02 0214 13067 010731370000203434007620260714560000             MAPF[0] D[D%AMEM1] MASK[3] DEST[AR] JUMP[PIGEN] $
02 0215                                                         ;Request a PI (if the assigned channel is not 0).
02 0216
```

```
03 0217
03 0218
03 0219
03 0220                                         ;INTERRUPTABLE F3 TAPE UCODE CONVERTED TO F4
03 0221
03 0222                                         ;-----------------------------------------------------------------------
03 0222                                         ------
03 0223                                         ;
03 0224                                         ;       Ucode for KENNEDY or PERTEC formatter and DMA tape controller.
03 0225                                         ;
03 0226                                         ;-----------------------------------------------------------------------
03 0226                                         ------
03 0227
03 0228
03 0229                                         ;TAPE READING AND WRITING CODE
03 0230                                         ; A-MEM USEAGE:
03 0231                                         ;               0       DISPATCH ADDR.
03 0232                                         ;               0       ;DATA PACKING MODE
03 0233                                         ;                        BIT 0: 0=PDP-10 CORE-DUMP, 1=INDUSTRY (32-bit
                                                 mode)
03 0234                                         ;                        BIT 1: NRZI Kluge Mode (to read old CCRMA & SCI
03 0234                                          tapes)
03 0235                                         ;               1       Timeout values for NCNTWT and FMNBWT.
03 0236                                         ;               2       Copy of last writ to TP.WC,
03 0237                                         ;                       Status at ENDX of tape operation.
03 0238                                         ;               3       DON'T USE... Storing into it clobbers IR left !
03 0239                                         ;               4       Next mem adr of xfer (STRTDC) -- 0 during
                                                 non-data ops
03 0240                                         ;               5       Remaining word count in current WCMA (STRTDC)
03 0241                                         ;               6       Not currently used-- will be  Data Channel PC
03 0242                                         ;               7       Interrupt state and transfer flags:
03 0243                                         ;                       1b0 = read, 1b1 = waiting for formatter to become
                                                 idle
03 0244                                         ;                       1b2 = mtop interrupt (set with read)
03 0245                                         ;                       18-35 CONI bits:
03 0246                                         ;                       7b35 = pi assignment. 10 = interrupt flag.
03 0247
03 0248                                         ;MAPF values
03 0249
03 0250      4 TP.RS = 4        ;read status (from formatter)
03 0251      2 TP.RC = 2        ;read control (controller status and un-fifo'd read data)
03 0251
03 0252
03 0253      2 TP.WF = 2        ;write formatter (send ctrl bits to formatter)
03 0254      4 TP.WM = 4        ;write mode control reg.
03 0255      1 TP.WC = 1        ;write control reg.
03 0256      5 TP.MR = 5        ;give Master Reset
03 0257      3 TP.WMA = 3       ;write (load) the CNTMA reg. (count and MA)
03 0258
03 0259                                         ;;MICROCODE TAPE STATUS BITS RETURNED IN LH AC AFTER MTRD AND MTWR and
                                                 MTOP IOTS
03 0260                                         ;MICFLG←←400000          ;SOME FLAG IS BEING RETURNED BY THE MICROCODE
03 0261                                         ;MICEOF←←200000          ;EOF SEEN FLAG FROM MICROCODE
03 0262                                         ;MICTPE←←100000          ;TAPE PARITY ERROR FLAG FROM MICROCODE
03 0263                                         ;MICLNE←←040000          ;LENGTH ERROR FLAG FROM MICROCODE
03 0264
03 0265                                         ;;HARDWARE STATUS BIT DEFINITIONS RETURNED IN RIGHT HALF BY MTRS
03 0266                                         ;     ----------- ----------- ----------- ----------- -----------
03 0266                                          -----------
03 0267                                         ;    !   !   !   !   !   !   !   !   !   !   !   !   !   !   !   !   !
03 0267                                          !   !
03 0268                                         ;    !   !   !   !   !   !   !   !   !   !   !   !   !   !ON
03 0268                                          !REW!WRT!BOT!RDY!EOT!
03 0269                                         ;    !   !   !   !   !   !   !   !   !   !   !   !   !LIN!   !PRT!   !
03 0269                                          !   !
03 0270                                         ;     ----------- ----------- ----------- ----------- -----------
03 0270                                          -----------
03 0271                                         ;     18                                                30  31  32  33  34
03 0271                                           35
03 0272                                         ;
03 0273                                         ;HDWNLN←←000040          ;UNIT IS ONLINE
03 0274                                         ;HDWREW←←000020          ;UNIT IS REWINDING
03 0275                                         ;HDWFPT←←000010          ;UNIT IS WRITE PROTECTED
03 0276                                         ;HDWBOT←←000004          ;UNIT IS AT LOAD POINT
03 0277                                         ;HDWRDY←←000002          ;UNIT IS READY
03 0278                                         ;HDWEOT←←000001          ;UNIT IS AT END OF TAPE
03 0279                                         ;
03 0280                                         ;;CONI BITS
03 0281                                         ;     ----------- ----------- ----------- ----------- -----------
03 0281                                          -----------
03 0282                                         ;    !   !   !   !   !   !   !   !   !   !   !   !   !   !   !   !   !
03 0282                                             !
03 0283                                         ;    !OVR!CTL!FMT!   !   !   !   !   !   !   !   !   !   !   !INT!  PI
03 0283                                          CHAN  !
03 0284                                         ;    !RUN!BSY!BSY!   !   !   !   !   !   !   !   !   !   !   !FLG!
03 0284                                             !
03 0285                                         ;     ----------- ----------- ----------- ----------- -----------
03 0285                                          -----------
03 0286                                         ;     18                                                30  31  32  33
03 0286                                           35
03 0287                                         ;
03 0288
03 0289
```

```
04 0290
04 0291
04 0292
04 0293                                                 .REPEAT TAPE [
04 0294
04 0295                                                   .opcode[725]  ;TAPE IOTS
04 0296
04 0297                                                 ;725 - Obsolete version of read (no word count, E = core address.)
04 0298                                                        D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TAPERD] $
04 0299                                                        D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TAPERD] $
04 0300
04 0301                                                 ;726 - MTAPE
04 0302                                                        D[CONST 7] DEST[DEV-ADR] NORM JUMP[TAPEMT] $
04 0303                                                        D[CONST 7] DEST[DEV-ADR] NORM JUMP[TAPEMT] $
04 0304                                                 ;727 - READ STATUS
04 0305                                                        D[CONST 7] DEST[DEV-ADR] NORM JUMP[TAPERS] $
04 0306                                                        D[CONST 7] DEST[DEV-ADR] NORM JUMP[TAPERS] $
04 0307                                                 ;730 - READ 1 RECORD, STORE UP TO (AC) WORDS STARTING E.
04 0308                                                        D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TAPENR] $
04 0309                                                        D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TAPENR] $
04 0310                                                 ;731 - WRITE 1 record of E words starting from CORE LOC IN AC.
04 0311                                                        D[CONST 7] DEST[DEV-ADR] NORM JUMP[TAPEWR] $
04 0312                                                        D[CONST 7] DEST[DEV-ADR] NORM JUMP[TAPEWR] $
04 0313                                                 ;732 - READ 1 record, store up to E words starting at (AC).
04 0314                                                        D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TAPERX] $
04 0315                                                        D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TAPERX] $
04 0316
04 0317                                                 .REPEAT 1 - WAITS [     ;These are merged under one opcode and act like
04 0318                                                                         ;normal PDP-10 I/O instructions under WAITS.
04 0319                                                 ;733
04 0320                                                        D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TPCONI] $
04 0321                                                        D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TPCONI] $
04 0322
04 0323                                                 ;734
04 0324                                                        D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TPCONO] $
04 0325                                                        D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TPCONO] $
04 0326
04 0327                                                 ;735
04 0328                                                        D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TPCNSO] $
04 0329                                                        D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TPCNSO] $
04 0330
04 0331                                                 ;736
04 0332                                                        D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TPDATI] $
04 0333                                                        D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TPDATI] $
04 0334                                                 ];REPEAT 1 - WAITS
04 0335
04m0336                                                  ] [
04m0336
04m0336                                                   .opcode[725]   [xlist
04m0336                                                    list  ];TAPE IOTS
04m0336
04m0336                                                 ;725 - Obsolete version of read (no word count, E = core address.)
04m0336 07652 010731170040000014016460255714570000            D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TAPERD] $
04m0336 07653 010731170040000014016460255714570000            D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TAPERD] $
04m0336
04m0336                                                 ;726 - MTAPE
04m0336 07654 010731170040000014016460255714570000            D[CONST 7] DEST[DEV-ADR] NORM JUMP[TAPEMT] $
04m0336 07655 010731170040000014016460255714570000            D[CONST 7] DEST[DEV-ADR] NORM JUMP[TAPEMT] $
04m0336                                                 ;727 - READ STATUS
04m0336 07656 010731170040000014016460255714570000            D[CONST 7] DEST[DEV-ADR] NORM JUMP[TAPERS] $
04m0336 07657 010731170040000014016460255714570000            D[CONST 7] DEST[DEV-ADR] NORM JUMP[TAPERS] $
04m0336                                                 ;730 - READ 1 RECORD, STORE UP TO (AC) WORDS STARTING E.
04m0336 07660 010731170040000014016460255714570000            D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TAPENR] $
04m0336 07661 010731170040000014016460255714570000            D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TAPENR] $
04m0336                                                 ;731 - WRITE 1 record of E words starting from CORE LOC IN AC.
04m0336 07662 010731170040000014016460255714570000            D[CONST 7] DEST[DEV-ADR] NORM JUMP[TAPEWR] $
04m0336 07663 010731170040000014016460255714570000            D[CONST 7] DEST[DEV-ADR] NORM JUMP[TAPEWR] $
04m0336                                                 ;732 - READ 1 record, store up to E words starting at (AC).
04m0336 07664 010731170040000014016460255714570000            D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TAPERX] $
04m0336 07665 010731170040000014016460255714570000            D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TAPERX] $
04m0336
04m0336                                                 .REPEAT 1 - WAITS [     ;These are merged under one opcode and act like
04m0336                                                                         ;normal PDP-10 I/O instructions under WAITS.
04m0336                                                 ;733
04m0336                                                        D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TPCONI] $
04m0336                                                        D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TPCONI] $
04m0336
04m0336                                                 ;734
04m0336                                                        D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TPCONO] $
04m0336                                                        D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TPCONO] $
04m0336
04m0336                                                 ;735
04m0336                                                        D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TPCNSO] $
04m0336                                                        D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TPCNSO] $
04m0336
04m0336                                                 ;736
04m0336                                                        D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TPDATI] $
04m0336                                                        D[CONST 7] DEST[DEV-ADR] NORM  JUMP[TPDATI] $
04m0336                                                 ];REPEAT 1 - WAITS
04m0336
04 0336                                                  ];TAPE
04 0337
04 0338
04 0339                                                   .repeat 1 - TAPE[ ILLIOT[725 736] ]
04 0340
04 0341
04m0342                                                   .RELOC
04m0342                                                  [.USE[HILOC]
04m0342                                                  [ xlist
04 0343                                                    list ]                                  ]
04 0344                                                 .REPEAT TBOOT [
04 0345
04 0346                                                 TAPEOI: NEOI $          ;WHEN CONFLICT WOULD ARISE IF EOI IN SAME INSTR
04 0347
04 0348
04 0349                                                 ;RESET the formatter and drive.
04 0350
04 0351                                                 TAPRST: D[LABEL TAPINT] DEST[D%AMEM0] $
04 0352                                                        ALU[0] DEST[D%AMEM7] NORM $
04 0353                                                 KNYCLR:
```

```
04 0359                                                 KNYRGO: NORM PUSHJ[DCINIT] $
04 0360                                                         ;Init the data channel.
04 0361                                                 D[D%AMEM0]  ROT[1] MASK[1] DEST[AR] NORM $
04 0362                                                         ;Get the 32-bit mode flag.
04 0363                                                 START-OUT D[AR] ROT[35. - 29.] DEST[IOD] NORM $
04 0364                                                         ;Position it for the hardware.
04 0365                                                 MAPF[TP.WM] D[CONST 0] DEST[Q]   C-OUT JUMP[KNYGOA] $
04 0366                                                         ;Send command to formatter.
04 0367
04 0368                                                   ;Send a tape-motion command to the formatter.  Call with command bits
04 0369                                                   ;  (except for FMTR ENABLE and GO) in Q.  Clobbers Q, HOLD, AR
04 0370                                                 KNYGOA: START-OUT D[CONST 1] ROT[35. - 26.] DEST[IOD] $
04 0371                                                          ;Send the FORMATTER ENABLE bit.
04 0372                                                 MAPF[TP.WF] D[CONST 1] ROT[35. - 26.] ALU[DORQ] DEST[Q]
04 0372                                                  PUSHJ[FMNBWT] $
04 0373                                                           ;Add the FMTR ENBL bit to the command word.
04 0374                                                           ;Wait for FORMATTER NOT BUSY.
04 0375                                                 START-OUT D[CONST 1] ROT[35. - 33.] ALU[D#Q] DEST[IOD] C-OUT $
04 0376                                                           ;Set the GO bit to fmtr (except: on RWD, CLEAR the bit !)
04 0377                                                 MAPF[TP.WF] START-OUT D[CONST 1] ROT[35. - 33.] ALU[-D&Q]
04 0377                                                  DEST[IOD] C550 $
04 0378                                                           ;Send command word again, without GO bit.
04 0379                                                 MAPF[TP.WF] LONG POPJ $
04 0380
04 0381
04 0382                                                 FMNBWT:   ;Wait for formatter to be not busy.
04 0383                                                           ;Return tape status in MEM;  timeout in 164 msec.
04 0384                                                           ;Duration of loop should be 10 usec. (for TRCHECK).
04 0385                                                     .   START-IN D[CONST 1] ROT[14.] DEST[AR] NORM $
04 0386                                                 FMNBW1: MAPF[TP.RS] D[IOD] DEST[HOLD] C800 $ ;GET STATUS BITS.
04 0387                                                 D[AR] ALU[D-1] DEST[AR] C550 OBUS<0  JUMP[FMTHNG] $
04 0388                                                 D[CONST 14.] ROT[LLOAD-ROT] DEST[LLOAD] C600 $
04 0389                                                 C500 LOOP[.] $
04 0390                                                   ;We execute this instr. 15. times, for a 7.5 usec delay.
04 0391                                                 START-IN D[MEM] ROT[7] C550 -OBUS<0 JUMP[FMNBW1] $
04 0392                                                   ;check for 'BUSY'
04 0393                                                 D[AR] DEST[D%AMEM1] NORM POPJ $
04 0394                                                   ;Save ending timeout count (for TRCHECK)
04 0395                                                 FMTHNG: SPEC[MU-POP] $
04 0396                                                 NORM PUSHJ[KNYRS1] $
04 0397                                                         ;Blast the formatter.
04 0398                                                 D[CONST 42] ROT[30.] DEST[AC] SPEC[MU-POP] NEOI $
04 0399                                                         ;Return error code for 'hung fmtr' and abort.
04 0400
04 0401                                                 NCNTWT:   ;Wait for TP CNT GO to be off.
04 0402                                                           ;Return tape status in MEM;  timeout in 164 msec.
04 0403                                                           ;Duration of loop should be 10 usec. (for TRCHECK).
04 0404                                                 NCNTW1: MAPF[TP.RC] D[IOD] DEST[HOLD] C800 $ ;GET STATUS BITS.
04 0405                                                 D[AR] ALU[D-1] DEST[AR] C550 OBUS<0 POPJ $
04 0406                                                   ;Exit if we time out-- probably just a short record.
04 0407                                                 D[CONST 14.] ROT[LLOAD-ROT] DEST[LLOAD] C600 $
04 0408                                                 C500 LOOP[.] $
04 0409                                                   ;We execute this instr. 15. times, for a 7.5 usec delay.
04 0410                                                 START-IN D[MEM] ROT[19.] C550 OBUS<0 JUMP[NCNTW1] $
04 0411                                                   ;check for not TP CNT GO
04 0412                                                 D[AR] DEST[D%AMEM1] NORM POPJ $
04 0413                                                   ;Save ending timeout count (for TRCHECK)
04 0414
04 0415
04 0416                                                 ;INITIALIZE DATA CHANNEL
04 0417
04 0418                                                 DCINIT: START-OUT $
04 0419                                                 MAPF[TP.MR] D[CONST 8.] ROT[LLOAD-ROT] DEST[LLOAD] NORM $
04 0420                                                         ;Give TP MR
04 0421                                                 START-OUT ALU[0] DEST[IOD] C600 $
04 0422                                                         ;Load 0 into cntma (the COUNT and MA registers)
04 0423                                                 MAPF[TP.WMA] START-OUT D[MASK 12.] ROT[35. - 13.] DEST[IOD] C600
04 0423                                                  $
04 0424                                                         ;Load cntma again-- since COUNT is currently  0, this
04 0424                                                  will
04 0425                                                         ; force BUF CNT to be 0.  But, we put -1 into COUNT this
04 0425
04 0426                                                         ; time, which, with BUF CNT = 0, will make BUF CNT load
04 0427                                                         ; properly when STARTDC loads cntma.
04 0428                                                 MAPF[TP.WMA] START-OUT D[CONST 2] DEST[IOD] C600 $
04 0429                                                         ;Set MBUSY
04 0430                                                 MAPF[TP.WC] C550 LOOP[.] $
04 0431                                                         ;Wait for a few usec.  This clears mem rq.
04 0432                                                 START-OUT ALU[0] DEST[IOD] NORM $
04 0433                                                         ;Clr MBUSY
04 0434                                                 MAPF[TP.WC] C600 ALU[0] DEST[D%AMEM2] POPJ $
04 0435                                                         ;Zero out copy of TP.WC contents.
04m0436                                                  ] [
04m0436
04m0436 13070 540731170000060554161623254204160000     TAPEOI: NEOI $          ;WHEN CONFLICT WOULD ARISE IF EOI IN SAME INSTR
04m0436
04m0436
04m0436                                                 ;RESET the formatter and drive.
04m0436
04m0436 13071 010731170000000000001003655314160000     TAPRST: D[LABEL TAPINT] DEST[D%AMEM0] $
04m0436 13072 010241170000060554161163654314160000             ALU[0] DEST[D%AMEM7] NORM $
04m0436                                                 KNYCLR:
04m0436 13073 010241170031060554161523654314160000     KNYRS1: START-OUT ALU[0]  DEST[IOD] $
04m0436                                                         ;Turn off "FORMATTER ENABLE"
04m0436 13074 010731170200060554161624254114160000             MAPF[TP.WF] LONG POPJ $
04m0436
04m0436                                                 ; KNYRGO -- Called to start tape motion on reads.
04m0436 13075 010731170000000014161622254314560000     KNYRGO: NORM PUSHJ[DCINIT] $
04m0436                                                         ;Init the data channel.
04m0436 13076 010731370000060540203623660314160000             D[D%AMEM0]  ROT[1] MASK[1] DEST[AR] NORM $
04m0436                                                         ;Get the 32-bit mode flag.
04m0436 13077 010731170031060541561523654314160000             START-OUT D[AR] ROT[35. - 29.] DEST[IOD] NORM $
04m0436                                                         ;Position it for the hardware.
04m0436 13100 010730170400000014001620255514560000             MAPF[TP.WM] D[CONST 0] DEST[Q]   C-OUT JUMP[KNYGOA] $
04m0436                                                         ;Send command to formatter.
04m0436
04m0436
04m0436                                                   ;Send a tape-motion command to the formatter.  Call with command bits
04m0436                                                   ;  (except for FMTR ENABLE and GO) in Q.  Clobbers Q, HOLD, AR
04m0436 13101 010731170031060542703523655571416000     KNYGOA: START-OUT D[CONST 1] ROT[35. - 26.] DEST[IOD] $
04m0436                                                          ;Send the FORMATTER ENABLE bit.
```

```
04m0436                                                 ;Set the GO bit to fmtr (except: on RWD, CLEAR the bit !)
04m0436 13104 010651170231060540403523655614165000     MAPF[TP.WF] START-OUT D[CONST 1] ROT[35. - 33.] ALU[-D&Q]
04m0436                                               DEST[IOD] C550 $
04m0436                                                 ;Send command word again, without GO bit.
04m0436 13105 010731170200060554161624254114165000     MAPF[TP.WF] LONG POPJ $
04m0436
04m0436
04m0436                                               FMNBWT:   ;Wait for formatter to be not busy.
04m0436                                                 ;Return tape status in MEM;  timeout in 164 msec.
04m0436                                                 ;Duration of loop should be 10 usec. (for TRCHECK).
04m0436 13106 010731370030060543403623655714165000     START-IN D[CONST 1] ROT[14.] DEST[AR] NORM $
04m0436 13107 010731170400060554161503655711416000   FMNBW1: MAPF[TP.RS] D[IOD] DEST[HOLD] C600 $ ;GET STATUS BITS.
04m0436 13110 010721204000000014161620254214556000     D[AR] ALU[D-1] DEST[AR] C550 OBUS<0  JUMP[FMTHNG] $
04m0436 13111 010731170010060540034003655514170000     D[CONST 14.] ROT[LLOAD-ROT] DEST[LLOAD] C600 $
04m0436 13112 010731070000262254161620254214556000     C500 LOOP[.] $
04m0436                                                   ;We execute this instr. 15. times, for a 7.5 usec delay.
04m0436 13113 010731006030262161761620254615566000     START-IN D[MEM] ROT[7] C550 -OBUS<0 JUMP[FMNBW1] $
04m0436                                                   ;check for 'BUSY'
04m0436 13114 010731170000060554161024254314165000     D[AR] DEST[D%AMEM1] NORM POPJ $
04m0436                                                   ;Save ending timeout count (for TRCHECK)
04m0436 13115 010731170003060554161623654314165000   FMTHNG: SPEC[MU-POP] $
04m0436 13116 010731170000261674161622254314556000     NORM PUSHJ[KNYRS1] $
04m0436                                                         ;Blast the formatter.
04m0436 13117 540733170003060547505623255604165000     D[CONST 4Z] ROT[30.] DEST[AC] SPEC[MU-POP] NEOI $
04m0436                                                         ;Return error code for 'hung fmtr' and abort.
04m0436
04m0436
04m0436                                               NCNTWT:   ;Wait for TP CNT GO to be off.
04m0436                                                 ;Return tape status in MEM;  timeout in 164 msec.
04m0436                                                 ;Duration of loop should be 10 usec. (for TRCHECK).
04m0436 13120 010731170200060554161503655711416000   NCNTW1: MAPF[TP.RC] D[IOD] DEST[HOLD] C800 $ ;GET STATUS BITS.
04m0436 13121 010721204000060554161624254214165000     D[AR] ALU[D-1] DEST[AR] C550 OBUS<0 POPJ $
04m0436                                                   ;Exit if we time out-- probably just a short record.
04m0436 13122 010731170010060540034003655514170000     D[CONST 14.] ROT[LLOAD-ROT] DEST[LLOAD] C600 $
04m0436 13123 010731070000262474161620254214556000     C500 LOOP[.] $
04m0436                                                   ;We execute this instr. 15. times, for a 7.5 usec delay.
04m0436 13124 010731004030262404761620254615566000     START-IN D[MEM] ROT[19.] C550 OBUS<0 JUMP[NCNTW1] $
04m0436                                                   ;check for not TP CNT GO
04m0436 13125 010731170000060554161024254314165000     D[AR] DEST[D%AMEM1] NORM POPJ $
04m0436                                                   ;Save ending timeout count (for TRCHECK)
04m0436
04m0436
04m0436
04m0436                                               ;INITIALIZE DATA CHANNEL
04m0436
04m0436 13126 010731170031060554161623654314165000   DCINIT: START-OUT $
04m0436 13127 010731170510060540020003655714170000     MAPF[TP.MR] D[CONST 8.] ROT[LLOAD-ROT] DEST[LLOAD] NORM $
04m0436                                                         ;Give TP MR
04m0436 13130 010241170031060554161523654114165000     START-OUT ALU[0] DEST[IOD] C600 $
04m0436                                                         ;Load 0 into cntma (the COUNT and MA registers)
04m0436 13131 010731170331060545431523647114165000     MAPF[TP.WMA] START-OUT D[MASK 12.] ROT[35. - 13.] DEST[IOD] C600
04m0436                                               $
04m0436                                                         ;Load cntma again-- since COUNT is currently  0, this
04m0436                                               will
04m0436                                                         ; force BUF CNT to be 0.  But, we put -1 into COUNT this
04m0436
04m0436                                                         ; time, which, with BUF CNT = 0, will make BUF CNT load
04m0436                                                         ; properly when STARTDC loads cntma.
04m0436 13132 010731170331060554005523655514165000     MAPF[TP.WMA] START-OUT D[CONST 2] DEST[IOD] C600 $
04m0436                                                         ;Set MBUSY
04m0436 13133 010731070100262674161620254214556000     MAPF[TP.WC] C550 LOOP[.] $
04m0436                                                         ;Wait for a few usec.  This clears mem rq.
04m0436 13134 010241170031060554161523654314165000     START-OUT ALU[0] DEST[IOD] NORM $
04m0436                                                         ;Clr MBUSY
04m0436 13135 010241170100060554161044254114165000     MAPF[TP.WC] C600 ALU[0] DEST[D%AMEM2] POPJ $
04m0436                                                         ;Zero out copy of TP.WC contents.
04 0436                                               ];TBOOT
04 0437
04 0438
04 0439                                               .REPEAT TAPE [
04 0440
04 0441                                               ;START DATA CHANNEL to write or read a record. Transfer up to C(IR-ADR)
04 0441                                               words;
04 0442                                               ;  starting address is in A-MEM[4].
04 0443                                               ;  This routine is for interrupt type transfers - does not wait for
04 0444                                               ;  transfer to complete, but enables for not FMRT BSY and not TP CNT GO.
04 0444
04 0445
04 0446                                               INT-STARTDC:
04 0447                                                 D[IR] MASK[18.] NORM DEST[D%AMEM5] PUSHJ[DCGO] $
04 0448                                                   ;Set word count. (Better not be zero.)
04 0449                                                 MAPF[0] D[CONST 12] ROT[12.] DEST[Q] NORM $
04 0450                                                   ;MAPF[0] FOR DCGO. SETUP BITS FOR -FMTR BSY AND -TP CNT GO.
04 0451                                                 START-OUT D[D%AMEM2] ALU[DORQ] DEST[IOD] C600 JUMP[WCPOPJ] $
04 0452                                                   ;Enable for -FMTR BSY and -TP CNT GO.
04 0453
04 0454
04 0455                                               ;START DATA CHANNEL to write or read a record.  Transfer up to C(IR-ADR)
04 0455                                               words;
04 0456                                               ;  starting address is in A-MEM[4].
04 0457                                               ;  This is used for non-interrupt transfers. Waits for transfer
04 0457                                               completion.
04 0458
04 0459                                               STRTDC: D[IR] MASK[18.] NORM DEST[D%AMEM5] $
04 0460                                                         ;Initialize word count
04 0461                                                 D[LIT 40000] DEST[D%AMEM1] NORM $
04 0462                                                         ;Initialize timeout count (for NCNTWT and FMNBWT)
04 0463                                               STDC1:  MAPF[0] C600 PUSHJ[DCGO] $
04 0464                                                   ;start xfer -- MAPF is relevant if we looped back here on READ
04 0464
04 0465                                                 MAPF[0] START-IN D[D%AMEM1] DEST[AR] C600 PUSHJ[NCNTW1] $
04 0466                                                   ;Wait for TP CNT GO to be off (continue with current timeout
04 0466                                               count)
04 0467                                                   ;MAPF is relevant if we came from DCGWR below.
04 0468                                                 D[D%AMEM7] C550 -OBUS<0 JUMP[STDC1] $
04 0469                                                   ;Jump if this is a WRITE operation.
04 0470                                                 START-OUT D[AR] C550 -OBUS<0 JUMP[STDC1] $
04 0471                                                   ;READ op, so set MEM RQ to store last word.
04 0472                                                   ;Jump unless we timed out waiting for ENDX of last COUNT.
04 0473                                               WCDONE: MAPF[0] START-IN D[D%AMEM1] DEST[AR] C600 JUMP[FMNBW1] $
04 0474                                                   ;Enter FMNBWT with the timeout count left from NCNTWT.
04 0475                                                   ;The MAPF sets MEM RQ if we have fallen in from previous instr.
```

```
04 0480                                                         ;Jump if it is less than 1400
04 0481                                                  D[LIT 1400] DEST[Q AR] NORM $
04 0482                                                         ;It is not. Use 1400 instead.
04 0483                                                 DCG1:   D[MEM] ALU[D-Q] DEST[D%AMEM5] NORM $
04 0484                                                         ;Decrement remaining WC by amount of current WC.
04 0485                                                 D[D%AMEM4] DEST[HOLD] SHORT $
04 0486                                                         ;Get current starting adr
04 0487                                                 D[MEM] ALU[D+Q] DEST[D%AMEM4] NORM $
04 0488                                                         ;Increment it by current WC.
04 0489                                                 D[D%AMEM0] C550 OBUS<0 JUMP[. + 2] $
04 0490                                                         ;Are we in 32-bit mode ?  Jump if so.
04 0491                                                  D[AR] ROT[2] ALU[D+Q] DEST[AR] NORM.JUMP[. + 2] $
04 0492                                                         ;Form byte count (=5*word count).
04 0493                                                  D[AR] ROT[2] ALU[D] DEST[AR] NORM $
04 0494                                                         ;Form byte count (=4*word count).
04 0495                                                 D[AR] ALU[D-1] DEST[AR] NORM $
04 0496                                                         ;Adjust count to be right for hdwr.
04 0497                                                 D[AR] ROT[35. - 13.] DEST[Q] NORM $
04 0498                                                         ;Align count at bit 13.
04 0499                                                 START-OUT D[MEM] MASK[21.] ALU[DORQ] DEST[IOD] NORM $
04 0500                                                         ;Include mem addr and load into COUNT, MA
04 0501                                                         ;Also sets BUF CNT to 0 (if IN) or 5 (if OUT)-- this
04 0502                                                         ; depends on COUNT not =0 and BUF CNT not = 4 or 5
04 0503                                                 MAPF[TP.WMA] D[CONST 2] DEST[D%AMEM2] C600 $
04 0504                                                         ;Remember what we are about to set bits in TP.WC to so we
04 0504
04 0505                                                         ; can set the enable bits in TP.WC later.
04 0506                                                 START-OUT D[CONST 2] DEST[IOD] C600 $
04 0507                                                         ;Set M BUSY
04 0508                                                 MAPF[TP.WC] START-OUT C600 $
04 0509                                                         ;Set CNT GO
04 0510                                                 MAPF[13] D[D%AMEM7] C600 OBUS<0 POPJ $
04 0511                                                         ;Return if this is a READ op.
04 0512                                                 DCGWR:  START-OUT D[D%AMEM5] C550 -OBUS=0 POPJ $
04 0513                                                         ;Set MEM RQ to fetch first word.
04 0514                                                         ;Return unless this is the last WCMA in the record.
04 0515                                                 MAPF[0] D[CONST 12] DEST[D%AMEM2] C600 $
04 0516                                                         ;Remember what we set TP.WC to so we can set enable bits
04 0516
04 0517                                                         ; later.
04 0518                                                 START-OUT D[CONST 12] DEST[IOD] NORM  $
04 0519                                                         ;Last one.  Set TP ENB LAST BYTE as well as M BUSY
04 0520                                                 WCPOPJ: MAPF[TP.WC] C600 POPJ $
04 0521
04 0522                                                 WCDONE1:SPEC[MU-POP] JUMP[WCDONE] $
04 0523
04 0524
04 0525                                                   ;Routine to gather ending status for a tape operation.
04 0526                                                   ;Returns status + ending MA in Q
04 0527
04 0528                                                 ITRCHK: START-IN NORM $
04 0529                                                         MAPF[TP.RS] D[IOD] DEST[HOLD] C800 JUMP[TRCHKB] $
04 0530                                                           ;Get tape status in MEM and go assemble status in Q.
04 0531
04 0532                                                   ;Here to do the data xfer for a non-interrupt read operation.
04 0533                                                 TRP2:   D[D%AMEM7] DEST[Q] SHORT $
04 0534                                                           ;SET READ FLAG IN A-MEM[7]
04 0535                                                         D[CONST 1] ROT[35.] ALU[DORQ] DEST[D%AMEM7] NORM PUSHJ[STRTDC] $
04 0535
04 0536                                                           ;Flag op. as a READ, start the data channel
04 0537                                                         JUMP[TRCHKB] $
04 0538
04 0539                                                 TRCHECK:
04 0540                                                         PUSHJ[FMNBWT] $
04 0541                                                           ;Wait for formatter not busy.
04 0542                                                 TRCHKB: ALU[0] DEST[Q] NORM $
04 0543                                                           ;Assume no errors.
04 0544                                                         D[MEM] ROT[32.] C550 -OBUS<0 PUSHJ[TRERR] $
04 0545                                                           ;check for hard error status.
04 0546                                                         D[MEM] ROT[33.] C550 -OBUS<0 PUSHJ[TREOF] $
04 0547                                                           ;check for EOF seen by formatter
04 0548                                                 TRDONE: START-IN NORM $
04 0549                                                         MAPF[TP.RC] START-IN D[IOD] DEST[AR] C600  $
04 0550                                                           ;Get data channel status bits.
04 0551                                                         MAPF[3] D[IOD] MASK[21.] ALU[DORQ] DEST[Q D%AMEM5] C800 $
04 0552                                                           ;Include ending MA value in status info.
04 0553                                                         D[AR] ROT[19.] C550 -OBUS<0 JUMP[TPMAOK] $
04 0554                                                           ;If TP CNT GO is off,
04 0555                                                         D[AR] ROT[24.] MASK[3] ALU[D-1] C550 -OBUS=0 JUMP[TPMAOK] $
04 0556                                                           ;  or if BUF CNT is not =1, then right no. of words were
04 0556                                                  stored.
04 0557                                                          ALU[Q-1] DEST[Q D%AMEM5] NORM $
04 0558                                                            ;Otherwise, 1 extra word was stored, so decrement ending MA.
04 0559
04 0560                                                 ;Note: we no longer support NRZI kludge mode, and no longer
04 0561                                                 ; clear part of read buffer that no data was read into.
04 0562
04 0563                                                 TPMAOK: D[D%AMEM5] DEST[Q] NORM POPJ $
04 0564                                                           ;Recover ending status and return.
04 0565
04 0566                                                 TREOF:  D[CONST 60] ROT[30.] ALU[DORQ] DEST[Q]  NORM POPJ $
04 0567                                                 TRERR:  D[CONST 50] ROT[30.] ALU[DORQ] DEST[Q] NORM POPJ $
04 0568                                                           ;Flag hard read error to progm.
04 0569
04 0570                                                  ;OPCODE 726 -- MTAPE FUNCTIONS.
04 0571                                                         ;Set/Clear NRZI kludge mode used to be fns 60/61.
04 0572
04 0573
04 0574                                                 TAPEMT: D[IR] MASK [18.] DEST[Q] NORM
04 0575                                                 .repeat waitsfix [
04 0576                                                                 COND[USER] PUSHJ[CKIOTU]
04 0577                                                 ];.repeat waitsfix
04 0578                                                 $
04 0579                                                         D[CONST 60] ALU[D#Q] COND[OBUS=0] JUMP[MUUO] C550 $
04 0580                                                         D[CONST 61] ALU[D#Q] COND[OBUS=0] JUMP[MUUO] C550 $
04 0581                                                                 ;NO LONGER SUPPORT NRZI MODE
04 0582                                                         NORM PUSHJ[. + 2] $
04 0583                                                         NEOI $          ;HDWR HACK TRYING TO PROGRAM!
04 0584
04 0585                                                                 D[CONST 0] ALU[D#Q] COND[OBUS=0] JUMP[TAPERW] C550 $
```

```
04 0591                                                 D[CONST 51] ALU[D#Q] COND[OBUS=0] JUMP[TPSETDMP] C550 $
04 0592
04 0593                                                 D[D%AMEM7] MASK[3] -OBUS=0 C550 JUMP[INT-MT] $
04 0594                                                         ;JUMP IF THIS IS INTERRUPT VERSION OF INSTRUCTION.
04 0595                                                 ALU[0] DEST[D%AMEM4] NORM PUSHJ[TRCHECK] $
04 0596                                                     ;Wait for op. to finish and get ending status in Q.
04 0597                                                     ; The 0 in A-MEM[4] prevents TRCHECK from clearing read buf !
04 0597
04 0598                                                 ALU[Q] DEST[AC] SPEC[MU-POP] NEOI $
04 0599                                                     ;Return status to caller in his AC (same as READ)
04 0600
04 0601
04 0602                                         TWREOF:  ;WRITE AN ENDX OF FILE (TAPE MARK)
04 0603                                                 D[CONST 1] ROT[35. - 28.] DEST[Q] NORM $
04 0604                                                   ;Get WFM (WRITE EOF) cmd bit for formatter
04 0605                                         TERAS1: D[CONST 1] ROT[35. - 22.] ALU[DORQ] DEST[HOLD] NORM JUMP[KNYMTP]
                                                 $
04 0606                                                   ;Add WRT CMD bit and start command.
04 0607
04 0608                                         TERASE: D[CONST 3] ROT[35. - 29.] DEST[Q] NORM JUMP[TERAS1] $
04 0609                                                   ;ERASE A 3.75" GAP ON THE TAPE (get WFM and ERASE bits for
                                                 formatter)
04 0610
04m0611                                                  ] [
04m0611
04m0611                                          ;START DATA CHANNEL to write or read a record. Transfer up to C(IR-ADR)
                                                  words;
04m0611                                          ;  starting address is in A-MEM[4].
04m0611                                          ;  This routine is for interrupt type transfers - does not wait for
04m0611                                          ;  transfer to complete, but enables for not FMRT BSY and not TP CNT GO.
04m0611
04m0611
04m0611                                          INT-STARTDC:
04m0611 13136 0107311700000000014045122257714560000             D[IR] MASK[18.] NORM DEST[D%AMEMS] PUSHJ[DCGO] $
04m0611                                                           ;Set word count. (Better not be zero.)
04m0611 13137 0107301700000685430256236557141600000             MAPF[0] D[CONST 12] ROT[12.] DEST[Q] NORM $
04m0611                                                           ;MAPF[0] FOR DCGO. SETUP BITS FOR -FMTR BSY AND -TP CNT GO.
04m0611 13140 0106311700310000141615202611145600000             START-OUT D[D%AMEM2] ALU[DORQ] DEST[IOD] C600 JUMP[WCPOPJ] $
04m0611                                                           ;Enable for -FMTR BSY and -TP CNT GO.
04m0611
04m0611
04m0611                                          ;START DATA CHANNEL to write or read a record.  Transfer up to C(IR-ADR)
04m0611                                           words;
04m0611                                          ;  starting address is in A-MEM[4].
04m0611                                          ;  This is used for non-interrupt transfers. Waits for transfer
04m0611                                           completion.
04m0611
04m0611 13141 0107311700000605540451236577141600000     STRTDC: D[IR] MASK[18.] NORM DEST[D%AMEMS] $
04m0611                                                           ;Initialize word count
04m0611 13142 0107311700000001000010236553141600000             D[LIT 40000] DEST[D%AMEM1] NORM $
04m0611                                                           ;Initialize timeout count (for NCNTWT and FMNBWT)
04m0611 13143 0107311700000000014161622254114560000     STDC1:  MAPF[0] C600 PUSHJ[DCGO] $
04m0611                                                           ;start xfer -- MAPF is relevant if we looped back here on READ
04m0611
04m0611 13144 0107313700302624141616222605145600000             MAPF[0] START-IN D[D%AMEM1] DEST[AR] C600 PUSHJ[NCNTW1] $
04m0611                                                           ;Wait for TP CNT GO to be off (continue with current timeout
04m0611                                                           count)
04m0611                                                           ;MAPF is relevant if we came from DCGWR below.
04m0611 13145 0107310060002630741616202636145600000             D[D%AMEM7] C550 -OBUS<0 JUMP[STDC1] $
04m0611                                                           ;Jump if this is a WRITE operation.
04m0611 13146 0107310060312630741616202542145600000             START-OUT D[AR] C550 -OBUS<0 JUMP[STDC1] $
04m0611                                                           ;READ op, so set MEM RQ to store last word.
04m0611                                                           ;Jump unless we timed out waiting for ENDX of last COUNT.
04m0611 13147 0107313700302621741616202605145600000     WCDONE: MAPF[0] START-IN D[D%AMEM1] DEST[AR] C600 JUMP[FMNBW1] $
04m0611                                                           ;Enter FMNBWT with the timeout count left from NCNTWT.
04m0611                                                           ;The MAPF sets MEM RQ if we have fallen in from previous instr.
04m0611
04m0611
04m0611 13150 0107302020000000141615002626145600000     DCGO:   D[D%AMEMS] DEST[Q AR HOLD] C550 OBUS=0  JUMP[WCDONE1] $
04m0611                                                           ;Get remaining word count. If =0, we are done.
04m0611 13151 0116110040000000001431620255614560000             D[CONST 14] ROT[6] ALU[Q-D] C550 OBUS<0 JUMP[DCG1] $
04m0611                                                           ;Jump if it is less than 1400
04m0611 13152 0107303700000000003001623655314160000              D[LIT 1400] DEST[Q AR] NORM $
04m0611                                                           ;It is not. Use 1400 instead.
04m0611 13153 0116211700000605541611236547151600000     DCG1:   D[MEM] ALU[D-Q] DEST[D%AMEMS] NORM $
04m0611                                                           ;Decrement remaining WC by amount of current WC.
04m0611 13154 0107311700000605541615036623741600000             D[D%AMEM4] DEST[HOLD] SHORT $
04m0611                                                           ;Get current starting adr
04m0611 13155 0106011700000605541611036547151600000             D[MEM] ALU[D+Q] DEST[D%AMEM4] NORM $
04m0611                                                           ;Increment it by current WC.
04m0611 13156 0107310040002634141616202602145600000             D[D%AMEM0] C550 OBUS<0 JUMP[. + 2] $
04m0611                                                           ;Are we in 32-bit mode ?  Jump if so.
04m0611 13157 0106013700002634205616202543145600000              D[AR] ROT[2] ALU[D+Q] DEST[AR] NORM JUMP[. + 2] $
04m0611                                                           ;Form byte count (=5*word count).
04m0611 13160 0107313700000605405616236543141600000              D[AR] ROT[2] ALU[D] DEST[AR] NORM $
04m0611                                                           ;Form byte count (=4*word count).
04m0611 13161 0107213700000605541616236543141600000             D[AR] ALU[D-1] DEST[AR] NORM $
04m0611                                                           ;Adjust count to be right for hdwr.
04m0611 13162 0107301700000605455616236543141600000             D[AR] ROT[35. - 13.] DEST[Q] NORM $
04m0611                                                           ;Align count at bit 13.
04m0611 13163 0106311700310605540535236547151600000             START-OUT D[MEM] MASK[21.] ALU[DORQ] DEST[IOD] NORM $
04m0611                                                           ;Include mem addr and load into COUNT, MA
04m0611                                                           ;Also sets BUF CNT to 0 (if IN) or 5 (if OUT)-- this
04m0611                                                           ; depends on COUNT not =0 and BUF CNT not = 4 or 5
04m0611 13164 0107311703000605540050436555141600000             MAPF[TP.WMA] D[CONST 2] DEST[D%AMEM2] C600 $
04m0611                                                           ;Remember what we are about to set bits in TP.WC to so we
04m0611
04m0611                                                           ; can set the enable bits in TP.WC later.
04m0611 13165 0107311700310605540055236555141600000             START-OUT D[CONST 2] DEST[IOD] C600 $
04m0611                                                           ;Set M BUSY
04m0611 13166 0107311701310605541616236541141600000             MAPF[TP.WC] START-OUT C600 $
04m0611                                                           ;Set CNT GO
04m0611 13167 0107310053000605541616242635141600000             MAPF[13] D[D%AMEM7] C600 OBUS<0 POPJ $
04m0611                                                           ;Return if this is a READ op.
04m0611 13170 0107310000310605541616242626141600000     DCGWR:  START-OUT D[D%AMEMS] C550 -OBUS=0 POPJ $
04m0611                                                           ;Set MEM RQ to fetch first word.
04m0611                                                           ;Return unless this is the last WCMA in the record.
04m0611 13171 0107311700000605540250436555141600000             MAPF[0] D[CONST 12] DEST[D%AMEM2] C600 $
04m0611                                                           ;Remember what we set TP.WC to so we can set enable bits
04m0611
```

```
04m0611 13174 010731170003263174161620254314560000       WCDONE1:SPEC[MU-POP] JUMP[WCDONE] $
04m0611
04m0611
04m0611                                                     ;Routine to gather ending status for a tape operation.
04m0611                                                     ;Returns status + ending MA in Q
04m0611
04m0611 13175 010731170030006055416162365431416000       ITRCHK: START-IN NORM $
04m0611 13176 010731170400000014161500257114560000               MAPF[TP.RS] D[IOD] DEST[HOLD] C800 JUMP[TRCHKB] $
04m0611                                                             ;Get tape status in MEM and go assemble status in Q.
04m0611
04m0611                                                     ;Here to do the data xfer for a non-interrupt read operation.
04m0611 13177 010730170000060554161623663774160000       TRPZ:   D[D%AMEM7] DEST[Q] SHORT $
04m0611                                                             ;SET READ FLAG IN A-MEM[7]
04m0611 13200 010631170000263030603162255714560000               D[CONST 1] ROT[35.] ALU[DORQ] DEST[D%AMEM7] NORM PUSHJ[STRTDC] $
04m0611
04m0611                                                             ;Flag op. as a READ, start the data channel
04m0611 13201 010731170000000014161620254314560000               JUMP[TRCHKB] $
04m0611
04m0611                                                     TRCHECK:
04m0611 13202 010731170000262154161622254314560000               PUSHJ[FMNBWT] $
04m0611                                                             ;wait for formatter not busy.
04m0611 13203 010240170000060554161623654314160000       TRCHKB: ALU[0] DEST[Q] NORM $
04m0611                                                             ;Assume no errors.
04m0611 13204 010731006000000001016162225461556000               D[MEM] ROT[32.] C550 -OBUS<0 PUSHJ[TRERR] $
04m0611                                                             ;check for hard error status.
04m0611 13205 010731006000000001036162225461556000               D[MEM] ROT[33.] C550 -OBUS<0 PUSHJ[TREOF] $
04m0611                                                             ;check for EOF seen by formatter
04m0611 13206 010731170030006055416162365431416000       TRDONE: START-IN NORM $
04m0611 13207 010731370230006055416162365711416000               MAPF[TP.RC] START-IN D[IOD] DEST[AR] C600  $
04m0611                                                             ;Get data channel status bits.
04m0611 13210 010630170300060554053123657114160000               MAPF[3] D[IOD] MASK[21.] ALU[DORQ] DEST[Q D%AMEM5] C800 $
04m0611                                                             ;Include ending MA value in status info.
04m0611 13211 010731006000000000476162025421456000               D[AR] ROT[19.] C550 -OBUS<0 JUMP[TPMAOK] $
04m0611                                                             ;If TP CNT GO is off,
04m0611 13212 010721000000000000600762025421456000               D[AR] ROT[24.] MASK[3] ALU[D-1] C550 -OBUS=0 JUMP[TPMAOK] $
04m0611                                                             ;  or if BUF CNT is not =1, then right no. of words were
04m0611                                                      stored.
04m0611 13213 010210170000060554161123654314160000                ALU[Q-1] DEST[Q D%AMEM5] NORM $
04m0611                                                              ;Otherwise, 1 extra word was stored, so decrement ending MA.
04m0611
04m0611                                                     ;Note: we no longer support NRZI kludge mode, and no longer
04m0611                                                     ; clear part of read buffer that no data was read into.
04m0611
04m0611 13214 010730170000060554161624262714160000       TPMAOK: D[D%AMEM5] DEST[Q] NORM POPJ $
04m0611                                                             ;Recover ending status and return.
04m0611
04m0611 13215 010630170000060547541624255714160000       TREOF:  D[CONST 60] ROT[30.] ALU[DORQ] DEST[Q]  NORM POPJ $
04m0611 13216 010630170000060547521624255714160000       TRERR:  D[CONST 50] ROT[30.] ALU[DORQ] DEST[Q] NORM POPJ $
04m0611                                                             ;Flag hard read error to progm.
04m0611
04m0611                                                      ;OPCODE 726 -- MTAPE FUNCTIONS.
04m0611                                                            ;Set/Clear NRZI kludge mode used to be fns 60/61.
04m0611
04m0611
04m0611                                                     TAPEMT: D[IR] MASK [18.] DEST[Q] NORM
04m0611                                                     .repeat waitsfix [
04m0611                                                                     COND[USER] PUSHJ[CKIOTU]
04m0611                                                     ];[
04m0611                                                                     COND[USER] PUSHJ[CKIOTU]
04 0611                                                     ].repeat waitsfix
04 0612 13217 010730124000000001404562225771456000       $
04 0613 13220 010661002000102114141620255614560000               D[CONST 60] ALU[D#Q] COND[OBUS=0] JUMP[MUUO] C550 $
04 0614 13221 010661002000102114141362025561456000               D[CONST 61] ALU[D#Q] COND[OBUS=0] JUMP[MUUO] C550 $
04 0615                                                                     ;NO LONGER SUPPORT NRZI MODE
04 0616 13222 010731170000264514161622254314560000               NORM PUSHJ[. + 2] $
04 0617 13223 540731170000060554161623254204160000               NEOI $          ;HDWR HACK TRYING TO PROGRAM!
04 0618
04 0619 13224 010661002000000001400162025561456000               D[CONST 0] ALU[D#Q] COND[OBUS=0] JUMP[TAPERW] C550 $
04 0620 13225 010661002000000001400362225561456000               D[CONST 1] ALU[D#Q] COND[OBUS=0] PUSHJ[TWREOF] C550 $
04 0621 13226 010661002000000001401362225561456000               D[CONST 5] ALU[D#Q] COND[OBUS=0] PUSHJ[TAPEFR] C550 $
04 0622 13227 010661002000000001401562225561456000               D[CONST 6] ALU[D#Q] COND[OBUS=0] PUSHJ[TAPEBR] C550 $
04 0623 13230 010661002000000001402762225561456000               D[CONST 13] ALU[D#Q] COND[OBUS=0] PUSHJ[TERASE] C550 $
04 0624 13231 010661002000000001412162025561456000               D[CONST 50] ALU[D#Q] COND[OBUS=0] JUMP[TPSETIND] C550 $
04 0625 13232 010661002000000001412362025561456000               D[CONST 51] ALU[D#Q] COND[OBUS=0] JUMP[TPSETDMP] C550 $
04 0626
04 0627 13233 010731000000000001400762026361456000               D[D%AMEM7] MASK[3] -OBUS=0 C550 JUMP[INT-MT] $
04 0628                                                                     ;JUMP IF THIS IS INTERRUPT VERSION OF INSTRUCTION.
04 0629 13234 010241170000264054161102254314560000               ALU[0] DEST[D%AMEM4] NORM PUSHJ[TRCHECK] $
04 0630                                                                 ;Wait for op. to finish and get ending status in Q.
04 0631                                                                 ; The 0 in A-MEM[4] prevents TRCHECK from clearing read buf !
04 0631
04 0632 13235 540233170003060554161623254204160000               ALU[Q] DEST[AC] SPEC[MU-POP] NEOI $
04 0633                                                                 ;Return status to caller in his AC (same as READ)
04 0634
04 0635
04 0636                                                     TWREOF:  ;WRITE AN ENDX OF FILE (TAPE MARK)
04 0637 13236 010730170000060541603623655714160000               D[CONST 1] ROT[35. - 28.] DEST[Q] NORM $
04 0638                                                                 ;Get WFM (WRITE EOF) cmd bit for formatter
04 0639 13237 010631170000000003203500255714560000       TERAS1: D[CONST 1] ROT[35. - 22.] ALU[DORQ] DEST[HOLD] NORM JUMP[KNYMTP]
04 0639                                                      $
04 0640                                                                 ;Add WRT CMD bit and start command.
04 0641
04 0642 13240 010730170000264761407620255714560000       TERASE: D[CONST 3] ROT[35. - 29.] DEST[Q] NORM JUMP[TERAS1] $
04 0643                                                                 ;ERASE A 3.75" GAP ON THE TAPE (get WFM and ERASE bits for
04 0643                                                      formatter)
04 0644
04 0611                                                      ];TAPE
04 0612
04 0613
04 0614                                                      .REPEAT TBOOT [
04 0615
04 0616                                                     KNYMTP: START-OUT NORM $
04 0617                                                             ;Give TP MR to clear mode, error status
04 0618                                                     MAPF[TP.MR] D[MEM] DEST[Q] C-OUT PUSHJ[KNYGOA] $
04 0619                                                       ;Put command bits in Q and start formatter.
04 0620                                                     START-OUT D[CONST 1] ROT[35. - 26.] DEST[IOD] NORM $
04 0621                                                             ;Clear all command bits except FORMATTER ENABLE.
04 0622                                                     MAPF[TP.WF] ALU[-1] DEST[Q] LONG POPJ $
04 0623                                                             ;Put -1 in Q in case we are returning to TAPEMT.
04 0624
04 0625                                                     TAPERW: ;REWIND
```

```
04m0629
04m0629 13241 010731170031060554161623654314160000        KNYMTP: START-OUT NORM $
04m0629                                                              ;Give TP MR to clear mode, error status
04m0629 13242 010730170500262034161622254515560000                  MAPF[TP.MR] D[MEM] DEST[Q] C-OUT PUSHJ[KNYGOA] $
04m0629                                                              ;Put command bits in Q and start formatter.
04m0629 13243 010731170031060542203523655714160000                  START-OUT D[CONST 1] ROT[35. - 26.] DEST[IOD] NORM $
04m0629                                                                      ;Clear all command bits except FORMATTER ENABLE.
04m0629 13244 010720170200060554001624255514160000                  MAPF[TP.WF] ALU[-1] DEST[Q] LONG POPJ $
04m0629                                                                      ;Put -1 in Q in case we are returning to TAPEMT.
04m0629
04m0629                                                             TAPERW: ;REWIND
04m0629 13245 010731170000026503405150025571456000                         D[CONST 24] DEST[HOLD] NORM JUMP[KNYMTP] $
04m0629                                                                      ;RWD AND GO BITS -- KNYGOA WILL DELETE THE GO BIT !
04m0629
04 0629                                                               ];TBOOT
04 0630
04 0631                                                               .REPEAT TAPE [
04 0632
04 0633                                                             TAPEFR: ;SKIP FORWARD ONE RECORD.
04 0634                                                                     D[CONST 0] DEST[HOLD] NORM JUMP[KNYMTP] $
04 0635                                                                      ;START A READ, BUT IGNORE THE DATA.
04 0636
04 0637                                                             TAPEBR: ;SKIP BACKWARD ONE RECORD.
04 0638                                                                     D[CONST 1] ROT[35. - 27.] DEST[HOLD] NORM JUMP[KNYMTP] $
04 0639                                                                      ;JUST A READ BACKWARD, WITH THE DATA IGNORED.
04 0640
04 0641                                                             TPSETIND:       ;Set industry compatable mode
04 0642                                                                     D[D%AMEM0] DEST[Q] NORM $    ;Stupid A-MEM
04 0643                                                                     D[CONST 1] ROT[35.] ALU[DORQ] DEST[D%AMEM0] JUMP[TAPEOI] $
04 0644                                                                      ;Set appropriate bit and done
04 0645                                                             TPSETDMP:       ;Set POP-10 Dump Mode
04 0646                                                                     D[D%AMEM0] DEST[Q] NORM $    ;Stupid A-MEM
04 0647                                                                     D[CONST 1] ROT[35.] ALU[-D&Q] DEST[D%AMEM0] JUMP[TAPEOI] $
04 0648                                                                      ;Clear industry compatable mode and done.
04 0649
04 0650                                                             INT-MT: ALU[0] DEST[D%AMEM4] NORM $
04 0651                                                                      ;ZERO COUNT REGISTER TO BE SAFE
04 0652                                                                     D[D%AMEM7] MASK[33.] DEST[Q] SHORT $
04 0653                                                                      ;GET READY TO SET READ BIT, MTAPE BIT, AND WAITING FOR
04 0653                                                              FORMATTER BIT.
04 0654                                                                     D[CONST 7] ROT[33.] ALU[DORQ] DEST[D%AMEM7] NORM $
04 0655                                                                      ;READ BIT SO INTERRUPT CALLS ITRCHK, MTAPE BIT SO NO LENGTH
04 0655                                                              ERROR CHK
04 0656                                                                      ; FORMATTER INTERRUPT BIT SO DOESN'T MESS AROUND WITH TRANSFER
04 0656                                                              COUNTS
04 0657                                                                     ALU[0] DEST[D%AMEM2] NORM $
04 0658                                                                      ;NOT WRITING ANY BITS TO TP.WC, SO ZERO THIS OUT.
04 0659                                                                      ;(DON'T REMEMBER REAL LAST BITS WRITTEN, SINCE A-MEM[2] IS USED
04 0659
04 0660                                                                      ; FOR BOTH REMEMBERING TP.WC VALUES AND ENDING STATUS.)
04 0661                                                                     D[CONST 2] ROT[12.] DEST[IOD] START-OUT NORM $
04 0662                                                                      ;ENABLE FOR FORMATTER NOT BUSY
04 0663                                                                     MAPF[TP.WC] SPEC[MU-POP] JUMP[MAIN] C600 $
04 0664                                                                      ;AND RETURN FROM INSTRUCTION.
04 0665
04 0666
04 0667                                                              ;OPCODE 727 -- READ STATUS BITS FROM TAPE DRIVE.
04 0668
04 0669                                                             .DEFINE TSS[A B]
04 0670                                                             [  ;MOVE BIT A OF AR TO BIT B OF Q.
04 0671                                                                     D[AR] ROT[1 + A] MASK[1] DEST[HOLD] NORM $
04 0672                                                                     D[MEM] ROT[35. - B] ALU[DORQ] DEST[Q] NORM $ ]
04 0673
04 0674
04 0675                                                             .REPEAT WAITS [
04 0676                                                                     .PAIR
04 0677                                                             ];.REPEAT WAITS
04 0678
04 0679                                                             TAPERS:
04 0680                                                             .REPEAT WAITS [
04 0681                                                                     CLR-DEV-FROM-INTR D[IR] ROT[12. + 1 + 1] MASK[4] DEST[Q]
04 0682                                                                                 COND[USER] PUSHJ[CKIOTU] $
04 0683                                                                             ;Treat device 370 and 374 as identical for now...
04 0684                                                                             ;Extract IOT decode * 2.  Note we can do this because the
04 0684
04 0685                                                                             ;machine has already done indexing/indirection and bits
04 0686                                                                             ;13:17 are guaranteed zero
04 0687                                                                     D[LABEL TAPE-DISPATCH] ALU[D+Q] DEST[AR] NORM $
04 0688                                                                     D[AR] ROT[MUA-ROT] ODISP LONG $
04 0689                                                                             ;Dispatch of type of IOT
04 0690
04 0691                                                             ;Allocate space for dispatch table
04 0692                                                                     .PAIR   ;Limit address to 12 bits in width
04 0693                                                             TAPE-DISPATCH:
04 0694                                                             :. + 20
04 0695                                                                     .ORG [TAPE-DISPATCH]
04 0696                                                             ];.REPEAT WAITS
04 0697                                                             .REPEAT 1 - WAITS [
04 0698                                                                     UIOTRP[MUUO] $
04 0699                                                             ];.REPEAT 1 - WAITS
04 0700
04 0701                                                                     START-IN NORM $
04 0702                                                                      ;NEED TO CHECK FOR FORMATTER BUSY
04 0703                                                                     MAPF[TP.RS] D[IOD] ALU[NOTD] DEST[AR] C800
04 0704                                                                      ;GET STATUS BITS
04 0705                                                             ;Only the .RELOC really needs to be under WAITS conditional.
04 0706                                                             .REPEAT 1 - WAITS [ $ ]
04 0707                                                             .REPEAT WAITS [
04 0708                                                                     JUMP[TAPERS0] $
04 0709                                                                      ;GET STATUS BITS
04 0710                                                                 .RELOC
04 0711                                                             TAPERS0:
04 0712                                                             ];.REPEAT WAITS
04 0713                                                                     D[AR] ROT[7] OBUS<0 C550 JUMP[TAPERS1] $
04 0714                                                                      ;JUMP IF FORMTTER STILL BUSY, DON'T TOUCH AND RUIN XFER.
04 0715                                                                      ;JUST USE THESE BITS.
04 0716
04 0717                                                                     START-OUT ALU[0] DEST[IOD] NORM $
04 0718                                                                      ;CLEAR THE MODE CTRL REGISTER.
04 0719                                                                     MAPF[TP.WM] START-OUT
```

```
04 0725                                                 ;NOW RE-ARRANGE THE BITS
04 0726                                         TAPERS1:
04 0727                                                 TSS[ 3 30. ]    ;ON LINE
04 0728                                                 TSS[ 11. 31. ]  ;REWINDING
04 0729                                                 TSS[ 4 32. ]    ;FILE PROTECT
04 0730                                                 TSS[ 12. 33. ]  ;LOAD POINT
04 0731                                                 TSS[ 10. 34. ]  ;READY
04 0732                                                 TSS[ 34. 35. ]  ;ENDX OF TAPE
04 0733                                                 ALU[Q] SMAC $
04 0734
04 0735
04 0736                                            ;725 - OBSOLETE VERSION OF READ
04 0737
04 0738                                         TAPERD:
04 0739                                         .repeat 1 - waitsfix [
04 0740                                                 UIOTRP[MUUO] $
04 0741                                                 D[D%AMEM7] MASK[33.] DEST[Q] SHORT $
04 0742                                         ];.repeat 1 - waitsfix
04 0743                                         .repeat waitsfix [
04 0744                                                 D[D%AMEM7] MASK[33.] DEST[Q] NORM COND[USER] PUSHJ[CKIOTU] $
04 0745                                                         ;Make sure we're in Exec or IOT-user mode
04 0746                                         ];.repeat waitsfix
04 0747                                                         ;CLEAR OUT READ, MTAPE AND INTERRUPT FLAGS.
04 0748                                                 ALU[Q] DEST[D%AMEM7] NORM $
04 0749                                                 D[CONST 14] ROT[6] DEST[IR-ADR] NORM PUSHJ[KNYRGO] $
04 0750                                                         ;Fake a word count of 1400
04 0751                                                 D[MA] DEST[D%AMEM4] NORM PUSHJ[TRPZ] $
04 0752                                                 ALU[Q] DEST[AC] ACSEL[AC] NEOI $
04 0753                                                         ;Move status into AC.
04 0754
04 0755                                         ;730, AC/COUNT.  READ WHOLE RECORD, STORING UP TO COUNT WORDS STARTING AT
04 0755                                          EFF ADR.
04 0756                                                 ;SWAP AC AND IR, THEN DO TAPERX.
04 0757
04 0758                                         TAPENR:
04 0759                                         .REPEAT 1 - WAITS [
04 0760                                                 UIOTRP[MUUO] $
04 0761                                         ];.REPEAT 1 - WAITS
04 0762                                                 D[IR] DEST[0-AC IR-ADR] ACSEL[AC]  NORM JUMP[TAPERX] $
04 0763
04 0764
04 0765                                         ;732 - (AC) IS START ADDR., E IS # OF WORDS TO READ.
04 0766
04 0767                                         TAPERX:
04 0768                                         .repeat 1 - waitsfix [
04 0769                                                 UIOTRP[MUUO] $
04 0770                                                 D[D%AMEM7] MASK[33.] DEST[Q] SHORT $
04 0771                                         ];.repeat 1 - waitsfix
04 0772                                         .repeat waitsfix [
04 0773                                                 D[D%AMEM7] MASK[33.] DEST[Q] NORM COND[USER] PUSHJ[CKIOTU] $
04 0774                                                         ;Make sure we're in Exec or IOT-user mode
04 0775                                         ];.repeat waitsfix
04 0776                                                         ;CLEAR ALL FLAG BITS IN A-MEM[7]
04 0777                                                 ALU[Q] DEST[D%AMEM7] NORM PUSHJ[KNYRGO] $ ;GET TAPE STARTED.
04 0778                                                 D[D%AMEM7] MASK[3] -OBUS=0 CS50 JUMP[INT-READ] $
04 0779                                                         ;JUMP IF WE ARE TO DO THE INTERRUPT FLAVOR OF
04 0779                                          INSTRUCTION.
04 0780                                                 ALU[AC] ACSEL[AC] DEST[D%AMEM4] PUSHJ[TRPZ] NORM $ ;READ REC.
04 0781                                                 D[AR] ROT[16.] -OBUS<0 JUMP[TNRP3] $
04 0782                                                         ;Was record longer than word count ?
04 0783                                                         ;Jump unless FIFO RDY FOR BUF is on, indicating
04 0784                                                         ;  that tape supplied more bytes after count ran out.
04 0785                                                 D[LIT 440000000000] ALU[DORQ] DEST[Q] NORM $ ;YES, SET BIT 3.
04 0786                                         TNRP3:  ALU[Q] DEST[AC] ACSEL[AC] SPEC[MU-POP] NEOI $  ;NO, NOT TOO LONG.
04 0786
04 0787                                                         ;MOVE STATUS INTO AC.
04 0788
04 0789                                         ;HERE FOR INTERRUPT FLAVOR OF TAPE READ INSTRUCTIONS, WHERE PI ASSIGNMENT
04 0789
04 0790                                         ; IS NON-ZERO.
04 0791
04 0792                                         INT-READ:
04 0793                                                 D[D%AMEM7] MASK[34.] DEST[Q] NORM $
04 0794                                                         ;GET PLACE WHERE TO SET READ FLAG, CLEAR INTERRUPT STATE
04 0794                                          FLAG
04 0795                                                         ; IN BIT 1.
04 0796                                                 D[CONST 1] ROT[35.] ALU[DORQ] DEST[D%AMEM7] JUMP[INT-WRITE] NORM
04 0796                                          $
04 0797                                                         ;SET READ FLAG IN A-MEM[7].
04 0798                                                         ;REST OF STUFF SAME AS FOR WRITE.
04 0799
04 0800
04 0801                                          ;731, AC/ADR, E/+COUNT.  WRITE RECORD OF +COUNT
04 0802                                                 ; WORDS, DATA FROM ADR.
04 0803                                                 ;SET AC:=0 IF OPERATION COMPLETED SUCCESSFULLY.
04 0804                                                 ; SET AC:=<SETZ> + HIGHEST ADR READ IF REACHED
04 0805                                                 ; EOT DURING OPERATION (CURRENTLY THIS IS THE ONLY
04 0806                                                 ; ERROR CONDITION.)  OPERATION IS COMPLETED EVEN
04 0807                                                 ; IF EOT IS PASSED.
04 0808
04 0809
04 0810                                         TAPEWR:
04 0811                                         .repeat 1 - waitsfix [
04 0812                                                 UIOTRP[MUUO] $
04 0813                                         ];.repeat 1 - waitsfix
04 0814                                                 D[D%AMEM7] MASK[33.] DEST[Q] NORM
04 0815                                         .repeat waitsfix [
04 0816                                                         COND[USER] PUSHJ[CKIOTU]
04 0817                                         ];.repeat waitsfix
04 0818                                         $               ;Setup to clear the read, mtape, and interrupt state
04 0818                                          flags.
04 0819                                                         ;Make sure we're in Exec or IOT-user mode
04 0820                                                 ALU[Q] DEST[D%AMEM7] NORM PUSHJ[DCINIT] $
04 0821                                                         ;Init. the data channel.
04 0822                                                 D[D%AMEM0]  ROT[1] MASK[1] DEST[AR] NORM $
04 0823                                                         ;Get the 32-bit mode flag.
04 0824                                                 D[AR] ROT[35. - 29.] DEST[Q] NORM $
04 0825                                                         ;Position it for the hardware.
04 0826                                                 START-OUT D[CONST 1] ROT[35. - 28.] ALU[DORQ] DEST[IOD] NORM $
04 0827                                                         ;Set the OUT bit.
04 0828                                                 MAPF[IP WM] D[CONST 1] ROT[35. - 22.] DEST[Q] PUSHJ[KNYGO0] $
```

```
04 0832                                                         ; instruction.
04 0833                                                 ALU[AC] DEST[D%AMEM4] NORM PUSHJ[STRTDC] $
04 0834                                                         ;Get word count and start the channel.
04 0835                                                 D[MEM] ROT[32.] -OBUS<0 JUMP[TWERR] $ ;Test HARD ERR.
04 0836                                                 ALU[0] DEST[AC] NORM $ ;We return 0 in AC if no EOT.
04 0837                                                 D[MEM] ROT[34.] C550 OBUS<0 JUMP[TAPEOI] $
04 0838                                                  ;If no EOT seen, all done.
04 0839                                                 D[CONST 60] ROT[35. - 5] ALU[DORAC] DEST[AC] NEOI $
04 0840                                                  ;Turn on bit 0 to indicate EOT seen during operation.
04 0841
04 0842                                         ;Here for the case where PI assignment is non-zero - just start the
04 0843                                         ; transfer and return. Also come here at ENDX of INT-READ.
04 0844
04 0845                                         INT-WRITE:
04 0846                                                 ALU[AC] DEST[D%AMEM4] PUSHJ[INT-STARTDC] NORM $
04 0847                                                   ;PUT COUNT IN AMEM[4], START THE TRANSFER
04 0848                                                 NEOI $
04 0849                                                   ;AND RETURN FROM THIS INSTRUCTION WITHOUT CHANGING AC.
04 0850
04 0851
04 0852
04 0853                                         TWERR:  ;;Error occurred during write.
04 0854
04 0855                                                 NORM PUSHJ[KNYRS1] $
04 0856                                                   ;Blast the tape formatter and drive
04 0857                                                 D[CONST 50] ROT[30.] DEST[AC] NEOI $
04 0858                                                   ;And give error return to program. We no longer
04 0859                                                   ;do backspace erase gap rewrite at micro-level, MACRO level
04 0860                                                   ; must do this.
04 0861
04 0862
```

```
05 0863
05 0864
05 0865
05 0866                                 TAPINT: START-OUT D[D%AMEM2] DEST[IOD] NORM $
05 0867                                         MAPF[TP.WC] START-IN CB00 $
05 0868                                           ;CLEAR ENABLES
05 0869                                         MAPF[TP.RS] D[IOD] DEST[AR] CB00 $
05 0870                                           ;GET STATUS IN AR FOR CHECKING.
05 0871                                         D[D%AMEM7] ROT[1] OBUS<0 CS50 JUMP[FMTDON] $
05 0872                                           ;JUMP IF EXPECTING A -FMT BUSY INTERRUPT
05 0873                                         D[D%AMEM7] -OBUS<0 CS50 JUMP[TAPINTWRT] $
05 0874                                           ;JUMP IF THIS XFER IS A WRITE.
05 0875                                         START-OUT $
05 0876                                           ;THE MAPF[0] WRITES LAST WORD TO MEMORY.
05 0877                                         MAPF[0] D[AR] ROT[7] OBUS<0 CS50 JUMP[INT-RDDONE] $
05 0878                                           ;IF FORMATTER NO LONGER BUSY, THEN WE READ RECORD SHORTER
05 0879                                           ; THAN THE COUNT, SO FORGET REST OF TRANSFER.
05 0880
05 0881                                 ;HERE ALWAYS IF WRITE OR IF READ AND FORMATTER IS STILL GOING.
05 0882                                 ; MAY EITHER BE ENDX OF TRANSFER OR TIME TO CONTINUE CURRENT LONG RECORD.
05 0882
05 0883                                 ; TAPE COUNT GO HAS TO BE OFF SINCE ITS EITHER WRITE, OR READ WITH
05 0883                                  FORMATTER
05 0884                                 ; IDLE.
05 0885
05 0886                                 TAPINTWRT:
05 0887                                         D[D%AMEM5] OBUS=0 CS50 JUMP[TAPEXFRDON] $
05 0888                                           ;IF WE HAVE EXHAUSED COUNT, TRANSFER IS DONE.
05 0889                                           ; COME BACK TO FMTDON WHEN -FMT BUSY COMES TRUE.
05 0890                                         PUSHJ[DCGO] NORM $
05 0891                                           ;START UP NEXT PART OF THE TRANSFER
05 0892                                         MAPF[0] D[CONST 12] ROT[12.] DEST[Q] JUMP[TAPINTENB] NORM $
05 0893                                           ;MAPF 0 for DCGO. Enable for -FMTR BSY and -TP CNT GO.
05 0894                                 TAPEXFRDON:
05 0895                                         D[D%AMEM7] DEST[Q] SHORT $
05 0896                                           ;GET DATA TO OR IN WAITING FOR FORMAT DONE BIT
05 0897                                         D[CONST 1] ROT[35. - 1] ALU[DORQ] DEST[D%AMEM7] NORM $
05 0898                                           ;WE CONTINUE AT FMTDON ON NEXT INTERRUPT.
05 0899                                         D[CONST 02] ROT[12.] DEST[Q] NORM JUMP[TAPINTENB] $
05 0900                                           ;BITS FOR JUST -FMT BUSY INTERRUPT ENABLE
05 0901
05 0902                                 TAPINTENB:;HERE WITH BITS FOR INTERRUPT ENABLES IN Q.
05 0903                                         D[D%AMEM2] ALU[DORQ] DEST[IOD] START-OUT NORM $
05 0904                                           ;ENABLE FOR BITS IN Q.
05 0905                                         MAPF[TP.WC] CB00
05 0906                                          JUMP[MAIN]  $
05 0907                                           ;AND RETURN FROM THIS MICROINTERRUPT.
05 0908
05 0909                                 FMTDON: D[AR] ROT[7] -OBUS<0 CS50 JUMP[.] $
05 0910                                           ;HANG IF FORMATTER IS STILL BUSY.
05 0911                                         D[D%AMEM7] -OBUS<0 CS50 JUMP[INT-WRTDONE] $
05 0912                                           ;JUMP IF ITS NOT A READ.
05 0913
05 0914                                 ;HERE ALSO IF FORMATTER SHUT DOWN BECAUSE NOT ENOUGH DATA ON TAPE TO
05 0914                                  SATISFY
05 0915                                 ; THE CHANNEL'S COUNT.
05 0916
05 0917                                 INT-RDDONE:
05 0918                                         PUSHJ[ITRCHK] NORM $
05 0919                                           ;GET STATUS OF XFER INTO Q
05 0920                                         D[D%AMEM7] ROT[2] OBUS<0 JUMP[INT-RDWCOK] CS50 $
05 0921                                           ;IF NON-DATA OPERATION, DON'T CHECK FOR WORD COUNT.
05 0922                                         D[AR] ROT[16.] -OBUS<0 JUMP[INT-RDWCOK] CS50 $
05 0923                                           ;JUMP IF NO PROBLEM WITH WORD COUNT
05 0924                                         D[CONST 44] ROT[30.] ALU[DORQ] DEST[Q] NORM $
05 0925                                           ;RECORD WAS LONGER THAN WORD COUNT, SET LENGTH ERROR.
05 0926                                 INT-RDWCOK:
05 0927                                         ALU[Q] DEST[D%AMEM2] NORM $
05 0928                                           ;STORE STATUS IN AMEM[2] FOR THE TRANSFER STATUS READING INST.
05 0928
05 0929                                 TAPEPI: D[D%AMEM7] DEST[Q] SHORT $
05 0930                                           ;GET CONI WORD.
05 0931                                         D[CONST 10] ALU[DORQ] DEST[D%AMEM7] NORM $
05 0932                                           ;SET INTERRUPT BIT FOR CONSO
05 0933                                         D[D%AMEM7] DEST[Q AR] MASK[3] -OBUS=0 JUMP[PIGEN] CS50 $
05 0934                                           ;IF PI CHANNEL ASSIGNED, GO INTERRUPT.
05 0935                                         NEOI $
05 0936                                           ;CLEARED PI ASSIGNMENT OUT DURING XFER?
05 0937
05 0938                                 INT-WRTDONE:
05 0939                                         ALU[0] DEST[D%AMEM2 Q] NORM $
05 0940                                           ;CLEAR OUT ACCUMULATED STATUS
05 0941                                         D[AR] ROT[32.] OBUS<0 JUMP[. + 2] CS50 $
05 0942                                           ;JUMP IF NO HARD ERROR
05 0943                                         D[CONST 5] ROT[33.] ALU[DORQ] DEST[D%AMEM2 Q] PUSHJ[KNYRS1] $
05 0944                                           ;HARD ERROR, SET 180 FOR ANY ERROR, 182 FOR HARD WRITE ERROR
05 0945                                         D[AR] ROT[34.] OBUS<0 CS50 JUMP[TAPEPI] $
05 0946                                           ;IF EOT NOT SET, JUST CAUSE PI ON PROPER CHANNEL.
05 0947                                         D[CONST 6] ROT[33.] ALU[DORQ] DEST[D%AMEM2] JUMP[TAPEPI] NORM $
05 0948                                           ;EOT, SET IT IN AMEM AND GO CAUSE PI
05 0949
05 0950
```

```
06 0951
06 0952
06 0953
06 0954                                 ;TPCONI INSTRUCTION, OPCODE 733.
06 0955                                 ; 10 MEANS INTERRUPT HAS BEEN REQUESTED.
06 0956                                 ; 33-35 PI ASSIGNMENT.
06 0957
06 0958                                 .REPEAT WAITS [
06 0959                                         .ORG[TAPE-DISPATCH + 12]
06 0960                                 ];.REPEAT WAITS
06 0961                                 TPCONI:
06 0962                                 .REPEAT 1 - WAITS [
06 0963                                         UIOTRP[MUUO] $
06 0964                                 ];.REPEAT 1 - WAITS
06 0965
06 0966                                         START-IN D[CONST 6] ROT[15.] DEST[Q] PUSHJ[TPCONIBTS] NORM $
06 0967                                           ;GET THE HI ORDER CONI BITS
06 0968                                         D[D%AMEM7] MASK[15.] ALU[DORQ] SMAC $
06 0969                                           ;STORE CONI BITS INTO EA.
06 0970                                           ;ALSO STORE INTERRUPT STATE BITS FOR DEBUGGING.
06 0971
06 0972                                 ;CAUTION:  Still in dispatch table under WAITS switch
06 0973
06 0974                                 .REPEAT WAITS [
06 0975                                 TPCONSZ: START-IN D[CONST 6] ROT[15.] DEST[Q] PUSHJ[TPCONIBTS] NORM $
06 0976                                           ;GET THE HI ORDER CONI BITS
06 0977                                         D[D%AMEM7] MASK[15.] ALU[DORQ] DEST[Q] NORM JUMP[XXCONSZ] $
06 0978                                           ;STORE CONI BITS INTO Q AND DO USUAL CONSZ THING
06 0979                                 TPCONSO: START-IN D[CONST 6] ROT[15.] DEST[Q] PUSHJ[TPCONIBTS] NORM $
06 0980                                           ;GET THE HI ORDER CONI BITS
06 0981                                         D[D%AMEM7] MASK[15.] ALU[DORQ] DEST[Q] NORM JUMP[XXCONSO] $
06 0982                                           ;STORE CONI BITS INTO Q AND DO USUAL CONSO THING
06 0983
06 0984                                         .RELOC
06 0985                                 ];.REPEAT WAITS
06 0986
06 0987                                 ;TPCONO INSTRUCTION, OPCODE 734.
06 0988                                 ; 100 MEANS CLEAR FORMATTER.
06 0989                                 ; 20 MEANS CLEAR INTERRUPT FLAG.
06 0990                                 ; 33-35 ARE TO SET PI ASSIGNMENT.
06 0991
06 0992                                 .REPEAT WAITS [
06 0993                                         .ORG[TAPE-DISPATCH + 10]
06 0994                                 ];.REPEAT WAITS
06 0995                                 .REPEAT 1 - WAITS [
06 0996                                 TPCONO: UIOTRP[MUUO] $
06 0997                                 ];.REPEAT 1 - WAITS
06 0998
06 0999                                         D[IR] ROT[29.] OBUS<0 C550 PUSHJ[TAPCLR] $
06 1000                                           ;IF 100 BIT IS SET, CLEAR THE FORMATTER.
06 1001                                 .REPEAT 1 - WAITS [
06 1002                                 ;Only the .reloc really needs to be under WAITS switch.  The JUMP and
06 1002                                  NORM
06 1003                                 ;are done separately to make BINCOM succeed.
06 1004                                         D[CONST 7] ALU[NOTD] DEST[Q] SHORT $
06 1005                                           ;MASK TO CLEAR PI ASSIGNMENT
06 1006                                 ];.REPEAT 1 - WAITS
06 1007                                 .REPEAT WAITS  [
06 1008                                         D[CONST 7] ALU[NOTD] DEST[Q] JUMP[TPCONO2] NORM $
06 1009                                           ;MASK TO CLEAR PI ASSIGNMENT
06 1010                                     .reloc
06 1011                                 TPCONO2:
06 1012                                 ];.REPEAT WAITS
06 1013                                         D[IR] ROT[31.] -OBUS<0 C550 JUMP[. + 2] $
06 1014                                           ;JUMP IF NOT CLEARING IT
06 1015                                         D[CONST 17] ALU[NOTD] DEST[Q] SHORT $
06 1016                                           ;CLEARING INTERRUPT FLAG, GET DIFFERENT MASK
06 1017                                         D[D%AMEM7] ALU[D&Q] DEST[Q] SHORT $
06 1018                                           ;CLEAR OUT PI ASSIGNMENT AND MAYBE INTERRUPT FLAG
06 1019                                         D[IR] MASK[3] ALU[DORQ] DESTID%AMEM7] NORM JUMP[TAPEOI] $
06 1020                                           ;STORE PI ASSIGNMENT IN A-MEM[7] AND RETURN FROM INSTRUCTION.
06 1021
06 1022                                 TAPCLR: PUSHJ[DCINIT] NORM $
06 1023                                           ;GET RID OF INTERRUPT ENABLES, RESET CHANNEL
06 1024                                         JUMP[KNYCLR] NORM $
06 1025                                           ;AND CLEAR FORMATTER AND DRIVE.
06 1026
06 1027                                 ;TPCNSO - OPCODE 735.
06 1028                                 ; SKIP IF BITS IN EA ARE SET.
06 1029                                 ; SEE TPCONI FOR BIT ASSIGNMENTS.
06 1030
06 1031
06 1032                                 .REPEAT 1 - WAITS [
06 1033
06 1034                                 TPCNSO: UIOTRP[MUUO] $
06 1035                                         START-IN D[CONST 6] ROT[15.] DEST[Q] PUSHJ[TPCONIBTS] NORM $
06 1036                                           ;GET THE HI ORDER BITS INTO Q.
06 1037                                         D[D%AMEM7] MASK[18.] ALU[DORQ] DEST[Q] SHORT $
06 1038                                           ;GET WHAT BITS ARE.
06 1039                                         D[MA] ALU[D&Q] CONDSKP[-ALU=0] $
06 1040                                           ;IF ANY MASKED BITS ON => SKIP
06 1041
06 1042                                 ;LEAVE OUT TPCNSZ, DON'T WASTE THE SPACE.
06 1043
06 1044                                 ];.REPEAT 1 - WAITS  (includes comment because WAITS includes CONSZ)
06 1045
06 1046                                 ;TPDATI - OPCODE 736
06 1047                                 ;RETURNS TRANSFER STATUS (BITS, ENDING ADDRESS) TO C(EA).
06 1048
06 1049                                 .REPEAT WAITS [
06 1050                                         .ORG[TAPE-DISPATCH + 2]
06 1051                                 ];.REPEAT WAITS
06 1052                                 .REPEAT 1 - WAITS [
06 1053                                 TPDATI: UIOTRP[MUUO] $
06 1054                                 ];.REPEAT 1 - WAITS
06 1055
06 1056                                         D[D%AMEM2] SMAC $
06 1057                                           ;STORE STATUS SAVED ON LAST XFER TO C(EA)
06 1058                                 .REPEAT WAITS [
06 1059                                         NOP $
```

```
06 1065                                                 .RELOC
06 1066                                         ];REPEAT WAITS
06 1067
06 1068
06 1069                                         TPCONIBTS:
06 1070                                                 MAPF[TP.RC] D[IOD] ALU[D&Q] DEST[Q] C800 START-IN $
06 1071                                                   ;1B10 = OVERRUN, CLEARED ON MASTER RESET.
06 1072                                                   ;1B19 = CONTROLLER BUSY.
06 1073                                                 MAPF[TP.RS] D[IOD] DEST[AR] C800 $
06 1074                                                   ;GET FORMATTER STATUS IN AR
06 1075                                                 D[AR] ROT[7] OBUS<0 C550 POPJ $
06 1076                                                   ;RETURN IF FORMATTER IS IDLE.
06 1077                                                 D[CONST 1] ROT[35. - 20.] ALU[DORQ] DEST[Q] NORM POPJ $
06 1078                                                   ;FORMATTER IS BUSY, SET THE BIT.
06 1079                                                   ;1B20 = FORMATTER BUSY
06 1080
06m1081                                                   ] [
06m1081
06m1081                                         TAPEFR: ;SKIP FORWARD ONE RECORD.
06m1081 13246 010731170000265034001500255714560000             D[CONST 0] DEST[HOLD] NORM JUMP[KNYMTP] $
06m1081                                                   ;START A READ, BUT IGNORE THE DATA.
06m1081
06m1081                                         TAPEBR: ;SKIP BACKWARD ONE RECORD.
06m1081 13247 010731170000265022003500255714560000             D[CONST 1] ROT[35. - 27.] DEST[HOLD] NORM JUMP[KNYMTP] $
06m1081                                                   ;JUST A READ BACKWARD, WITH THE DATA IGNORED.
06m1081
06m1081                                         TPSETIND:       ;Set industry compatable mode
06m1081 13250 010730170000060554161623660314160000             D[D%AMEM0] DEST[Q] NORM $   ;Stupid A-MEM
06m1081 13251 010631170000261610603000255714560000             D[CONST 1] ROT[35.] ALU[DORQ] DEST[D%AMEM0] JUMP[TAPEOI] $
06m1081                                                   ;Set appropriate bit and done
06m1081                                         TPSETDMP:       ;Set PDP-10 Dump Mode
06m1081 13252 010730170000060554161623660314160000             D[D%AMEM0] DEST[Q] NORM $   ;Stupid A-MEM
06m1081 13253 010651170000261610603000255714560000             D[CONST 1] ROT[35.] ALU[-D&Q] DEST[D%AMEM0] JUMP[TAPEOI] $
06m1081                                                   ;Clear industry compatable mode and done.
06m1081
06m1081 13254 010241170000060554161103654314160000     INT-MT: ALU[0] DEST[D%AMEM4] NORM $
06m1081                                                   ;ZERO COUNT REGISTER TO BE SAFE
06m1081 13255 010730170000060554103623663774160000             D[D%AMEM7] MASK[33.] DEST[Q] SHORT $
06m1081                                                   ;GET READY TO SET READ BIT, MTAPE BIT, AND WAITING FOR
06m1081                                          FORMATTER BIT.
06m1081 13256 010631170000060550217163655714160000             D[CONST 7] ROT[33.] ALU[DORQ] DEST[D%AMEM7] NORM $
06m1081                                                   ;READ BIT SO INTERRUPT CALLS ITRCHK, MTAPE BIT SO NO LENGTH
06m1081                                          ERROR CHK
06m1081                                                   ; FORMATTER INTERRUPT BIT SO DOESN'T MESS AROUND WITH TRANSFER
06m1081                                          COUNTS
06m1081 13257 010241170000060554161043654314160000             ALU[0] DEST[D%AMEM2] NORM $
06m1081                                                   ;NOT WRITING ANY BITS TO TP.WC, SO ZERO THIS OUT.
06m1081                                                   ;(DON'T REMEMBER REAL LAST BITS WRITTEN, SINCE A-MEM[2] IS USED
06m1081
06m1081                                                   ; FOR BOTH REMEMBERING TP.WC VALUES AND ENDING STATUS.)
06m1081 13260 010731170031060543005523655714160000             D[CONST 2] ROT[12.] DEST[IOD] START-OUT NORM $
06m1081                                                   ;ENABLE FOR FORMATTER NOT BUSY
06m1081 13261 010731170103227114161620254114560000             MAPF[TP.WC] SPEC[MU-POP] JUMP[MAIN] C600 $
06m1081                                                   ;AND RETURN FROM INSTRUCTION.
06m1081
06m1081
06m1081                                          ;OPCODE 727 -- READ STATUS BITS FROM TAPE DRIVE.
06m1081
06m1081                                         .DEFINE TSS[A B]
06m1081                                         [  ;MOVE BIT A OF AR TO BIT B OF Q.
06m1081                                                 D[AR] ROT[1 + A] MASK[1] DEST[HOLD] NORM $
06m1081                                                 D[MEM] ROT[35. - B] ALU[DORQ] DEST[Q] NORM $ ]
06m1081
06m1081
06m1081                                         .REPEAT WAITS [
06m1081                                                 .PAIR
06m1081                                         ];[
06m1081                                                 .PAIR
06m1081                                          [:. \ 2 + .
06m1081                                            ]].REPEAT WAITS
06 1081
06 1082
06 1083                                         TAPERS:
06 1084                                         .REPEAT WAITS [
06 1085                                                 CLR-DEV-FROM-INTR D[IR] ROT[12. + 1 + 1] MASK[4] DEST[Q]
06 1086                                                                 COND[USER] PUSHJ[CKIOTU] $
06 1087                                                         ;Treat device 370 and 374 as identical for now...
06 1088                                                         ;Extract IOT decode * 2.  Note we can do this because the
06 1088
06 1089                                                         ;machine has already done indexing/indirection and bits
06 1090                                                         ;13:17 are guaranteed zero
06 1091                                                 D[LABEL TAPE-DISPATCH] ALU[D+Q] DEST[AR] NORM $
06 1092                                                 D[AR] ROT[MUA-ROT] ODISP LONG $
06 1093                                                         ;Dispatch of type of IOT
06 1094
06 1095                                         ;Allocate space for dispatch table
06 1096                                                 .PAIR   ;Limit address to 12 bits in width
06 1097                                         TAPE-DISPATCH:
06 1098                                         :. + 20
06 1099                                                 .ORG [TAPE-DISPATCH]
06m1100                                         ];[
06m1100                                                 CLR-DEV-FROM-INTR D[IR] ROT[12. + 1 + 1] MASK[4] DEST[Q]
06m1100 13262 010730124040000000341162225771456000             COND[USER] PUSHJ[CKIOTU] $
06m1100                                                         ;Treat device 370 and 374 as identical for now...
06m1100                                                         ;Extract IOT decode * 2.  Note we can do this because the
06m1100
06m1100                                                         ;machine has already done indexing/indirection and bits
06m1100                                                         ;13:17 are guaranteed zero
06m1100 13263 010601370000000000000162365531416000     D[LABEL TAPE-DISPATCH] ALU[D+Q] DEST[AR] NORM $
06m1100 13264 010731170000000000160007254114560000             D[AR] ROT[MUA-ROT] ODISP LONG $
06m1100                                                         ;Dispatch of type of IOT
06m1100
06m1100                                         ;Allocate space for dispatch table
06m1100                                                 .PAIR    [:. \ 2 + .
06m1100                                           ];Limit address to 12 bits in width
06m1100                                         TAPE-DISPATCH:
06m1100                                         :. + 20
06m1100                                                 .ORG [TAPE-DISPATCH]
06m1100                                          [ xlist
06m1100                                            list ]].REPEAT WAITS
06 1100                                         .REPEAT 1 - WAITS [
06 1101
```

```
06 1107                                                         MAPF[TP.RS] D[IOD] ALU[NOTD] DEST[AR] C800
06 1108                                                           ;GET STATUS BITS
06 1109                                                 ;Only the .RELOC really needs to be under WAITS conditional.
06 1110                                                 .REPEAT 1 - WAITS [ $ ]
06 1111                                                 .REPEAT WAITS [
06 1112                                                         JUMP[TAPERS0] $
06 1113                                                           ;GET STATUS BITS
06 1114                                                     .RELOC
06 1115                                                 TAPERS0:
06 1116                                                 ];[
06m1116 13267 010771370400000001416162025711456000             JUMP[TAPERS0] $
06m1116                                                           ;GET STATUS BITS
06m1116                                                     .RELOC
06m1116                                                  [.USE[HILOC]
06m1116                                                  [ xlist
06m1117                                                    list ]                                       ]TAPERS0:
06 1116                                                 ].REPEAT WAITS
06 1117 13306 010731004000000000176162025421456000             D[AR] ROT[7] OBUS<0 CS50 JUMP[TAPERS1] $
06 1118                                                           ;JUMP IF FORMTTER STILL BUSY, DON'T TOUCH AND RUIN XFER.
06 1119                                                           ;JUST USE THESE BITS.
06 1120
06 1121 13307 010241170031060554161523654314116000             START-OUT ALU[0] DEST[IOD] NORM $
06 1122                                                           ;CLEAR THE MODE CTRL REGISTER.
06 1123                                                         MAPF[TP.WM] START-OUT
06 1124 13310 010731170431060542203523655514116000              D[CONST 1] ROT[35. - 26.] DEST[IOD] C-OUT $
06 1125                                                           ;ENABLE THE FORMATTER.
06 1126 13311 010240170230060554161623654114116000             MAPF[TP.WF] START-IN ALU[0] DEST[Q] C800 $
06 1127                                                           ;READ STATUS BITS.
06 1128 13312 010771370400060554161623657114116000             MAPF[TP.RS] D[IOD] ALU[NOTD] DEST[AR] C800 $
06 1129                                                           ;NOW RE-ARRANGE THE BITS
06 1130                                                 TAPERS1:
06m1131                                                         TSS[ 3 30. ]
06m1131                                                 [  ;MOVE BIT 3 OF AR TO BIT 30. OF Q.
06m1131 13313 010731170000060541003503654314116000             D[AR] ROT[1 + 3] MASK[1] DEST[HOLD] NORM $
06 1131 13314 010630170000060541361623654715160000             D[MEM] ROT[35. - 30.] ALU[DORQ] DEST[Q] NORM $ ];ON LINE
06m1132                                                         TSS[ 11. 31. ]
06m1132                                                 [  ;MOVE BIT 11. OF AR TO BIT 31. OF Q.
06m1132 13315 010731170000060543003503654314116000             D[AR] ROT[1 + 11.] MASK[1] DEST[HOLD] NORM $
06 1132 13316 010630170000060541161623654715160000             D[MEM] ROT[35. - 31.] ALU[DORQ] DEST[Q] NORM $ ];REWINDING
06m1133                                                         TSS[ 4 32. ]
06m1133                                                 [  ;MOVE BIT 4 OF AR TO BIT 32. OF Q.
06m1133 13317 010731170000060541203503654314116000             D[AR] ROT[1 + 4] MASK[1] DEST[HOLD] NORM $
06 1133 13320 010630170000060540761623654715160000             D[MEM] ROT[35. - 32.] ALU[DORQ] DEST[Q] NORM $ ];FILE PROTECT
06m1134                                                         TSS[ 12. 33. ]
06m1134                                                 [  ;MOVE BIT 12. OF AR TO BIT 33. OF Q.
06m1134 13321 010731170000060543203503654314116000             D[AR] ROT[1 + 12.] MASK[1] DEST[HOLD] NORM $
06 1134 13322 010630170000060540561623654715160000             D[MEM] ROT[35. - 33.] ALU[DORQ] DEST[Q] NORM $ ];LOAD POINT
06m1135                                                         TSS[ 10. 34. ]
06m1135                                                 [  ;MOVE BIT 10. OF AR TO BIT 34. OF Q.
06m1135 13323 010731170000060542603503654314116000             D[AR] ROT[1 + 10.] MASK[1] DEST[HOLD] NORM $
06 1135 13324 010630170000060540361623654715160000             D[MEM] ROT[35. - 34.] ALU[DORQ] DEST[Q] NORM $ ];READY
06m1136                                                         TSS[ 34. 35. ]
06m1136                                                 [  ;MOVE BIT 34. OF AR TO BIT 35. OF Q.
06m1136 13325 010731170000060550603503654314116000             D[AR] ROT[1 + 34.] MASK[1] DEST[HOLD] NORM $
06 1136 13326 010630170000060540161623654715160000             D[MEM] ROT[35. - 35.] ALU[DORQ] DEST[Q] NORM $ ];ENDX OF TAPE
06 1137 13327 010231310000226654161561254304560000             ALU[Q] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<20] LBJUMP[SEQI] ]$
06 1138
06 1139
06 1140                                                    ;72S - OBSOLETE VERSION OF READ
06 1141
06 1142                                                 TAPERD:
06 1143                                                 .repeat 1 - waitsfix [
06 1144                                                         UIOTRP[MUUO] $
06 1145                                                         D[D%AMEM7] MASK[33.] DEST[Q] SHORT $
06 1146                                                 ];.repeat 1 - waitsfix
06 1147                                                 .repeat waitsfix [
06 1148                                                         D[D%AMEM7] MASK[33.] DEST[Q] NORM COND[USER] PUSHJ[CKIOTU] $
06 1149                                                                 ;Make sure we're in Exec or IOT-user mode
06m1150                                                 ];[
06m1150 13330 010730124000000001410362226371456000             D[D%AMEM7] MASK[33.] DEST[Q] NORM COND[USER] PUSHJ[CKIOTU] $
06m1150                                                                 ;Make sure we're in Exec or IOT-user mode
06 1150                                                 ].repeat waitsfix
06 1151                                                                 ;CLEAR OUT READ, MTAPE AND INTERRUPT FLAGS.
06 1152 13331 010231170000060554161163654314116000             ALU[Q] DEST[D%AMEM7] NORM $
06 1153 13332 610731170000261721431442255714560000             D[CONST 14] ROT[6] DEST[IR-ADR] NORM PUSHJ[KNYRGO] $
06 1154                                                                 ;Fake a word count of 1400
06 1155 13333 010731170000263774161102256714560000             D[MA] DEST[D%AMEM4] NORM PUSHJ[TRP2] $
06 1156 13334 540233170000060554161623254204160000             ALU[Q] DEST[AC] ACSEL[AC] NEOI $
06 1157                                                                 ;Move status into AC.
06 1158
06 1159                                                 ;730, AC/COUNT.  READ WHOLE RECORD, STORING UP TO COUNT WORDS STARTING AT
06 1159                                                  EFF ADR.
06 1160                                                         ;SWAP AC AND IR, THEN DO TAPERX.
06 1161
06 1162                                                 TAPENR:
06 1163                                                 .REPEAT 1 - WAITS [
06 1164                                                         UIOTRP[MUUO] $
06 1165                                                 ];.REPEAT 1 - WAITS
06 1166 13335 610732170000000001416144025771456000             D[IR] DEST[IO-AC IR-ADR] ACSEL[AC]  NORM JUMP[TAPERX] $
06 1167
06 1168
06 1169                                                 ;732 - (AC) IS START ADDR., E IS # OF WORDS TO READ.
06 1170
06 1171                                                 TAPERX:
06 1172                                                 .repeat 1 - waitsfix [
06 1173                                                         UIOTRP[MUUO] $
06 1174                                                         D[D%AMEM7] MASK[33.] DEST[Q] SHORT $
06 1175                                                 ];.repeat 1 - waitsfix
06 1176                                                 .repeat waitsfix [
06 1177                                                         D[D%AMEM7] MASK[33.] DEST[Q] NORM COND[USER] PUSHJ[CKIOTU] $
06 1178                                                                 ;Make sure we're in Exec or IOT-user mode
06m1179                                                 ];[
06m1179 13336 010730124000000001410362226371456000             D[D%AMEM7] MASK[33.] DEST[Q] NORM COND[USER] PUSHJ[CKIOTU] $
06m1179                                                                 ;Make sure we're in Exec or IOT-user mode
06 1179                                                 ].repeat waitsfix
06 1180                                                                 ;CLEAR ALL FLAG BITS IN A-MEM[7]
06 1181 13337 010231170000261734161162254314560000             ALU[Q] DEST[D%AMEM7] NORM PUSHJ[KNYRGO] $ ;GET TAPE STARTED.
06 1182 13340 010731000000000001400762026361456000             D[D%AMEM7] MASK[3] -OBUS=0 CS50 JUMP[INT-READ] $
06 1183                                                                 ;JUMP IF WE ARE TO DO THE INTERRUPT FLAVOR OF
06 1183                                                  INSTRUCTION.
```

```
06 1189 13343 010630170100000000001623655314160000             D[LIT 4400000000000] ALU[DORQ] DEST[Q] NORM $ ;YES, SET BIT 3.
06 1190 13344 540233170003060554161623254204160000     TNRP3:  ALU[Q] DEST[AC] ACSEL[AC] SPEC[MU-POP] NEDI $  ;NO, NOT TOO LONG.
06 1190
06 1191                                                                 ;MOVE STATUS INTO AC.
06 1192
06 1193                                                 ;HERE FOR INTERRUPT FLAVOR OF TAPE READ INSTRUCTIONS, WHERE PI ASSIGNMENT
06 1193
06 1194                                                 ; IS NON-ZERO.
06 1195
06 1196                                                 INT-READ:
06 1197 13345 010730170000060554105623663714160000             D[D%AMEM7] MASK[34.] DEST[Q] NORM $
06 1198                                                                 ;GET PLACE WHERE TO SET READ FLAG, CLEAR INTERRUPT STATE
06 1198                                                  FLAG
06 1199                                                                 ; IN BIT 1.
06 1200 13346 010631170000000010603160255714560000             D[CONST 1] ROT[35.] ALU[DORQ] DEST[D%AMEM7] JUMP[INT-WRITE] NORM
06 1200                                                  $
06 1201                                                                 ;SET READ FLAG IN A-MEM[7].
06 1202                                                                 ;REST OF STUFF SAME AS FOR WRITE.
06 1203
06 1204
06 1205                                                  ;731, AC/ADR, E/+COUNT.  WRITE RECORD OF +COUNT
06 1206                                                         ; WORDS, DATA FROM ADR.
06 1207                                                         ;SET AC:=0 IF OPERATION COMPLETED SUCCESSFULLY.
06 1208                                                         ; SET AC:=<SET2> + HIGHEST ADR READ IF REACHED
06 1209                                                         ; EOT DURING OPERATION (CURRENTLY THIS IS THE ONLY
06 1210                                                         ; ERROR CONDITION.)  OPERATION IS COMPLETED EVEN
06 1211                                                         ; IF EOT IS PASSED.
06 1212
06 1213
06 1214                                                 TAPEWR:
06 1215                                                 .repeat 1 - waitsfix [
06 1216                                                         UIOTRP[MUUO] $
06 1217                                                 ];.repeat 1 - waitsfix
06 1218                                                         D[D%AMEM7] MASK[33.] DEST[Q] NORM
06 1219                                                 .repeat waitsfix [
06 1220                                                                 COND[USER] PUSHJ[CKIOTU]
06m1221                                                 ];[
06m1221                                                                 COND[USER] PUSHJ[CKIOTU]
06 1221                                                 ].repeat waitsfix
06 1222 13347 010730124000000001410362226371456000     $               ;Setup to clear the read, mtape, and interrupt state
06 1222                                                  flags.
06 1223                                                                 ;Make sure we're in Exec or IOT-user mode
06 1224 13350 010231170000262554161162254314560000             ALU[Q] DEST[D%AMEM7] NORM PUSHJ[DCINIT] $
06 1225                                                                 ;Init. the data channel.
06 1226 13351 010731370000060540203623660314160000             D[D%AMEM0]  ROT[1] MASK[1] DEST[AR] NORM $
06 1227                                                                 ;Get the 32-bit mode flag.
06 1228 13352 010730170000060541561623654314160000             D[AR] ROT[35. - 29.] DEST[Q] NORM $
06 1229                                                                 ;Position it for the hardware.
06 1230 13353 010631170031060541603523655714160000             START-OUT D[CONST 1] ROT[35. - 28.] ALU[DORQ] DEST[IOD] NORM $
06 1231                                                                 ;Set the OUT bit.
06 1232 13354 010730170400262023203622255714560000             MAPF[TP.WM] D[CONST 1] ROT[35. - 22.] DEST[Q] PUSHJ[KNYGOA] $
06 1233                                                                 ;Issue WRITE command to formatter.
06 1234 13355 010731000000000014007620263614560000             D[D%AMEM7] MASK[3] -OBUS=0 C550 JUMP[INT-WRITE] $
06 1235                                                                 ;If non-zero PI assignment, do the interrupt form of the
06 1236
06 1236                                                                 ; instruction.
06 1237 13356 010331170000263034161102254314560000             ALU[AC] DEST[D%AMEM4] NORM PUSHJ[STRTDC] $
06 1238                                                                 ;Get word count and start the channel.
06 1239 13357 010731006000000001016162025471556000             D[MEM] ROT[32.] -OBUS<0 JUMP[TWERR] $ ;Test HARD ERR.
06 1240 13360 010243170000060554161623654314160000             ALU[0] DEST[AC] NORM $ ;We return 0 in AC if no EOT.
06 1241 13361 010731004000261610561620254615560000             D[MEM] ROT[34.] C550 OBUS<0 JUMP[TAPEOI] $
06 1242                                                                  ;If no EOT seen, all done.
06 1243 13362 540533170000060547541623255604160000             D[CONST 60] ROT[35. - 5] ALU[DORAC] DEST[AC] NEOI $
06 1244                                                                  ;Turn on bit 0 to indicate EOT seen during operation.
06 1245
06 1246                                                 ;Here for the case where PI assignment is non-zero - just start the
06 1247                                                 ; transfer and return. Also come here at ENDX of INT-READ.
06 1248
06 1249                                                 INT-WRITE:
06 1250 13363 010331170000262754161102254314560000             ALU[AC] DEST[D%AMEM4] PUSHJ[INT-STARTDC] NORM $
06 1251                                                           ;PUT COUNT IN AMEM[4], START THE TRANSFER
06 1252 13364 540731170000060554161623254204160000             NEOI $
06 1253                                                           ;AND RETURN FROM THIS INSTRUCTION WITHOUT CHANGING AC.
06 1254
06 1255
06 1256
06 1257                                                 TWERR:  ;;Error occurred during write.
06 1258
06 1259 13365 010731170000261674161622254314560000             NORM PUSHJ[KNYRS1] $
06 1260                                                           ;Blast the tape formatter and drive
06 1261 13366 540733170000060547521623255604160000             D[CONST 50] ROT[30.] DEST[AC] NEOI $
06 1262                                                           ;And give error return to program. We no longer
06 1263                                                           ;do backspace erase gap rewrite at micro-level, MACRO level
06 1264                                                           ; must do this.
06 1265
06 1266
06 1267
06 1268
06 1269
06 1270 13367 010731170031060554161523661314160000     TAPINT: START-OUT D[D%AMEM2] DEST[IOD] NORM $
06 1271 13370 010731170130060554161623654114160000             MAPF[TP.WC] START-IN C600 $
06 1272                                                           ;CLEAR ENABLES
06 1273 13371 010731370400060554161623657114160000             MAPF[TP.RS] D[IOD] DEST[AR] C800 $
06 1274                                                           ;GET STATUS IN AR FOR CHECKING.
06 1275 13372 010731004000000000361620263614560000             D[D%AMEM7] ROT[1] OBUS<0 C550 JUMP[FMTDON] $
06 1276                                                           ;JUMP IF EXPECTING A -FMT BUSY INTERRUPT
06 1277 13373 010731006000000001416162026361456000             D[D%AMEM7] -OBUS<0 C550 JUMP[TAPINTWRT] $
06 1278                                                           ;JUMP IF THIS XFER IS A WRITE.
06 1279 13374 010731170031060554161623654314160000             START-OUT $
06 1280                                                           ;THE MAPF[0] WRITES LAST WORD TO MEMORY.
06 1281 13375 010731004000000000176162025421456000             MAPF[0] D[AR] ROT[7] OBUS<0 C550 JUMP[INT-RDDONE] $
06 1282                                                           ;IF FORMATTER NO LONGER BUSY, THEN WE READ RECORD SHORTER
06 1283                                                           ; THAN THE COUNT, SO FORGET REST OF TRANSFER.
06 1284
06 1285                                                 ;HERE ALWAYS IF WRITE OR IF READ AND FORMATTER IS STILL GOING.
06 1286                                                 ; MAY EITHER BE ENDX OF TRANSFER OR TIME TO CONTINUE CURRENT LONG RECORD.
06 1286
06 1287                                                 ; TAPE COUNT GO HAS TO BE OFF SINCE ITS EITHER WRITE, OR READ WITH
06 1287                                                  FORMATTER
06 1288                                                 ; IDLE.
```

```
06 1294 13377 010731170000263214161622254314560000                PUSHJ[DCGO] NORM $
06 1295                                                              ;START UP NEXT PART OF THE TRANSFER
06 1296 13400 010730170000000000302562025571456000                MAPF[0] D[CONST 12] ROT[12.] DEST[Q] JUMP[TAPINTENB] NORM $
06 1297                                                              ;MAPF 0 for DCGO. Enable for -FMTR BSY and -TP CNT GO.
06 1298                                                           TAPEXFRDON:
06 1299 13401 010730170000060554161623663774160000                D[D%AMEM7] DEST[Q] SHORT $
06 1300                                                              ;GET DATA TO OR IN WAITING FOR FORMAT DONE BIT
06 1301 13402 010631170000060550403163655714160000                D[CONST 1] ROT[35. - 1] ALU[DORQ] DEST[D%AMEM7] NORM $
06 1302                                                              ;WE CONTINUE AT FMTDON ON NEXT INTERRUPT.
06 1303 13403 010730170000000003005620255714560000                D[CONST 02] ROT[12.] DEST[Q] NORM JUMP[TAPINTENB] $
06 1304                                                              ;BITS FOR JUST -FMT BUSY INTERRUPT ENABLE
06 1305
06 1306                                                           TAPINTENB:;HERE WITH BITS FOR INTERRUPT ENABLES IN Q.
06 1307 13404 010631170031060554161523661314160000                D[D%AMEM2] ALU[DORQ] DEST[IOD] START-OUT NORM $
06 1308                                                              ;ENABLE FOR BITS IN Q.
06 1309                                                           MAPF[TP.WC] C800
06 1310 13405 010731170100227114161620254114560000                 JUMP[MAIN]  $
06 1311                                                              ;AND RETURN FROM THIS MICROINTERRUPT.
06 1312
06 1313 13406 010731006000270141761620254214560000           FMTDON: D[AR] ROT[7] -OBUS<0 C550 JUMP[.] $
06 1314                                                              ;HANG IF FORMATTER IS STILL BUSY.
06 1315 13407 010731006000000014161620263614560000                D[D%AMEM7] -OBUS<0 C550 JUMP[INT-WRTDONE] $
06 1316                                                              ;JUMP IF ITS NOT A READ.
06 1317
06 1318                                                           ;HERE ALSO IF FORMATTER SHUT DOWN BECAUSE NOT ENOUGH DATA ON TAPE TO
06 1318                                                            SATISFY
06 1319                                                           ; THE CHANNEL'S COUNT.
06 1320
06 1321                                                           INT-RDDONE:
06 1322 13410 010731170000263734161622254314560000                PUSHJ[ITRCHK] NORM $
06 1323                                                              ;GET STATUS OF XFER INTO Q
06 1324 13411 010731004000000000561620263614560000                D[D%AMEM7] ROT[2] OBUS<0 JUMP[INT-RDWCOK] C550 $
06 1325                                                              ;IF NON-DATA OPERATION, DON'T CHECK FOR WORD COUNT.
06 1326 13412 010731006000000000416162025421456000                D[AR] ROT[16.] -OBUS<0 JUMP[INT-RDWCOK] C550 $
06 1327                                                              ;JUMP IF NO PROBLEM WITH WORD COUNT
06 1328 13413 010630170000060547511623655714160000                D[CONST 44] ROT[30.] ALU[DORQ] DEST[Q] NORM $
06 1329                                                              ;RECORD WAS LONGER THAN WORD COUNT, SET LENGTH ERROR.
06 1330                                                           INT-RDWCOK:
06 1331 13414 010231170000060554161043654314160000                ALU[Q] DEST[D%AMEM2] NORM $
06 1332                                                              ;STORE STATUS IN AMEM[2] FOR THE TRANSFER STATUS READING INST.
06 1332
06 1333 13415 010730170000060554161623663774160000           TAPEPI: D[D%AMEM7] DEST[Q] SHORT $
06 1334                                                              ;GET CONI WORD.
06 1335 13416 010631170000060554021163655714160000                D[CONST 10] ALU[DORQ] DEST[D%AMEM7] NORM $
06 1336                                                              ;SET INTERRUPT BIT FOR CONSO
06 1337 13417 010730200000203434007620263614560000                D[D%AMEM7] DEST[Q AR] MASK[3] -OBUS=0 JUMP[PIGEN] C550 $
06 1338                                                              ;IF PI CHANNEL ASSIGNED, GO INTERRUPT.
06 1339 13420 540731170000060554161623254204160000                NEOI $
06 1340                                                              ;CLEARED PI ASSIGNMENT OUT DURING XFER?
06 1341
06 1342                                                           INT-WRTDONE:
06 1343 13421 010240170000060554161043654314160000                ALU[0] DEST[D%AMEM2 Q] NORM $
06 1344                                                              ;CLEAR OUT ACCUMULATED STATUS
06 1345 13422 010731004000270510161620254214560000                D[AR] ROT[32.] OBUS<0 JUMP[. + 2] C550 $
06 1346                                                              ;JUMP IF NO HARD ERROR
06 1347 13423 010630170000261670213042255714560000                D[CONST 5] ROT[33.] ALU[DORQ] DEST[D%AMEM2 Q] PUSHJ[KNYRS1] $
06 1348                                                              ;HARD ERROR, SET 1B0 FOR ANY ERROR, 1B2 FOR HARD WRITE ERROR
06 1349 13424 010731004000270330561620254214560000                D[AR] ROT[34.] OBUS<0 C550 JUMP[TAPEPI] $
06 1350                                                              ;IF EOT NOT SET, JUST CAUSE PI ON PROPER CHANNEL.
06 1351 13425 010631170000027033021504025571456000                D[CONST 6] ROT[33.] ALU[DORQ] DEST[D%AMEM2] JUMP[TAPEPI] NORM $
06 1352                                                              ;EOT, SET IT IN AMEM AND GO CAUSE PI
06 1353
06 1354
06 1355
06 1356
06 1357
06 1358                                                           ;TPCONI INSTRUCTION, OPCODE 733.
06 1359                                                           ; 10 MEANS INTERRUPT HAS BEEN REQUESTED.
06 1360                                                           ; 33-35 PI ASSIGNMENT.
06 1361
06 1362                                                           .REPEAT WAITS [
06 1363                                                                  .ORG[TAPE-DISPATCH + 12]
06m1364                                                           ];[
06m1364                                                                  .ORG[TAPE-DISPATCH + 12]
06m1364                                                            [ xlist
06 1364                                                              list ]].REPEAT WAITS
06 1365                                                           TPCONI:
06 1366                                                           .REPEAT 1 - WAITS [
06 1367                                                                  UIOTRP[MUUO] $
06 1368                                                           ];.REPEAT 1 - WAITS
06 1369
06 1370 13300 010730170030000000361562225571456000                START-IN D[CONST 6] ROT[15.] DEST[Q] PUSHJ[TPCONIBTS] NORM $
06 1371                                                              ;GET THE HI ORDER CONI BITS
06 1372 13301 010631310000226654037561263704560000                D[D%AMEM7] MASK[15.] ALU[DORQ] SMAC  [ IFRQ DEST[MEMSTO AR]
06 1372                                                            COND[MA<20] LBJUMP[SEOI] ]$
06 1373                                                              ;STORE CONI BITS INTO EA.
06 1374                                                              ;ALSO STORE INTERRUPT STATE BITS FOR DEBUGGING.
06 1375
06 1376                                                           ;CAUTION:  Still in dispatch table under WAITS switch
06 1377
06 1378                                                           .REPEAT WAITS [
06 1379                                                           TPCONSZ: START-IN D[CONST 6] ROT[15.] DEST[Q] PUSHJ[TPCONIBTS] NORM $
06 1380                                                              ;GET THE HI ORDER CONI BITS
06 1381                                                                  D[D%AMEM7] MASK[15.] ALU[DORQ] DEST[Q] NORM JUMP[XXCONSZ] $
06 1382                                                              ;STORE CONI BITS INTO Q AND DO USUAL CONSZ THING
06 1383                                                           TPCONSO: START-IN D[CONST 6] ROT[15.] DEST[Q] PUSHJ[TPCONIBTS] NORM $
06 1384                                                              ;GET THE HI ORDER CONI BITS
06 1385                                                                  D[D%AMEM7] MASK[15.] ALU[DORQ] DEST[Q] NORM JUMP[XXCONSO] $
06 1386                                                              ;STORE CONI BITS INTO Q AND DO USUAL CONSO THING
06 1387
06 1388                                                                  .RELOC
06m1389                                                           ];[
06m1389 13302 010730170030000000361562225571456000           TPCONSZ: START-IN D[CONST 6] ROT[15.] DEST[Q] PUSHJ[TPCONIBTS] NORM $
06m1389                                                              ;GET THE HI ORDER CONI BITS
06m1389 13303 010630170000025523403762026371456000                D[D%AMEM7] MASK[15.] ALU[DORQ] DEST[Q] NORM JUMP[XXCONSZ] $
06m1389                                                              ;STORE CONI BITS INTO Q AND DO USUAL CONSZ THING
06m1389 13304 010730170030000000361562225571456000           TPCONSO: START-IN D[CONST 6] ROT[15.] DEST[Q] PUSHJ[TPCONIBTS] NORM $
06m1389                                                              ;GET THE HI ORDER CONI BITS
06m1389 13305 010630170000025521403762026371456000                D[D%AMEM7] MASK[15.] ALU[DORQ] DEST[Q] NORM JUMP[XXCONSO] $
06m1389                                                              ;STORE CONI BITS INTO Q AND DO USUAL CONSO THING
```

```
06 1390
06 1391                                         ;TPCONO INSTRUCTION, OPCODE 734.
06 1392                                         ; 100 MEANS CLEAR FORMATTER.
06 1393                                         ; 20 MEANS CLEAR INTERRUPT FLAG.
06 1394                                         ; 33-35 ARE TO SET PI ASSIGNMENT.
06 1395
06 1396                                         .REPEAT WAITS [
06 1397                                                 .ORG[TAPE-DISPATCH + 10]
06m1398                                         ];[
06m1398                                                 .ORG[TAPE-DISPATCH + 10]
06m1398                                          [ xlist
06 1398                                            list ]].REPEAT WAITS
06 1399                                         .REPEAT 1 - WAITS [
06 1400                                         TPCONO: UIOTRP[MUUO] $
06 1401                                         ];.REPEAT 1 - WAITS
06 1402
06 1403 13276 010731004000000007361622257614560000             D[IR] ROT[29.] OBUS<0 C550 PUSHJ[TAPCLR] $
06 1404                                                           ;IF 100 BIT IS SET, CLEAR THE FORMATTER.
06 1405                                         .REPEAT 1 - WAITS [
06 1406                                         ;Only the .reloc really needs to be under WAITS switch.  The JUMP and
                                                 NORM
06 1407                                         ;are done separately to make BINCOM succeed.
06 1408                                                 D[CONST 7] ALU[NOTD] DEST[Q] SHORT $
06 1409                                                   ;MASK TO CLEAR PI ASSIGNMENT
06 1410                                         ];.REPEAT 1 - WAITS
06 1411                                         .REPEAT WAITS  [
06 1412                                                 D[CONST 7] ALU[NOTD] DEST[Q] JUMP[TPCONO2] NORM $
06 1413                                                   ;MASK TO CLEAR PI ASSIGNMENT
06 1414                                            .reloc
06 1415                                         TPCONO2:
06m1416                                         ];[
06m1416 13277 010770170000000001401762025571456000             D[CONST 7] ALU[NOTD] DEST[Q] JUMP[TPCONO2] NORM $
06m1416                                                   ;MASK TO CLEAR PI ASSIGNMENT
06m1416                                            .reloc
06m1416                                          [.USE[HILOC]
06m1416                                          [ xlist
06m1417                                            list ]                                ]TPCONO2:
06 1416                                         ].REPEAT WAITS
06 1417 13426 010731006000270607761620257614560000             D[IR] ROT[31.] -OBUS<0 C550 JUMP[. + 2] $
06 1418                                                   ;JUMP IF NOT CLEARING IT
06 1419 13427 010770170000060554037623655774160000             D[CONST 17] ALU[NOTD] DEST[Q] SHORT $
06 1420                                                   ;CLEARING INTERRUPT FLAG, GET DIFFERENT MASK
06 1421 13430 010641070000060554161623663774160000             D[D%AMEM?] ALU[D&Q] DEST[Q] SHORT $
06 1422                                                   ;CLEAR OUT PI ASSIGNMENT AND MAYBE INTERRUPT FLAG
06 1423 13431 010631170000261614007160257714560000             D[IR] MASK[3] ALU[DORQ] DEST[D%AMEM?] NORM JUMP[TAPEOI] $
06 1424                                                   ;STORE PI ASSIGNMENT IN A-MEM[?] AND RETURN FROM INSTRUCTION.
06 1425
06 1426 13432 010731170000262554161622254314560000     TAPCLR: PUSHJ[DCINIT] NORM $
06 1427                                                   ;GET RID OF INTERRUPT ENABLES, RESET CHANNEL
06 1428 13433 010731170000261674161620254314560000             JUMP[KNYCLR] NORM $
06 1429                                                   ;AND CLEAR FORMATTER AND DRIVE.
06 1430
06 1431                                         ;TPCNSO - OPCODE 735.
06 1432                                         ; SKIP IF BITS IN EA ARE SET.
06 1433                                         ; SEE TPCONI FOR BIT ASSIGNMENTS.
06 1434
06 1435
06 1436                                         .REPEAT 1 - WAITS [
06 1437
06 1438                                         TPCNSO: UIOTRP[MUUO] $
06 1439                                                 START-IN D[CONST 6] ROT[15.] DEST[Q] PUSHJ[TPCONIBTS] NORM $
06 1440                                                   ;GET THE HI ORDER BITS INTO Q.
06 1441                                                 D[D%AMEM?] MASK[18.] ALU[DORQ] DEST[Q] SHORT $
06 1442                                                   ;GET WHAT BITS ARE.
06 1443                                                 D[MA] ALU[D&Q] CONDSKP[-ALU=0] $
06 1444                                                   ;IF ANY MASKED BITS ON => SKIP
06 1445
06 1446                                         ;LEAVE OUT TPCNS2, DON'T WASTE THE SPACE.
06 1447
06 1448                                         ];.REPEAT 1 - WAITS  (includes comment because WAITS includes CONS2)
06 1449
06 1450                                         ;TPDATI - OPCODE 736
06 1451                                         ;RETURNS TRANSFER STATUS (BITS, ENDING ADDRESS) TO C(EA).
06 1452
06 1453                                         .REPEAT WAITS [
06 1454                                                 .ORG[TAPE-DISPATCH + 2]
06m1455                                         ];[
06m1455                                                 .ORG[TAPE-DISPATCH + 2]
06m1455                                          [ xlist
06 1455                                            list ]].REPEAT WAITS
06 1456                                         .REPEAT 1 - WAITS [
06 1457                                         TPDATI: UIOTRP[MUUO] $
06 1458                                         ];.REPEAT 1 - WAITS
06 1459
06 1460 13270 010731310000226654161561261304560000             D[D%AMEM2] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<20] LBJUMP[SEOI]
                                                 ]$
06 1461                                                   ;STORE STATUS SAVED ON LAST XFER TO C(EA)
06 1462                                         .REPEAT WAITS [
06 1463                                                 NOP $
06 1464                                                 JUMP[MUUO] $            ;BLKO
06 1465                                                 NOP $
06 1466                                                 JUMP[MUUO] $            ;DATAO
06 1467                                                 NOP $
06 1468
06 1469                                                 .RELOC
06m1470                                         ];[
06m1470 13271 010731170000060554161623654314160000             NOP $
06m1470 13272 010731170000102114161620254314560000             JUMP[MUUO] $            ;BLKO
06m1470 13273 010731170000060554161623654314160000             NOP $
06m1470 13274 010731170000102114161620254314560000             JUMP[MUUO] $            ;DATAO
06m1470 13275 010731170000060554161623654314160000             NOP $
06m1470
06m1470                                                 .RELOC
06m1470                                          [.USE[HILOC]
06m1470                                          [ xlist
06 1470                                            list ]                                ]]REPEAT WAITS
06 1471
06 1472
06 1473                                         TPCONIBTS:
06 1474 13434 010641070230060554161623657114160000             MAPF[TP.RC] D[IOD] ALU[D&Q] DEST[Q] C800 START-IN $
```

```
06 1080                                                                 ;RETURN IF FORMATTER IS IDLE.
06 1081 13437 0106301700000605436036242557141G000                     D[CONST 1] ROT[35. - 20.] ALU[DORQ] DEST[Q] NORM POPJ $
06 1082                                                                 ;FORMATTER IS BUSY, SET THE BIT.
06 1083                                                                 ;1820 = FORMATTER BUSY
06 1084
06 1085                                                         ];TAPE
06 1086
06 1087
06 1088                                                         ;This probably should be somewhere else...
06 1089                                                         .REPEAT WAITS [         ;If WAITS
06 1090                                                         .REPEAT 1 - LPT [       ;and no LPT
06 1091                                                         .REPEAT 1 - STANSW [    ;and no PAN
06 1092                                                         ;General CONSZ/CONSO termination
06 1093                                                         XXCONSO:
06 1094                                                                 D[IR] MASK[ZZ] ALU[D&Q] CONDSKP[-ALU=0] $
06m1095                                                         XXCONSZ:
06m1095                                                                 D[IR] MASK[ZZ] ALU[D&Q] CONDSKP[ALU=0] $
06m1095                                                         ];.REPEAT 1 - STANSW
06m1095                                                         ];.REPEAT 1 - LPT
06m1095                                                         ];[             ;If WAITS
06m1095                                                         .REPEAT 1 - LPT [       ;and no LPT
06m1095                                                         .REPEAT 1 - STANSW [    ;and no PAN
06m1095                                                         ;General CONSZ/CONSO termination
06m1095                                                         XXCONSO:
06m1095                                                                 D[IR] MASK[ZZ] ALU[D&Q] CONDSKP[-ALU=0] $
06m1095                                                         XXCONSZ:
06m1095                                                                 D[IR] MASK[ZZ] ALU[D&Q] CONDSKP[ALU=0] $
06m1095                                                         ];.REPEAT 1 - STANSW
06m1095                                                         ];.REPEAT 1 - LPT
06 1095                                                         ].REPEAT WAITS
06 1096
```

```
07 1097
07 1098
07 1099                                         .REPEAT TBOOT [
07 1100
07 1101                                         ;BOOTSTRAP LOADER FOR MACROCODE.
07 1102
07 1103                                         MBOOT:  D[CONST 0] ROT[JMEM-P-ROT] DEST[JMEM-P] $
07 1104                                                 ;Reset JMEM pdl ptr
07 1105                                                 D[AR] DEST[ALU1 Q] $
07 1106                                                 ;Save count of files to load.
07 1107                                                 D[CONST 7] DEST[DEV-ADR] PUSHJ[KNYCLR] $
07 1108                                                 ;Reset tape ctrlr
07 1109                                                 PUSHJ[TAPERW] $
07 1110                                                   ;Rewind the tape.
07 1111                                                 ALU[ALU1 Q-1] DEST[Q] -OBUS<0 JUMP[MBOOT1] $
07 1112                                                 ;Get count of files to load.
07 1113                                                 ;Clear memory only if bit 0 is on.
07 1114                                                 D[MASK 10.] DEST[MA] PUSHJ[CLRMEM] NORM $
07 1115                                                 ;Clear some memory.
07 1116
07 1117                                         MBOOT1: MAPF[TP.WM] START-OUT
07 1118                                                  D[CONST 1] ROT[35. - 26.] DEST[IOD] C-OUT $
07 1119                                                   ;ENABLE THE FORMATTER.
07 1120                                                 MAPF[TP.WF] START-IN ALU[0] DEST[Q] CB00 $
07 1121                                                   ;READ STATUS BITS.
07 1122                                                 MAPF[TP.RS] D[IOD] ALU[NOTD] DEST[AR] CB00 $
07 1123                                                 D[AR] ROT[12.]  -OBUS<0 JUMP[MBOOT1] $
07 1124                                                   ;Wait until tape is at load point.
07 1125
07 1126
07 1127                                         MBOOTR: D[CONST 7] DEST[DEV-ADR] CYLEN[LONG] PUSHJ[KNYRGO] $
07 1128                                                 ;Select device 7, start read.
07 1129                                                 PUSHJ[KNYWAIT] $
07 1130                                                 ;Wait for first byte.
07 1131                                                 MAPF[TP.RC] START-IN D[IOD] ALU[NOTD] DEST[HOLD] CB00
07 1131                                          PUSHJ[MBTBYX] $
07 1132                                                 ;Read 1st byte of new record.
07 1133                                                 ALU[0] DEST[MA] NORM JUMP[P2A] $
07 1134                                                 ;Clear MA and enter reading loop.
07 1135
07 1136                                         ;All D[MEM]'s here reading from HOLD.
07 1137
07 1138                                         P2:     D[MA] ALU[D+1] DEST[MA] PUSHJ[MBTBYTE] NORM $
07 1139                                         P2A:    D[MEM] ROT[34] DEST[Q] PUSHJ[MBTBYTE] NORM $
07 1140                                                 D[MEM] ROT[24] ALU[DORQ] DEST[Q] PUSHJ[MBTBYTE] NORM $
07 1141                                                 D[MEM] ROT[14] ALU[DORQ] DEST[Q] PUSHJ[MBTBYTE] NORM $
07 1142                                                 D[MEM] ROT[4] ALU[DORQ] DEST[Q] PUSHJ[MBTBYTE] NORM $
07 1143                                                 D[MEM] MASK[4]  ALU[DORQ] DEST[MEMSTO] NORM JUMP[P2] $
07 1144
07 1145
07 1146                                         MBTBYTE:
07 1147                                                 D[CONST 35] ROT[LLOAD-ROT + 1] DEST[LLOAD] NORM $
07 1148                                                 ;Set loop counter to do timeout
07 1149                                                 ;(TIMEOUT ABOUT 78 USEC)
07 1150                                                 START-IN NORM $
07 1151                                                 ;Ask tape for a byte and status thereof
07 1152                                         MBTBY1: MAPF[TP.RC] D[IOD] DEST[HOLD] CB00 LOOP[MBTBYZ] $
07 1153                                                 ;Read byte and status.  Byte comes complemented.
07 1154                                                 ;Result is put in HOLD to avoid synchronizer problems
07 1155                                                 ;Do timeout check and branch if still waiting
07 1156                                                 START-IN NORM PUSHJ[MBTCHECK] $
07 1157                                                 ;Byte wasn't ready in time.  Go find out why
07 1158                                                 PUSHJ[KNYRGO] $ ;START NEXT RECORD.
07 1159                                                 NORM PUSHJ[KNYWAIT] $
07 1160                                                 MAPF[TP.RC] START-IN D[IOD] ALU[NOTD] DEST[HOLD] CB00
07 1160                                          PUSHJ[MBTBYX] $
07 1161                                                 ;Read 1st byte of new record.
07 1162                                                 NORM SPEC[MU-POP] JUMP[P2A] $
07 1163                                                 ;Go for another record
07 1164                                         ;       ---
07 1165                                         MBTBY2:
07 1166                                                 START-IN D[MEM] ROT[26.] -OBUS<0 C550 JUMP[MBTBY1] $
07 1167                                                 ;Check for byte ready (this is a two instruction loop)
07 1168                                                 ;Start getting byte and status again in case we have
07 1169                                                 ;  to loop
07 1170                                                 START-IN MAPF[TP.RC] D[IOD] ALU[NOTD] DEST[HOLD] CB00 $
07 1171                                                 ;READ THE DATA AGAIN (NOW THAT IT'S STABLE !)
07 1172                                         MBTBYX: MAPF[1] D[MEM] MASK[10] DEST[HOLD] C550 POPJ $
07 1173                                                 ;MAPF[1] clears byte ready
07 1174                                                 ;Extract data byte from other status information
07 1175
07 1176                                         ;       ---
07 1177
07 1178                                         KNYWAIT: ;Wait for first byte of read data (BOOTSTRAP mode only)
07 1179                                                 START-IN SHORT $
07 1180                                                 MAPF[1] START-IN D[CONST 77] ROT[12.] DEST[AR] NORM $
07 1181                                                 ;Clear read data ready flag
07 1182                                         KNYW1:  MAPF[TP.RC] D[IOD] DEST[HOLD] C600 $
07 1183                                                 START-IN D[MEM] ROT[26.] OBUS<0 POPJ C550 $
07 1184                                                 ;Return if READ DATA RDY is now on.
07 1185                                                 START-IN D[AR] ALU[D-1] DEST[AR] C550 -OBUS=0 JUMP[KNYW1] $
07 1186                                                 NORM POPJ $
07 1187
07 1188                                         MBTCHECK:
07 1189                                                 START-IN NORM PUSHJ[FMNBWT] $
07 1190                                                   ;Wait for FORMATTER IDLE.
07 1191                                                 D[MEM] ROT[32.] C550 -OBUS<0 JUMP[MBTERR] $
07 1192                                                   ;Check for error status.
07 1193                                                 D[MEM] ROT[33.] C550 OBUS<0 POPJ $
07 1194                                                   ;If end-of-file not seen, return for more, else done.
07 1195                                                 ALU[0] DEST[Q] NORM $
07 1196                                                 ;Clear lights to indicate no errors.
07 1197                                         MBTDNX: ALU[0] DEST[DEV-ADR] NORM $
07 1198                                                 ;Setup to display code in lights
07 1199                                                 ;Select APR
07 1200                                                 ALU[Q] DEST[D%AMEM17] NORM $
07 1201                                                 ;Load lights from Q.
07 1202
07 1203                                         ;;      D[D%AMEM15]  MASK[2] DEST[AR] COND[OBUS=0] JUMP[.] NORM $
07 1204                                         ;;              ;WAIT FOR A SWITCH.
```

```
07 1210                                                         ;See if we should load more files.
07 1211                                                 D[CONST 12] DEST[AR] JUMP[HLTLOC] $
07 1212                                                         ;No, it's time to quit. Get proper halt code and halt.
07 1213
07 1214                                                 MBTERR: ALU[-1] DEST[Q] JUMP[MBTDNX] $
07 1215
07 1216
07 1217                                                 ;CLEAR MEM STARTING AT MA ON DOWN
07 1218
07 1219                                                 CLRMEM: ALU[0] DEST[MEMSTO] NORM $
07 1220                                                         ;CLEAR THIS ONE
07 1221                                                 D[MA] ALU[D-1] DEST[MA] COND[-OBUS<0] JUMP[. - 1] C600 $
07 1222                                                         ;CONTINUE TIL MA GOES NEGATIVE (JUT DID 0)
07 1223                                                 POPJ NORM $
07 1224
07m122S                                                  ] [
07m122S
07m122S                                                 ;BOOTSTRAP LOADER FOR MACROCODE.
07m122S
07m122S 13440 0107311700000605400000636557141700 0     MBOOT:  D[CONST 0] ROT[JMEM-P-ROT] DEST[JMEM-P] $
07m122S                                                         ;Reset JMEM pdl ptr
07m122S 13441 0107305700000605541616236543141600 0     D[AR] DEST[ALU1 Q] $
07m122S                                                         ;Save count of files to load.
07m122S 13442 0107311700402616740164622557145700 0     D[CONST 7] DEST[DEV-ADR] PUSHJ[KNYCLR] $
07m122S                                                         ;Reset tape ctrlr
07m122S 13443 0107311700002651341616222543145600 0     PUSHJ[TAPERW] $
07m122S                                                   ;Rewind the tape.
07m122S 13444 0102104060000000014161620254314560 0     ALU[ALU1 Q-1] DEST[Q] -OBUS<0 JUMP[MBOOT1] $
07m122S                                                         ;Get count of files to load.
07m122S                                                         ;Clear memory only if bit 0 is on.
07m122S 13445 7107311700000000014045622247314560 0     D[MASK 18.] DEST[MA] PUSHJ[CLRMEM] NORM $
07m122S                                                         ;Clear some memory.
07m122S
07m122S
07m122S                                                 MBOOT1: MAPF[TP.WM] START-OUT
07m122S 13446 0107311704310605422035236555141600 0      D[CONST 1] ROT[35. - 26.] DEST[IOD] C-OUT $
07m122S                                                   ;ENABLE THE FORMATTER.
07m122S 13447 0102401702300605541616236541141600 0     MAPF[TP.WF] START-IN ALU[0] DEST[Q] C800 $
07m122S                                                   ;READ STATUS BITS.
07m122S 13450 0107713704000605541616236571141600 0     MAPF[TP.RS] D[IOD] ALU[NOTD] DEST[AR] C800 $
07m122S 13451 0107310060002711431616202543145600 0     D[AR] ROT[12.]  -OBUS<0 JUMP[MBOOT1] $
07m122S                                                   ;Wait until tape is at load point.
07m122S
07m122S
07m122S
07m122S 13452 0107311700402617340164622555145700 0     MBOOTR: D[CONST 7] DEST[DEV-ADR] CYLEN[LONG] PUSHJ[KNYRGO] $
07m122S                                                         ;Select device 7, start read.
07m122S 13453 0107311700000000014161622254314560 0     PUSHJ[KNYWAIT] $
07m122S                                                         ;Wait for first byte.
07m122S 13454 0107711702300000014161502257114560 0     MAPF[TP.RC] START-IN D[IOD] ALU[NOTD] DEST[HOLD] C800
07m122S                                                  PUSHJ[MBTBYX] $
07m122S                                                         ;Read 1st byte of new record.
07m122S 13455 7102411700000000014161620254314560 0     ALU[0] DEST[MA] NORM JUMP[P2A] $
07m122S                                                         ;Clear MA and enter reading loop.
07m122S
07m122S                                                 ;All D[MEM]'s here reading from HOLD.
07m122S
07m122S 13456 7117011700000000141616222567145600 0     P2:     D[MA] ALU[D+1] DEST[MA] PUSHJ[MBTBYTE] NORM $
07m122S 13457 0107301700000000071616222547155600 0     P2A:    D[MEM] ROT[34] DEST[Q] PUSHJ[MBTBYTE] NORM $
07m122S 13460 0106301700000000051616222547155600 0     D[MEM] ROT[24] ALU[DORQ] DEST[Q] PUSHJ[MBTBYTE] NORM $
07m122S 13461 0106301700000000031616222547155600 0     D[MEM] ROT[14] ALU[DORQ] DEST[Q] PUSHJ[MBTBYTE] NORM $
07m122S 13462 0106301700000000011616222547155600 0     D[MEM] ROT[4] ALU[DORQ] DEST[Q] PUSHJ[MBTBYTE] NORM $
07m122S 13463 0106311700002713540115602547155600 0     D[MEM] MASK[4]  ALU[DORQ] DEST[MEMSTO] NORM JUMP[P2] $
07m122S
07m122S
07m122S                                                 MBTBYTE:
07m122S 13464 0107311700100605402720036557141700 0     D[CONST 35] ROT[LLOAD-ROT + 1] DEST[LLOAD] NORM $
07m122S                                                         ;Set loop counter to do timeout
07m122S                                                         ;(TIMEOUT ABOUT 78 USEC)
07m122S 13465 0107311700300605541616236543141600 0     START-IN NORM $
07m122S                                                         ;Ask tape for a byte and status thereof
07m122S 13466 0107310702000000141615002571145600 0     MBTBY1: MAPF[TP.RC] D[IOD] DEST[HOLD] C800 LOOP[MBTBYZ] $
07m122S                                                         ;Read byte and status.  Byte comes complemented.
07m122S                                                         ;Result is put in HOLD to avoid synchronizer problems
07m122S                                                         ;Do timeout check and branch if still waiting
07m122S 13467 0107311700300000014161622254314560 0     START-IN NORM PUSHJ[MBTCHECK] $
07m122S                                                         ;Byte wasn't ready in time.  Go find out why
07m122S 13470 0107311700002617341616222543145600 0     PUSHJ[KNYRGO] $ ;START NEXT RECORD.
07m122S 13471 0107311700000000141616222543145600 0     NORM PUSHJ[KNYWAIT] $
07m122S 13472 0107711702300000014161502257114560 0     MAPF[TP.RC] START-IN D[IOD] ALU[NOTD] DEST[HOLD] C800
07m122S                                                  PUSHJ[MBTBYX] $
07m122S                                                         ;Read 1st byte of new record.
07m122S 13473 0107311700003271374161620254314560 0     NORM SPEC[MU-POP] JUMP[P2A] $
07m122S                                                         ;Go for another record
07m122S                                                 ;       ---
07m122S                                                 MBTBY2:
07m122S 13474 0107310060302715465616202546155600 0     START-IN D[MEM] ROT[26.] -OBUS<0 C550 JUMP[MBTBY1] $
07m122S                                                         ;Check for byte ready (this is a two instruction loop)
07m122S                                                         ;Start getting byte and status again in case we have
07m122S                                                         ;  to loop
07m122S 13475 0107711702300605541615036571141600 0     START-IN MAPF[TP.RC] D[IOD] ALU[NOTD] DEST[HOLD] C800 $
07m122S                                                         ;READ THE DATA AGAIN (NOW THAT IT'S STABLE !)
07m122S 13476 0107311701000605540215042546151600 0     MBTBYX: MAPF[1] D[MEM] MASK[10] DEST[HOLD] C550 POPJ $
07m122S                                                         ;MAPF[1] clears byte ready
07m122S                                                         ;Extract data byte from other status information
07m122S
07m122S
07m122S                                                 ;       ---
07m122S
07m122S                                                 KNYWAIT: ;Wait for first byte of read data (BOOTSTRAP mode only)
07m122S 13477 0107311700300605541616236543741600 0     START-IN SHORT $
07m122S 13500 0107313701300605431776236557141600 0     MAPF[1] START-IN D[CONST 77] ROT[12.] DEST[AR] NORM $
07m122S                                                         ;Clear read data ready flag
07m122S 13501 0107311702000605541615036571141600 0     KNYW1:  MAPF[TP.RC] D[IOD] DEST[HOLD] C600 $
07m122S 13502 0107310040300605465616242546151600 0     START-IN D[MEM] ROT[26.] OBUS<0 POPJ C550 $
07m122S                                                         ;Return if READ DATA RDY is now on.
07m122S 13503 0107212000302720341616202542145600 0     START-IN D[AR] ALU[D-1] DEST[AR] C550 -OBUS=0 JUMP[KNYW1] $
07m122S 13504 0107311700000605541616242543141600 0     NORM POPJ $
07m122S
07m122S                                                 MBTCHECK:
07m122S 13505 0107311700302621541616222543145600 0     START-IN NORM PUSHJ[FMNEWT] $
07m122S                                                   ;Wait for FORMATTER IDLE.
```

```
07m1225 13511 010241170040060554160463654314170000       MBTDNX: ALU[0] DEST[DEV-ADR] NORM $
07m1225                                                           ;Setup to display code in lights
07m1225                                                           ;Select APR
07m1225 13512 010231170000060554161363654314160000               ALU[Q] DEST[D%AMEM17] NORM $
07m1225                                                           ;Load lights from Q.
07m1225
07m1225                                                   ;;      D[D%AMEM15]  MASK[2] DEST[AR] COND[OBUS=0] JUMP[.] NORM $
07m1225                                                   ;;                ;WAIT FOR A SWITCH.
07m1225                                                   ;;              ;CHECK START AND CONT SWITCHES.  RD NEXT FILE ON CONT
07m1225                                                   ;;        D[AR] MASK[1] COND[-OBUS=0] JUMP[GOMST0] C550 $
07m1225                                                   ;;            ;  ... START MAIN MICROCODE ON START SWITCH.
07m1225
07m1225 13513 010210422000271254161620254314560000               ALU[ALU1 Q-1] DEST[Q] -OBUS18  JUMP[MBOOTR] NORM $
07m1225                                                           ;See if we should load more files.
07m1225 13514 010731370000224234025620255714560000               D[CONST 12] DEST[AR] JUMP[HLTLOC] $
07m1225                                                           ;No, it's time to quit. Get proper halt code and halt.
07m1225
07m1225 13515 010720170000272234001620255714560000       MBTERR: ALU[-1] DEST[Q] JUMP[MBTDNX] $
07m1225
07m1225
07m1225                                                   ;CLEAR MEM STARTING AT MA ON DOWN
07m1225
07m1225 13516 010241170000060554161563654314160000       CLRMEM: ALU[0] DEST[MEMSTO] NORM $
07m1225                                                           ;CLEAR THIS ONE
07m1225 13517 710721006000272354161620256514560000               D[MA] ALU[D-1] DEST[MA] COND[-OBUS<0] JUMP[. - 1] C600 $
07m1225                                                           ;CONTINUE TIL MA GOES NEGATIVE (JUT DID 0)
07m1225 13520 010731170000060554161624254314160000               POPJ NORM $
07m1225
07 1225                                                    ];TBOOT
07 1226
07 1227
```

```
08 1228                                 ;CTY
08 1229
08 1230                                 .REPEAT CTY [
08 1231
08 1232                                 ;-----------------------------------------------------------------------
08 1232                                 ------
08 1233                                 ;
08 1234                                 ;       CTY - Console Teletype                          Device 120
08 1235                                 ;
08 1236                                 ;-----------------------------------------------------------------------
08 1236                                 ------
08 1237
08 1238                                 ;
08 1239                                 ;A-MEM Usage
08 1240                                 ;
08 1241                                 CTY-DISP =      40              ;Instruction and interrupt dispatch
08 1242                                 CTY-CONT =      41              ;Control bits for UART, etc.
08 1243                                 CTY-STATUS =    42              ;Firmware status
08 1244
08 1245                                 ;
08 1246                                 ;*** Meanings of hardware bits should be documented here.
08 1247                                 ;
08 1248                                 ;MAPF values
08 1249
08 1250                                 TTY.DI = 0      ;read data
08 1251                                 TTY.WD = 12     ;write data
08 1252                                 TTY.WC = 14     ;write control
08 1253
08 1254
08 1255                                   .opcode[712]  ;CTY & LPT
08 1256                                         D[CONST 4] DEST[DEV-ADR] NORM COND[-USER] LBJUMP[CTYIOT] $
08 1257                                         D[CONST 4] DEST[DEV-ADR] NORM COND[-USER] LBJUMP[CTYIOT] $
08 1258
08 1259
08 1260                                  .RELOC
08 1261
08 1262
08 1263
08 1264
08 1265                                 CTYDSP: ILGIOT $
08 1265                                  ;BLKI
08 1266                                         ILGIOT $
08 1267
08 1268                                         D[CTY-STATUS] DEST[Q] SPEC[IOB-IN] NORM $
08 1268                                  ;DATAI
08 1269                                         MAPF[TTY.DI] D[IOD] DEST[AR] JUMP[CTYDI] CYLEN[IOB-IN] $
08 1270
08 1271                                         ILGIOT $
08 1271                                  ;BLKO
08 1272                                         ILGIOT $
08 1273
08 1274                                         DFRQ ALU[MEMAC] DEST[HOLD] $
08 1274                                  ;DATAO
08 1275                                                 ;Get C(E) from either memory or ACs.
08 1276                                         D[CTY-STATUS] DEST[Q] JUMP[CTYDO] NORM $
08 1277                                                 ; GET CONI BITS
08 1278
08 1279                                         D[CTY-STATUS] MASK[7] DEST[Q] NORM $
08 1279                                  ;CONO
08 1280                                                 ; GET CONI BITS
08 1281                                         D[IR] MASK[7] ROT[40] ALU[-D&Q] DEST[Q] JUMP[CTYCO] NORM $
08 1282                                                 ;CLR THE CLR BITS
08 1283
08 1284                                         D[CTY-STATUS] DEST[Q] PUSHJ[CTYCI] NORM SPEC[IOB-IN] $
08 1284                                  ;CONI
08 1285                                         D[AR] SMAC $
08 1286
08 1287                                         D[CTY-STATUS] DEST[Q] PUSHJ[CTYCI] NORM SPEC[IOB-IN] $
08 1287                                  ;CONSZ
08 1288                                         D[IR] MASK[18.] DEST[Q] JUMP[CTYCZ] NORM $
08 1289
08 1290                                         D[CTY-STATUS] DEST[Q] PUSHJ[CTYCI] NORM SPEC[IOB-IN] $
08 1290                                  ;CONSO
08 1291                                         D[IR] MASK[18.] DEST[Q] JUMP[CTYCS] NORM $
08 1292
08 1293                                 LPTDSP:
08 1294                                 .REPEAT 1 - LPT [
08 1295                                     .REPEAT 10 [ILGIOT $
08 1296                                                 ILGIOT $
08 1297                                 ];.REPEAT 10
08 1298                                 ];.REPEAT 1 - LPT
08 1299
08 1300                                 .REPEAT LPT [           ;Allocate space for LPT dispatch
08 1301                                 :. + 20
08 1302                                 .repeat 1 - waitsfix [  ;This is a silly place to put it
08 1303                                 .insert F4LPTX                  ;LPT code
08 1304                                 ];.repeat 1 - waitsfix
08 1305                                 ];.REPEAT LPT
08 1306
08 1307
08 1308                                 ;;;CTY and 60 HZ CLOCK INTS COME HERE
08 1309
08 1310                                 CTYINT:   ;;; 60HZ clk shares DEV 4 w/ CTY.
08 1311                                         START-IN SHORT $
08 1312                                         MAPF[5] D[IOD] ALU[NOTD] DEST[AR] C600 $
08 1313                                                 ;Read the NET interface status...
08 1314                                         D[AR] MASK[1] C550 -OBUS=0 JUMP[CLKINT] $
08 1315                                                 ;Is 60HZ clk requesting an int. ? Jump if so.
08 1316                                         NORM JUMP[CTYIN1] $
08 1317                                                 ;Else it is the CTY's turn.
08 1318
08 1319
08 1320                                  ;We branch here from CTYINT if intrpt. is really 60HZ CLK.
08 1321
08 1322                                 CLKINT: START-OUT ALU[0] DEST[IOD] NORM $
08 1323                                           ;Clear hardware int. enb. for 60HZ CLK.
08 1324                                         MAPF[6] C600 $
08 1325
08 1326                                  .repeat TYMNET [
08 1327                                         CLR-DEV-FROM-INTR PUSHJ[TYMCLK] C600 $
```

```
00 1333                 CLKRDFLG:  ;Put hardware 60HZ CLK FLAG into AR35.
00 1334                         START-IN D[CONST 6] DEST[DEV-ADDRESS] NORM $
00 1335                           ;Select device 6.
00 1336                         MAPF[5] D[IOD] ROT[34. + 1] MASK[1] DEST[AR] CYLEN[IOB-IN] POPJ $
00 1336
00 1337                           ;Get bit and return.
00 1338
00 1339                 CLKRST:         ;Fall in to CLKCLR
00 1340                 CLKCLR: START-OUT D[CONST 4] DEST[DEV-ADDRESS] NORM $
00 1341                         ;Clear clk flag.
00 1342                 CLKENB: MAPF[7] D[AMEM-ABS APR-STATUS]  ROT[26.] DEST[IOD] C600 $
00 1343                         ;Get the APR conditions wd, put CLK INT ENB bit in bit
 35.
00 1344                         START-OUT NORM $
00 1345                         MAPF[6] C600 POPJ $
00 1346                         ;SET OR CLEAR THE HARDWARE INTRPT. ENB. ACCORDING TO
00 1347                         ; STATE OF CLK INT ENB BIT IN APR
00 1348
00 1349
00 1350                 CTYIN1: D[CTY-STATUS] DEST[Q] PUSHJ[CTYCI] NORM SPEC[IOB-IN] $
00 1351                         ; DO A CONI, GET BITS
00 1352                 D[CTY-CONT] MASK[11] DEST[IOD] SPEC[IOB-OUT] NORM $; DISABLE INTS
00 1352
00 1353                 MAPF[TTY.WC] CYLEN[IOB-OUT] D[CTY-STATUS] MASK[3] DEST[Q AR] $
00 1354                         ; GET PI CHAN
00 1355                 NORM  SPEC[CLR-DEV-FROM-INTR] JUMP[PIGEN] $;CAUSE INTR.
00 1356
00 1357                   .pair
00 1358                 CTYIOT:
00 1359                 .repeat 1 - waitsfix [
00 1360                 JUMP[MUUO] $    ;Illegal in Luser Mode.
00 1361                 ];.repeat 1 - waitsfix
00 1362                 .repeat waitsfix [
00 1363                 UIOTCK[MUUO] $  ;Some diagnostics check for too much CTY output
00 1364                 ];.repeat waitsfix
00 1365                 IOTDIS
00 1366
00 1367                 CTYDI:  D[CONST 40] ALU[-D&Q]  DEST[CTY-STATUS] NORM $; CLR TTI FLAG
00 1368                 D[CONST 10] ROT[3] DEST[Q] SHORT $
00 1369                 D[CTY-CONT] ALU[DORQ] DEST[IOD] SPEC[IOB-OUT] SHORT $;CLR RCV.
00 1369
 CHR
00 1370                 MAPF[TTY.WC]  CYLEN[IOB-OUT] $
00 1371                 D[CTY-CONT] DEST[IOD] SPEC[IOB-OUT] SHORT $
00 1372                 MAPF[TTY.WC] CYLEN[IOB-OUT] $
00 1373                 D[AR] MASK[10] SMAC $
00 1374                 CTYDO:  D[MEM] DEST[IOD] SPEC[IOB-OUT] NORM $;SEND CHR.
00 1375                 MAPF[TTY.WD] CYLEN[IOB-OUT] D[CONST 10] ALU[-D&Q] DEST[Q] $
00 1376                         ;CLR TTO FLAG
00 1377                 D[CTY-CONT] ALU[D+1] DEST[IOD] SPEC[IOB-OUT] NORM $
00 1378                         ;ENABLE UART STB
00 1379                 MAPF[TTY.WC] CYLEN[IOB-OUT]
00 1380                                 D[CONST 20] ALU[DORQ] DEST[CTY-STATUS]  $
00 1381                         ;SET TTO BUSY
00 1382                 D[CTY-CONT] DEST[IOD] SPEC[IOB-OUT] NORM $;CLR UART STB
00 1383                 MAPF[TTY.WC] CYLEN[IOB-OUT] D[CONST 7] ALU[D&Q] COND[OBUS=0]
00 1383
 JUMP[MAIN] $ ; DONE IF NO PI CHAN
00 1384                 D[CTY-CONT] DEST[Q] SHORT $; GET IOB-OUT BITS
00 1385                 D[CONST 4] ROT[11] ALU[DORQ] DEST[Q IOD] SPEC[IOB-OUT] NORM $
00 1385
 ;ENABLE XMT INT
00 1386                 MAPF[TTY.WC] CYLEN[IOB-OUT] ALU[Q] DEST[CTY-CONT]  JUMP[MAIN] $
00 1387                 CTYCO:  D[CONST 17] ROT[3] ALU[D&Q] DEST[Q] NORM $;CLR PI BITS
00 1388                 D[IR] MASK[7] ALU[DORQ] DEST[CTY-STATUS]  JUMP[CTYCO1] NORM $
00 1389                         ;OR IN NEW PI BITS
00 1390                 CTYCI:  MAPF[TTY.DI] D[IOD] DEST[HOLD] CYLEN[IOB-IN] $;GET UART BITS
00 1391                 D[CTY-CONT] DEST[IOD] SPEC[IOB-OUT] NORM $
00 1392                 MAPF[TTY.WC] D[MEM] ROT[26] MASK[1] COND[-OBUS=0] JUMP[CTYCI1]
00 1392
 CYLEN[MAX,IOB-OUT,C550] $; J IF RCV RDY
00 1393                 CTYCI2: D[CONST 10] ROT[3] ALU[-D&Q] DEST[Q AR] JUMP[CTYCI9] NORM $;CLR
00 1393
 TTI BUSY
00 1394                 CTYCI1: D[CONST 40] ALU[D&Q] COND[-OBUS=0] JUMP[CTYCI2] C550 $;J IF TTI
00 1394
 FLAG ON
00 1395                 D[CONST 10] ROT[3] ALU[D#Q] DEST[Q AR] NORM $;COMPL BUSY.
00 1396                 D[CONST 10] ROT[3] ALU[D&Q] COND[-OBUS=0] JUMP[CTYCI9] C550 $;J
00 1396
 IF BUSY NOW ON
00 1397                 D[CONST 40] ALU[DORQ] DEST[Q AR] NORM $; SET TTI FLAG
00 1398                 CTYCI9: D[MEM] ROT[25] MASK[1] COND[-OBUS=0] JUMP[CTYCI8] C550 $;J IF XMT
00 1398
 RDY
00 1399                 D[CONST 20] ALU[DORQ] DEST[CTY-STATUS]  POPJ NORM $
00 1400                         ;SET BUSY -- NOTE, NO "AR DEST" IS CORRECT
00 1401                 CTYCI8: D[CONST 20] ALU[D&Q] COND[OBUS=0] JUMP[CTYCI3] C550 $; J IF BUSY
00 1401
 OFF
00 1402                 D[CONST 10] ALU[DORQ] DEST[Q] NORM $; SET FLAG
00 1403                 CTYCI3: D[CONST 20] ALU[-D&Q] DEST[CTY-STATUS]  POPJ NORM $ ;CLR BUSY
00 1404                 CTYCO1: D[IR] ROT[41] MASK[10] DEST[Q] NORM $; GET SET&CLR BITS
00 1405                 D[CONST 10] ROT[3] ALU[D&Q] COND[OBUS=0] JUMP[CTYCO4] $
00 1406                         ; J IF CLR TTI FLAG OFF
00 1407                 D[CTY-CONT] MASK[11] DEST[Q] NORM $;GET IOB-OUT BITS
00 1408                 D[CONST 10] ROT[3] ALU[DORQ] DEST[IOD] SPEC[IOB-OUT] NORM $
00 1409                         ;GET CLR RCV RDY BIT
00 1410                 CTYCO3: MAPF[TTY.WC] CYLEN[IOB-OUT] $
00 1411                 ALU[Q] DEST[IOD] SPEC[IOB-OUT] NORM $;CLR IT
00 1412                 MAPF[TTY.WC] CYLEN[IOB-OUT] $
00 1413                 CTYCO4: D[CTY-CONT] MASK[11] DEST[Q] NORM $; GET IOB-OUT BITS
00 1414                 D[CTY-STATUS] MASK[3] COND[OBUS=0] JUMP[CTYCO5] C550 $
00 1415                         ; J IF NO PI CHAN
00 1416                 D[CONST 2] ROT[11] ALU[DORQ] DEST[Q] SHORT $; ENBL RCV INT
00 1417                 D[CTY-STATUS] ROT[41] MASK[2] COND[OBUS=0] JUMP[CTYCO5] C550 $
00 1418                         ; J IF NO OUT FLAG OR BUSY
00 1419                 D[CONST 4] ROT[11] ALU[DORQ] DEST[Q] SHORT $;ENBL XMT INT
00 1420                 CTYCO5: ALU[Q] DEST[IOD] SPEC[IOB-OUT] SHORT $; FIX INT ENBLS
00 1421                 MAPF[TTY.WC] CYLEN[IOB-OUT] ALU[Q] DEST[CTY-CONT]  JUMP[MAIN] $
00 1422                 ; **** HERE IS DEFN. OF CTY UART CONSTANTS ****
00 1423
00 1424                 CTYRST: D[CONST 4] DEST[DEV-ADR] NORM $ D[CONST 4] ROT[6] DEST[Q]
00 1424
 CLR-DEV-FROM-INTR NORM $
00 1425                 D[CONST 74] ALU[DORQ] DEST[Q CTY-CONT]  NORM $
00 1426                 D[CONST 3] ROT[6] ALU[DORQ] DEST[IOD] SPEC[IOB-OUT] NORM $; RESET
00 1426
 UART
00 1427                 MAPF[TTY.WC] CYLEN[IOB-OUT] ALU[0] DEST[CTY-STATUS]  $
00 1428                 ALU[Q] DEST[IOD] SPEC[IOB-OUT] NORM JUMP[CTYRS1] $
```

```
                                                        D[LIT 1000000 = CTYDSP] DEST[Q] $
                                                        D[LIT CTYINT] ALU[DORQ] DEST[D%AMEM0] NORM POPJ $


                                                 .pair
                                                CTYCZ:  D[AR] ALU[D&Q] SKPDSP[ALU=0] $
                                                        SKPDSP[ALWAYS] $
                                                CTYCS:  D[AR] ALU[D&Q] SKPDSP[-ALU=0] $
                                                        SKPDSP[ALWAYS] $

                                                 ] [

                                                ;-----------------------------------------------------------------------
                                                ------
                                                ;
                                                ;       CTY - Console Teletype                          Device 120
                                                ;
                                                ;-----------------------------------------------------------------------
                                                ------

                                                ;
                                                ;A-MEM Usage
                                                ;
        40      CTY-DISP =      40              ;Instruction and interrupt dispatch
        41      CTY-CONT =      41              ;Control bits for UART, etc.
        42      CTY-STATUS =    42              ;Firmware status

                                                ;
                                                ;*** Meanings of hardware bits should be documented here.
                                                ;
                                                ;MAPF values


        0 TTY.DI = 0        ;read data
        12      TTY.WD = 12     ;write data
        14      TTY.WC = 14     ;write control
                                                                                                        . #


                                                  .opcode[71Z]   [xlist
                                                   list  ];CTY & LPT
00m1443 07624 010731126040000014010461255714570000             D[CONST 4] DEST[DEV-ADR] NORM COND[-USER] LBJUMP[CTYIOT] $
00m1443 07625 010731126040000014010461255714570000             D[CONST 4] DEST[DEV-ADR] NORM COND[-USER] LBJUMP[CTYIOT] $


                                                 .RELOC
                                                 [.USE[HILOC]
                                                 [ xlist
                                                   list ]                                ]


00m1444 13521 540731170000060554161623254204160000            CTYDSP: ILGIOT $
                                                  ;BLKI
00m1444 13522 540731170000060554161623254204160000                   ILGIOT $

00m1444 13523 010730170030060554161623661314160000                   D[CTY-STATUS] DEST[Q] SPEC[IOB-IN] NORM $
                                                  ;DATAI
00m1444 13524 010731370000000014161620257114560000                   MAPF[TTY.DI] D[IOD] DEST[AR] JUMP[CTYDI] CYLEN[IOB-IN] $

00m1444 13525 540731170000060554161623254204160000                   ILGIOT $
                                                  ;BLKO
00m1444 13526 540731170000060554161623254204160000                   ILGIOT $

00m1444 13527 010331170000060554161503454312160000                   DFRQ ALU[MEMAC] DEST[HOLD] $
                                                  ;DATAO
                                                                ;Get C(E) from either memory or ACs.
00m1444 13530 010730170000000001416162026131456000                   D[CTY-STATUS] DEST[Q] JUMP[CTYDO] NORM $
                                                                ; GET CONI BITS

00m1444 13531 010730170000060554017623661314160000                   D[CTY-STATUS] MASK[7] DEST[Q] NORM $
                                                  ;CONO
                                                                ; GET CONI BITS
00m1444 13532 010650170000000001001762025771456000                   D[IR] MASK[7] ROT[40] ALU[-D&Q] DEST[Q] JUMP[CTYCO] NORM $
                                                                ;CLR THE CLR BITS

00m1444 13533 010730170030000014161622261314560000                   D[CTY-STATUS] DEST[Q] PUSHJ[CTYCI] NORM SPEC[IOB-IN] $
                                                  ;CONI
00m1444 13534 010731310000226654161561254304560000                   D[AR] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<20] LBJUMP[SEQI] ]$

00m1444 13535 010730170030000014161622261314560000                   D[CTY-STATUS] DEST[Q] PUSHJ[CTYCI] NORM SPEC[IOB-IN] $
                                                  ;CONSZ
00m1444 13536 010730170000000014045620257714560000                   D[IR] MASK[18.] DEST[Q] JUMP[CTYCZ] NORM $

00m1444 13537 010730170030000014161622261314560000                   D[CTY-STATUS] DEST[Q] PUSHJ[CTYCI] NORM SPEC[IOB-IN] $
                                                  ;CONSO
00m1444 13540 010730170000000001404562025771456000                   D[IR] MASK[18.] DEST[Q] JUMP[CTYCS] NORM $

                                                LPTDSP:
                                                .REPEAT 1 - LPT [
                                                    .REPEAT 10 [ILGIOT $
                                                                ILGIOT $
                                                ];.REPEAT 10
                                                ];.REPEAT 1 - LPT

                                                .REPEAT LPT [           ;Allocate space for LPT dispatch
                                                :. + 20
                                                .repeat 1 - waitsfix [  ;This is a silly place to put it
                                                .insert F4LPTX                  ;LPT code
                                                ];.repeat 1 - waitsfix
                                                ];[             ;Allocate space for LPT dispatch
                                                :. + 20
                                                .repeat 1 - waitsfix [  ;This is a silly place to put it
                                                .insert F4LPTX                  ;LPT code
                                                ];.repeat 1 - waitsfix
                                                ].REPEAT LPT


                                                ;;;CTY and 60 HZ CLOCK INTS COME HERE

00m1450 13561 010731170030060554161623654374160000            CTYINT:   ;;; 60HZ clk shares DEV 4 w/ CTY.
                                                                START-IN SHORT $
```

```
08 1456                                                                 ;Else it is the CTY's turn.
08 1457
08 1458
08 1459                                                  ;We branch here from CTYINT if intrpt. is really 60HZ CLK.
08 1460
08 1461 13565 010241170031060554161523654314160000     CLKINT: START-OUT ALU[0] DEST[IOD] NORM $
08 1462                                                           ;Clear hardware int. enb. for 60HZ CLK.
08 1463 13566 010731170600060554161623654114160000             MAPF[6] C600 $
08 1464
08 1465                                                  .repeat TYMNET [
08 1466                                                         CLR-DEV-FROM-INTR PUSHJ[TYMCLK] C600 $
08 1467                                                           ;Start up TYMNET interface if needed.
08 1468                                                    ];TYMNET
08 1469 13567 010731170040201554161620254114560000             CLR-DEV-FROM-INTR JUMP[APRCHK] C600 $
08 1470                                                           ;Cause PI on APR channel if enabled, and exit.
08 1471
08 1472                                                  CLKRDFLG:  ;Put hardware 60HZ CLK FLAG into AR35.
08 1473 13570 010731170030060554014463655714170000             START-IN D[CONST 6] DEST[DEV-ADDRESS] NORM $
08 1474                                                           ;Select device 6.
08 1475 13571 010731370500060550603624257114160000             MAPF[5] D[IOD] ROT[34. + 1] MASK[1] DEST[AR] CYLEN[IOB-IN] POPJ $
08 1475
08 1476                                                           ;Get bit and return.
08 1477
08 1478                                                  CLKRST:          ;Fall in to CLKCLR
08 1479 13572 010731170031060554010463655714170000     CLKCLR: START-OUT D[CONST 4] DEST[DEV-ADDRESS] NORM $
08 1480                                                           ;Clear clk flag.
08 1481 13573 010731170700000046561523670114160000     CLKENB: MAPF[7] D[AMEM-ABS APR-STATUS]  ROT[26.] DEST[IOD] C600 $
08 1482                                                           ;Get the APR conditions wd, put CLK INT ENB bit in bit
08 1482                                                  35.
08 1483 13574 010731170031060554161623654314160000             START-OUT NORM $
08 1484 13575 010731170600060554161624254114160000             MAPF[6] C600 POPJ $
08 1485                                                           ;SET OR CLEAR THE HARDWARE INTRPT. ENB. ACCORDING TO
08 1486                                                           ; STATE OF CLK INT ENB BIT IN APR
08 1487
08 1488
08 1489 13576 010730170030000014161622261314560000     CTYIN1: D[CTY-STATUS] DEST[Q] PUSHJ[CTYCI] NORM SPEC[IOB-IN] $
08 1490                                                           ; DO A CONI, GET BITS
08 1491 13577 010731170031060554023523660714160000             D[CTY-CONT] MASK[11] DEST[IOD] SPEC[IOB-OUT] NORM $; DISABLE INTS
08 1491
08 1492 13600 010730371400060554007623661114160000             MAPF[TTY.WC] CYLEN[IOB-OUT] D[CTY-STATUS] MASK[3] DEST[Q AR] $
08 1493                                                           ; GET PI CHAN
08 1494 13601 010731170040203434161620254314560000             NORM  SPEC[CLR-DEV-FROM-INTR] JUMP[PIGEN] $;CAUSE INTR.
08 1495
08m1496                                                    .pair
08m1496                                                   [:. \ 2 + .
08 1497                                                     ]CTYIOT:
08 1498                                                  .repeat 1 - waitsfix [
08 1499                                                         JUMP[MUUO] $    ;Illegal in Luser Mode.
08 1500                                                  ]:.repeat 1 - waitsfix
08 1501                                                  .repeat waitsfix [
08 1502                                                         UIOTCK[MUUO] $  ;Some diagnostics check for too much CTY output
08m1503                                                  ]:[
08m1503                                                         UIOTCK[MUUO]
08m1503 13602 010731006000102101561620246114560000     [       D[PC-FLAGS] ROT[6] COND[-OBUS<0] JUMP[MUUO] LONG $
08m1503                                                  ]$      ;Some diagnostics check for too much CTY output
08 1503                                                  ].repeat waitsfix
08m1504                                                         IOTDIS
08m1504
08m1504 13603 010730170040060543211623657714160000      [      CLR-DEV-FROM-INTR D[IR] ROT[13.] MASK[4] DEST[Q] NORM $
08m1504 13604 010600170000060543211623657774160000             D[IR] ROT[13.] MASK[4] ALU[D+Q] DEST[Q] SHORT $
08m1504 13605 010601370000060544456236603141600O0              D[D%AMEM0] ROT[18.] MASK[18.] ALU[D+Q] DEST[AR] NORM $
08m1504 13606 010731170000000000160007254114560000             D[AR] ROT[MUA-ROT] ODISP LONG $
08 1505                                                     ]
08 1506 13607 010651170000060554101043655714160000     CTYDI:  D[CONST 40] ALU[-D&Q]  DEST[CTY-STATUS] NORM $; CLR TTI FLAG
08 1507 13610 010730170000060540621623655774160000             D[CONST 10] ROT[3] DEST[Q] SHORT $
08 1508 13611 010631170031060554161523660774160000             D[CTY-CONT] ALU[DORQ] DEST[IOD] SPEC[IOB-OUT] SHORT $;CLR RCV.
08 1508                                                  CHR
08 1509 13612 010731171400060554161623654114160000             MAPF[TTY.WC]  CYLEN[IOB-OUT] $
08 1510 13613 010731170031060554161523660774160000             D[CTY-CONT] DEST[IOD] SPEC[IOB-OUT] SHORT $
08 1511 13614 010731171400060554161623654114160000             MAPF[TTY.WC] CYLEN[IOB-OUT] $
08 1512 13615 010731310000226654021561254304560000             D[AR] MASK[10] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<Z0]
08 1512                                                  LBJUMP[SEQI] ]$
08 1513 13616 010731170031060554161523654714160000     CTYDO:  D[MEM] DEST[IOD] SPEC[IOB-OUT] NORM $;SEND CHR.
08 1514 13617 010650171200060554021523655514160000             MAPF[TTY.WD] CYLEN[IOB-OUT] D[CONST 10] ALU[-D&Q] DEST[Q] $
08 1515                                                           ;CLR TTO FLAG
08 1516 13620 011701170031060554161523660714160000             D[CTY-CONT] ALU[D+1] DEST[IOD] SPEC[IOB-OUT] NORM $
08 1517                                                           ;ENABLE UART STB
08 1518                                                         MAPF[TTY.WC] CYLEN[IOB-OUT]
08 1519 13621 010631171400060554041043655514160000                       D[CONST Z0] ALU[DORQ] DEST[CTY-STATUS]  $
08 1520                                                           ;SET TTO BUSY
08 1521 13622 010731170031060554161523660714160000             D[CTY-CONT] DEST[IOD] SPEC[IOB-OUT] NORM $;CLR UART STB
08 1522 13623 010641003400227114017620255514560000             MAPF[TTY.WC] CYLEN[IOB-OUT] D[CONST 7] ALU[D&Q] COND[OBUS=0]
08 1522                                                  JUMP[MAIN] $ ; DONE IF NO PI CHAN
08 1523 13624 010730170000060554161623660774160000             D[CTY-CONT] DEST[Q] SHORT $; GET IOB-OUT BITS
08 1524 13625 010630170031060542211523655714160000             D[CONST 4] ROT[11] ALU[DORQ] DEST[Q IOD] SPEC[IOB-OUT] NORM $
08 1524                                                  ;ENABLE XMT INT
08 1525 13626 010231171400227114161020254114560000             MAPF[TTY.WC] CYLEN[IOB-OUT] ALU[Q] DEST[CTY-CONT]  JUMP[MAIN] $
08 1526 13627 010640170000060540637623655714160000     CTYCO:  D[CONST 17] ROT[3] ALU[D&Q] DEST[Q] NORM $;CLR PI BITS
08 1527 13630 010631170000000014017040257714560000             D[IR] MASK[7] ALU[DORQ] DEST[CTY-STATUS]  JUMP[CTYCO1] NORM $
08 1528                                                           ;OR IN NEW PI BITS
08 1529 13631 010731170000060554161503657114160000     CTYCI:  MAPF[TTY.DI] D[IOD] DEST[HOLD] CYLEN[IOB-IN] $;GET UART BITS
08 1530 13632 010731170031060554161523660714160000             D[CTY-CONT] DEST[IOD] SPEC[IOB-OUT] NORM $
08 1531 13633 010731001400000005403620254515560000             MAPF[TTY.WC] D[MEM] ROT[26] MASK[1] COND[-OBUS=0] JUMP[CTYCI1]
08 1531                                                  CYLEN[MAX,IOB-OUT,C550] $; J IF RCV RDY
08 1532 13634 010650370000000000621620255714560000     CTYCI2: D[CONST 10] ROT[3] ALU[-D&Q] DEST[Q AR] JUMP[CTYCI9] NORM $;CLR
08 1532                                                  TTI BUSY
08 1533 13635 010641000000274714101620255614560000     CTYCI1: D[CONST 40] ALU[D&Q] COND[-OBUS=0] JUMP[CTYCI2] C550 $;J IF TTI
08 1533                                                  FLAG ON
08 1534 13636 010660370000060540621623655714160000             D[CONST 10] ROT[3] ALU[D=Q] DEST[Q AR] NORM $;COMPL BUSY.
08 1535 13637 010641000000000000621620255614560000             D[CONST 10] ROT[3] ALU[D&Q] COND[-OBUS=0] JUMP[CTYCI9] C550 $;J
08 1535                                                  IF BUSY NOW ON
08 1536 13640 010630370000060554101623655714160000             D[CONST 40] ALU[DORQ] DEST[Q AR] NORM $; SET TTI FLAG
08 1537 13641 010731000000000000520362025461556000     CTYCI9: D[MEM] ROT[25] MASK[1] COND[-OBUS=0] JUMP[CTYCI8] C550 $;J IF XMT
08 1537                                                  RDY
08 1538 13642 010631170000060554041044255714160000             D[CONST 20] ALU[DORQ] DEST[CTY-STATUS]  POPJ NORM $
08 1539                                                           ;SET BUSY -- NOTE, NO "AR DEST" IS CORRECT
08 1540 13643 010641002000000014041620255614560000     CTYCI8: D[CONST 20] ALU[D&Q] COND[OBUS=0] JUMP[CTYCI3] C550 $; J IF BUSY
08 1540                                                  OFF
08 1541 13644 010630170000060554021623655714160000             D[CONST 10] ALU[DORQ] DEST[Q] NORM $; SET FLAG
08 1542 13645 010651170000060554041044255714160000     CTYCI3: D[CONST 20] ALU[-D&Q] DEST[CTY-STATUS]  POPJ NORM $ ;CLR BUSY
```

```
00 1548                                                                 ;GET CLR RCV RDY BIT
00 1549 13652 010731171400060554161523655411416000     CTYCO3: MAPF[TTY.WC] CYLEN[IOB-OUT] $
00 1550 13653 010231170031060554161523655431416000             ALU[Q] DEST[IOD] SPEC[IOB-OUT] NORM $;CLR IT
00 1551 13654 010731171400060554161623655411416000             MAPF[TTY.WC] CYLEN[IOB-OUT] $
00 1552 13655 010730170000060554023623660671416000     CTYCO4: D[CTY-CONT] MASK[11] DEST[Q] NORM $; GET IOB-OUT BITS
00 1553 13656 010731002000000001400762026121456000             D[CTY-STATUS] MASK[3] COND[IOBUS=0] JUMP[CTYCO5] C550 $
00 1554                                                                 ; J IF NO PI CHAN
00 1555 13657 010630170000060542205623655774160000             D[CONST 2] ROT[11] ALU[DORQ] DEST[Q] SHORT $; ENBL RCV INT
00 1556 13660 010731002000000001020562026121456000             D[CTY-STATUS] ROT[41] MASK[2] COND[IOBUS=0] JUMP[CTYCO5] C550 $
00 1557                                                                 ; J IF NO OUT FLAG OR BUSY
00 1558 13661 010630170000060542211623655774160000             D[CONST 4] ROT[11] ALU[DORQ] DEST[Q] SHORT $;ENBL XMT INT
00 1559 13662 010231170031060554161523655437416000     CTYCO5: ALU[Q] DEST[IOD] SPEC[IOB-OUT] SHORT $; FIX INT ENBLS
00 1560 13663 010231171400227114161020254114560000             MAPF[TTY.WC] CYLEN[IOB-OUT] ALU[Q] DEST[CTY-CONT]  JUMP[MAIN] $
00 1561                                                         ; **** HERE IS DEFN. OF CTY UART CONSTANTS ****
00 1562
00 1563 13664 010731170041060554010416365571417000     CTYRST: D[CONST 4] DEST[DEV-ADR] NORM $ D[CONST 4] ROT[6] DEST[Q]
00 1563 13665 010730170041060554141162365571416000      CLR-DEV-FROM-INTR NORM $
00 1564 13666 010630170000060554171023655714160000             D[CONST 74] ALU[DORQ] DEST[Q CTY-CONT]  NORM $
00 1565 13667 010631170031060541407523655714160000             D[CONST 3] ROT[6] ALU[DORQ] DEST[IOD] SPEC[IOB-OUT] NORM $; RESET
00 1565                                                         UART
00 1566 13670 010241171400060554161043654114160000             MAPF[TTY.WC] CYLEN[IOB-OUT] ALU[0] DEST[CTY-STATUS]  $
00 1567 13671 010231170031000014161520254314560000             ALU[Q] DEST[IOD] SPEC[IOB-OUT] NORM JUMP[CTYRS1] $
00 1568
00 1569                                                         CTYRS1:
00 1570                                                         ;Setup entry vectors: IOT vector in left half, interrupt vector in right
00 1570                                                          half
00 1571 13672 010731171400060554161623654114160000             MAPF[TTY.WC] C800 $
00 1572 13673 010730170027242000001623655314160000             D[LIT 1000000 * CTYDSP] DEST[Q] $
00 1573 13674 010631170000000027343004255314160000             D[LIT CTYINT] ALU[DORQ] DEST[D%AMEM0] NORM POPJ $
00 1574
00 1575
00m1576                                                         .pair
00m1576                                                         [:. \ 2 + .
00 1577 13676 540641000000073554161623254204160000               ]CTYCZ:      D[AR] ALU[D&Q] SKPDSP[ALU=0] $
00 1578 13677 540731172000073554161623254204160000             SKPDSP[ALWAYS] $
00 1579 13700 540641002000073554161623254204160000     CTYCS:  D[AR] ALU[D&Q] SKPDSP[-ALU=0] $
00 1580 13701 540731172000073554161623254204160000             SKPDSP[ALWAYS] $
00 1581
00 1443                                                         ];CTY
00 1444
00 1445
00 1446                                                         .repeat waitsfix [      ;A better place
00 1447                                                         .repeat LPT [   ;LPT should follow CTY as they share the same dispatch
00 1447                                                          table
00 1448                                                             .insert LPTX
00 1449                                                                ;LPT code (same source for F2 or F4)
00 1450                                                           ]
00m1451                                                         ];[     ;A better place
00m1451                                                         .repeat LPT [   ;LPT should follow CTY as they share the same dispatch
00m1451                                                          table
00m1451                                                             .insert LPTX
00m1451                                                                ;LPT code (same source for F2 or F4)
00m1451                                                           ]
00m1451                                                         [       ;LPT should follow CTY as they share the same dispatch table
00m1451                                                             .insert LPTX
```

```
01m0001                                         ;-----------------------------------------------------------------------------
01m0001                                         ------
01m0001                                         ;
01m0001                                         ;   LPT - Lineprinter Interface
01m0001                                         ;
01m0001                                         ;-----------------------------------------------------------------------------
01m0001                                         ------
01m0001                                         ;MAPF fields
01m0001       1 LPT-STATUS      = 1     ;Read status register
01m0001                                         ;       Bit 20  ;(firmware only)        ;Slew more paper
01m0001
01m0001                                         ;       Bit 21  ;-LPT READY             ;These three bits only present on
01m0001                                          F2
01m0001                                         ;       Bit 22  ;-LPT LOW PAPER
01m0001                                         ;       Bit 23  ;-LPT PRESENT
01m0001                                         ;
01m0001                                         ;       Bit 24  ;LPT 128                ;Rubout seen on WAITS (or slew
01m0001                                          paper if
01m0001                                         ;                                       ;bit 20 is set).
01m0001                                         ;       Bit 25  ;LPT 96
01m0001                                         ;       Bit 26  ;Not used
01m0001                                         ;
01m0001                                         ;       Bit 27  ;-LPT ON LINE
01m0001                                         ;       Bit 28  ;-SYNC LPT OK
01m0001                                         ;       Bit 29  ;-SYNC LPT DONE
01m0001                                         ;
01m0001                                         ;       Bit 30  ;SYNC ANY INT
01m0001                                         ;       Bit 31  ;SYNC DMA WAITING
01m0001       1 LPT-CONTROL     = 1     ;Set control register
01m0001                                         ;       Bit 16  ;LPT STATUS SPARE
01m0001                                         ;       Bit 17  ;LPT INT ENB
01m0001       2 LPT-DATA        = 2     ;Write data
01m0001                                         ;       Bit 26  ;LPT SPARE OUT
01m0001                                         ;       Bit 27  ;LPT PRINT CMD
01m0001                                         ;       Bit 28  ;LPT DATA BIT 8
01m0001                                         ;       ...
01m0001                                         ;       Bit 35  ;LPT DATA BIT 1
01m0001
01m0001
01m0001                                         ;A-MEM usage
01m0001                                         DEFINE-A-MEM[LPT-DISP 0]        [
01m0001       40                LPT-DISP = 0 + D%AMEM0
01m0001                                         ];Instruction and interrupt dispatch
01m0001                                         DEFINE-A-MEM[LPT-CONSAV 1]      [
01m0001       41                LPT-CONSAV = 1 + D%AMEM0
01m0001                                         ];Save control information including PI channel
01m0001                                         DEFINE-A-MEM[LPT-WORD 3]        [
01m0001       43                LPT-WORD = 3 + D%AMEM0
01m0001                                         ];Last data word read
01m0001                                         DEFINE-A-MEM[LPT-ROT 4]  [
01m0001       44                LPT-ROT = 4 + D%AMEM0
01m0001                                         ]       ;Rotation value for next byte, -n if none left
01m0001                                         DEFINE-A-MEM[LPT-POS 5]  [
01m0001       45                LPT-POS = 5 + D%AMEM0
01m0001                                         ]       ;Column position of the character
01m0001
01m0001                                         ;-----------------------------------------------------------------------------
01m0001                                         ------
01m0001                                         ;       Normal IOT dispatch for LPT
01m0001                                         ;-----------------------------------------------------------------------------
01m0001                                         ------
01m0001
01m0001                                           .org[LPTDSP]
01m0001                                         [ xlist
01m0002                                           list ]
01m0002                                            ;Since macro-device code is 124, dispatch is combined with CTY (see
01m0002                                          F4IO)
01m0002
01m0002                                                 .org[lptdsp]    [ xlist
01m0002                                           list ]        ;*$*$* TEMPORARY *$*$*$
01m0002                                         LPT-DISPATCH:
01m0002                                         ;BLKI LPT, - Not implemented
01m0002 13541 010731170000102114161620254314560000             JUMP[MUUO] $
01m0002 13542 010731170000060554161623654314160000             NOP $
01m0002                                         ;DATAI LPT,
01m0002 13543 010731106000273074161620254374560000             COND[-INTRPT] JUMP[.] SHORT $
01m0002                                                         ;Wait for something to happen.
01m0002 13544 010241310000226654161561254304560000             ALU[0] DEST[MEMSTO] MEMST  [ SMAC  [ IFRQ DEST[MEMSTO AR]
01m0002                                          COND[MA<20] LBJUMP[SEOI] ]]$
01m0002                                                         ;Do same thing as DATAI LPT would, i.e. read nothing.
01m0002                                         ;BLKO LPT, - Not implemented yet
01m0002 13545 010731170000102114161620254314560000             JUMP[MUUO] $
01m0002 13546 010731170000060554161623654314160000             NOP $
01m0002                                         ;DATAO LPT,
01m0002 13547 010331170000060554161503454314216000             FIXM1  [ DFRQ ALU[MEMAC] DEST[HOLD] ]$
01m0002 13550 010731170040000014032460254571457000             D[CONST LPT-UDEV] DEST[DEV-ADR] JUMP[LPTDO2] NORM $
01m0002                                                         ;Set device address to LPT
01m0002                                         ;CONO LPT,
01m0002 13551 010731170040000014032460254571457000             D[CONST LPT-UDEV] DEST[DEV-ADR] JUMP[LPTCO2] $
01m0002                                                         ;Set device address to LPT
01m0002 13552 010731170000060554161623654314160000             NOP $
01m0002                                         ;CONI LPT,
01m0002 13553 010731170030000014032462254571457000             D[CONST LPT-UDEV] DEST[DEV-ADDRESS] SPEC[IOB-IN] PUSHJ[LPTSTS]
01m0002                                          NORM $
01m0002                                                         ;Set device code and get status of LPT
01m0002 13554 010231310000226654161561254304560000             ALU[Q] DEST[MEMSTO] MEMST  [ SMAC  [ IFRQ DEST[MEMSTO AR]
01m0002                                          COND[MA<20] LBJUMP[SEOI] ]]$
01m0002                                                         ;Store status in memory in the usual way.
01m0002                                         ;CONS2 LPT,
01m0002 13555 010731170030000014032462254571457000             D[CONST LPT-UDEV] DEST[DEV-ADDRESS] SPEC[IOB-IN] PUSHJ[LPTSTS]
01m0002                                          NORM $
01m0002                                                         ;Set device code and get status of LPT
01m0002 13556 010731170000255234161620254314560000             JUMP[XXCONS2] NORM $
01m0002                                                         ;Finish reading status from LPT and go do generalized
01m0002                                          CONS2.
01m0002                                         ;CONSO LPT,
01m0002 13557 010731170030000014032462254571457000             D[CONST LPT-UDEV] DEST[DEV-ADDRESS] SPEC[IOB-IN] PUSHJ[LPTSTS]
01m0002                                          NORM $
01m0002                                                         ;Set device code and get status of LPT
01m0002 13560 010731170000255234161620254314560000             JUMP[XXCONSO] NORM $
```

```
01m0002                                                  [ xlist
01m0003                                                    list ]                                  ]
01m0003                                                 ;Continuation of CONO LPT,
01m0003                                                 ;Set control status
01m0003 13702 010731004000000000636162ZZ5761456000      LPTCO2: D[IR] ROT[25.] COND[OBUS<0] PUSHJ[LPTRST] C550 $
01m0003                                                          ;Reset LPT if bit 25 (clear printer) is on
01m0003 13703 010730170000060554045623657774160000              D[IR] MASK[22] DEST[Q] SHORT $
01m0003                                                          ;Use immediate form, i.e. Instruction as data.
01m0003                                                          ;Setup control information
01m0003 13704 010641002000000000144762025561456000              D[CONST 23] ROT[6] ALU[D&Q] COND[OBUS=0] JUMP[LPTCO3] C550 $
01m0003                                                          ;If setting DONE, BUSY or doing CLEAR, then enable micro
01m0003
01m0003                                                          ;interrupts so that BUSY gets cleared and DONE set at the
01m0003
01m0003                                                          ;appropriate times
01m0003 13705 010630170000060544403623655714160000              D[CONST 1] ROT[18.] ALU[DORQ] DEST[Q] NORM $
01m0003                                                          ;Turn on micro interrupts
01m0003 13706 010231170031060554161523654314160000      LPTCO3: ALU[Q] DEST[IOD] SPEC[IOB-OUT] NORM $
01m0003                                                          ;Start sending control information to LPT interface
01m0003                                                          MAPF[LPT-CONTROL] CYLEN[IOB-OUT]
01 0003 13707 010731170100060554161623654114160000      .REPEAT F4SW [ $ ]      [ $ ];Bug:  F4 can't do A-MEM write during
01 0003                                                  IOB-OUT cycle
01 0004 13710 010651170000060542403023655714160000                              D[CONST 1] ROT[35. - 25.] ALU[-D&Q]
01 0004                                                  DEST[LPT-CONSAV] $
01 0005                                                          ;Save control information with reset bit
01 0006 13711 010731004000227107361620250714560000              D[LPT-CONSAV] ROT[29.] COND[OBUS<0] FETCH-NEXT-INST  [ JUMP[MAIN]
01 0006                                                  ]NORM $
01 0007                                                          ;Check done bit
01 0008                                                          ;and start next instruction
01 0009 13712 010731170000000001416162025431456000              JUMP[LPTNT2] NORM $
01 0010                                                          ;Done bit set.  Try interrupt
01 0011
01 0012                                                  ;-----------------------------------------------------------------------
01 0012                                                  ------
01 0013                                                  ;Get LPT status
01 0014                                                  ;
01 0015                                                  ;Call with:     SPEC[IOB-IN] PUSHJ[LPTSTS] $
01 0016                                                  ;-----------------------------------------------------------------------
01 0016                                                  ------
01 0017 13713 010730170100060554037623657114160000      LPTSTS: MAPF[LPT-STATUS] CYLEN[IOB-IN] D[IOD] MASK[15.] DEST[Q] $
01 0018                                                          ;Finish reading status.
01 0019                                                          ;Ignore stuff not being driven
01 0020 13714 010650170000060554021623647314160000              D[MASK 8.] ALU[-D&Q] DEST[Q] NORM $
01 0021                                                          ;Turn off bits where PI channels and done/busy bits go.
01 0022 13715 010650170000060542203623655714160000              D[CONST 1] ROT[35. - 26.] ALU[-D&Q] DEST[Q] NORM $
01 0023                                                          ;Turn off unimplemented bit (it's line overflow at SAIL)
01 0023
01 0024                                                  .REPEAT 1 - WAITS [
01 0025                                                          D[LPT-CONSAV] MASK[8] ALU[DORQ] DEST[Q] POPJ NORM $
01 0026                                                          ;Fill in PI channels and we're done for now.
01 0027                                                  ];.REPEAT 1 - WAITS
01 0028                                                  .REPEAT WAITS [
01 0029                                                          D[LPT-CONSAV] MASK[8] ALU[DORQ] DEST[Q] NORM $
01 0030                                                          ;Fill in PI channels
01 0031                                                          D[LPT-CONSAV] ROT[1 + 24.] MASK[1] DEST[AR] NORM $
01 0032                                                          ;Rubout flag on?
01 0033                                                          D[AR] ROT[35. - 24.] ALU[DORQ] DEST[Q] POPJ $
01 0034                                                          ;Yes, return in CONI word instead of 128-character drum
01 0034                                                  bit.
01m0035                                                  ];[
01m0035 13716 010630170000060554021623660714160000              D[LPT-CONSAV] MASK[8] ALU[DORQ] DEST[Q] NORM $
01m0035                                                          ;Fill in PI channels
01m0035 13717 010731370000060546203623660714160000              D[LPT-CONSAV] ROT[1 + 24.] MASK[1] DEST[AR] NORM $
01m0035                                                          ;Rubout flag on?
01m0035 13720 010630170000060542761624254314160000              D[AR] ROT[35. - 24.] ALU[DORQ] DEST[Q] POPJ $
01m0035                                                          ;Yes, return in CONI word instead of 128-character drum
01m0035                                                  bit.
01 0035                                                  ].REPEAT WAITS
01 0036
01 0037                                                  ;Continuation of DATAO LPT,
01 0038 13721 010730170000060554161623660714160000      LPTDO2: D[LPT-CONSAV] DEST[Q] NORM $
01 0039                                                          ;Get status word
01 0040 13722 010650170000060541403623655714160000              D[CONST 1] ROT[6] ALU[-D&Q] DEST[Q] NORM $
01 0041                                                          ;Turn off DONE
01 0042 13723 010631170000060541405023655714160000              D[CONST 2] ROT[6] ALU[DORQ] DEST[LPT-CONSAV] NORM $
01 0043                                                          ;Turn on BUSY
01 0044 13724 010731000000000014141620254615560000              D[MEM] LEFT COND[-OBUS=0] JUMP[LPTPK7] CS50 $
01 0045                                                          ;Decide which format to use
01 0046 13725 010731170031060554161523654715160000              D[MEM] DEST[IOD] SPEC[IOB-OUT] NORM $
01 0047                                                          ;Send single character to LPT
01 0048                                                          MAPF[LPT-DATA] CYLEN[IOB-OUT]
01 0049 13726 010731170210060540016003655514170000                              d[const 7] rot[mua-rot] lload $
01 0050                                                          ;Finish outputting data
01 0051                                                          ;Begin delaying tactic (4 usec)
01 0052                                                  .repeat 1 - f4sw [
01 0053                                                          loop[.] c500 $
01 0054                                                  ];.repeat 1 - f4sw [
01 0055                                                  .repeat f4sw [
01 0056                                                          loop[.] long $          ;Claimed by F4DEF to be 500 ns
01m0057                                                  ];[
01m0057 13727 010731070000276574161620254114560000              loop[.] long $          ;Claimed by F4DEF to be 500 ns
01 0057                                                  ].repeat f4sw [
01 0058 13730 010631170031060544403523655774160000              D[CONST 1] ROT[18.] ALU[DORQ] DEST[IOD] SPEC[IOB-OUT] short $
01 0059                                                          ;Set control information and enable micro-interrupts.
01 0060 13731 010731170100227114161620254114560000              MAPF[LPT-CONTROL] CYLEN[IOB-OUT] FETCH-NEXT-INST  [ JUMP[MAIN] ]$
01 0060
01 0061                                                          ;Finish output and start fetch of next instruction
01 0062
01 0063                                                  ;-----------------------------------------------------------------------
01 0063                                                  ------
01 0064                                                  ;
01 0065                                                  ; Output word in PDP-10 ASCII format
01 0066                                                  ;
01 0067                                                  ;-----------------------------------------------------------------------
01 0067                                                  ------
01 0068 13732 010731170000060554073103655714160000      LPTPK7: D[CONST (1 + 4 * 7)] DEST[LPT-ROT] NORM $
01 0069                                                          ;Setup initial rotation for fetching first byte from word
01 0069
01 0070 13733 010731170000060554161063654715160000              D[MEM] DEST[LPT-WORD] NORM $
01 0071                                                          ;Save word from memory in A-MEM where we can use it
```

```
01 0076                                                         ;Doing this here is easier than comparing against 35.
01 0077                                                         ;That is, this way we count down 29,22,15,8,1,-6 or
01 0078                                                         ;perhaps 30,24,18,12,6,0,-6
01 0079 13736 0187303700000605520176236S171416000       D[LPT-WORD] ROT[R] MASK[7] DEST[Q AR] NORM $
01 0080                                                         ;Fetch byte from word
01 0081 13737 0107310040000000061616202606145600       D[LPT-CONSAV] ROT[24.] COND[OBUS<0] JUMP[LPTP78] C550 $
01 0082                                                         ;Jump if alternate character set bit set (rubout seen)
01 0083                                                         ;or slewing paper (if bit 20 is set)
01 0084 13740 0107310020000000076056202542145600       D[AR] ROT[1 + 30.] MASK[2] COND[OBUS=0] JUMP[LPTP7F] C550 $
01 0085                                                         ;Jump if control character
01 0086 13741 0106610000000000140176202472145600       D[MASK 7] ALU[D#Q] COND[-OBUS=0] JUMP[LPTP7C] C550 $
01 0087                                                         ;Jump if NOT a rubout
01 0088 13742 0107301700000605541616236607141600       D[LPT-CONSAV] DEST[Q] NORM $
01 0089                                                 ;;      D[CONST 1] ROT[35. - 24.] ALU[DORQ] DEST[LPT-CONSAV]
01 0090                                                 ;;                      JUMP[LPTP7D] NORM $
01 0091                                                         D[CONST 1] ROT[35. - 24.] ALU[DORQ] DEST[LPT-CONSAV]
01 0092 13743 0106311700000000026030202557145600                       JUMP[LPTP7E] NORM $
01 0093                                                         ;OR in alternate character set bit (rubout seen)
01 0094                                                 ;       ---
01 0095                                                 ;Rubout has been seen, meaning alternate character (hidden character) is
01 0096                                                 ;to be printed, or slewing paper if bit 20 set.
01 0097 13744 0107310040000000051616202607145600       LPTP78: D[LPT-CONSAV] ROT[20.] COND[OBUS<0] JUMP[LPTSL2] NORM $
01 0098 13745 0107301700000605541616236607141600               D[LPT-CONSAV] DEST[Q] NORM $
01 0099 13746 0106511700000605426030236557141600       LPTSLE: D[CONST 1] ROT[35. - 24.] ALU[-D&Q] DEST[LPT-CONSAV] NORM $
01 0100                                                         ;Turn off alternate character set bit (rubout seen) and
01 0100                                                  slew
01 0101                                                         ;paper bit
01 0102 13747 0107301700000605414055236557741600               D[CONST 2] ROT[6] DEST[Q] SHORT $
01 0103 13750 0106301700000605520176236617141600               D[LPT-WORD] ROT[R] MASK[7] ALU[DORQ] DEST[Q] NORM $
01 0104                                                         ;Turn on 200 bit
01 0105                                                 ;       \ /
01 0106                                                 ;Character is a normal character (or is ready to print)
01 0107 13751 0102311700310605541615236543741600       LPTP7C: ALU[Q] DEST[IOD] SPEC[IOB-OUT] SHORT $
01 0108                                                         ;Send out character
01 0109 13752 0117001702000605541616236625141600       LPTP7D: MAPF[LPT-DATA] CYLEN[IOB-OUT] D[LPT-POS] ALU[D+1] DEST[Q] $
01 0110                                                         ;Finish sending character.  Advance simulated printhead
01 0111 13753 0102311700000605541611236543141600       LPTP7S: ALU[Q] DEST[LPT-POS] NORM $
01 0112                                                 ;Character has been processed.  Advance to next byte or word.
01 0113 13754 0107301700000605541616236623141600       LPTP7E: D[LPT-ROT] DEST[Q] NORM $
01 0114 13755 0116110040000000140171002556145600               D[CONST 7] ALU[Q-D] DEST[LPT-ROT] COND[OBUS<0] JUMP[LPTIEN] C550
01 0114                                                  $
01 0115                                                         ;Advance byte pointer and check for end of word
01 0116                                                 .REPEAT F4SW [
01 0117                                                         NORM $  ;Wait some more.  Cycle lengthes don't fit mnemonics
01 0118                                                         NORM $
01m0119                                                 ];[
01m0119 13756 0107311700000605541616236543141600               NORM $  ;Wait some more.  Cycle lengthes don't fit mnemonics
01m0119 13757 0107311700000605541616236543141600               NORM $
01 0119                                                 ].REPEAT F4SW
01 0120 13760 0107311700300605541616236543741600               SPEC[IOB-IN] SHORT $
01 0121                                                         ;Start reading status from hardware
01 0122                                                         MAPF[LPT-STATUS] CYLEN[IOB-IN] D[IOD] ROT[1 + 29.] ALU[D#Q]
01 0122                                                  MASK[3]
01 0123 13761 0106610021002767074076202571145600                       COND[OBUS=0] JUMP[LPTP7A] $
01 0124                                                         ;If LPT is still ready to accept data, send some more.
01 0125                                                 ;       \ /
01 0126                                                 ;Enable micro-interrupts so that we can finish processing this word
01 0126                                                  later.
01 0127 13762 0107301702000605541616236605141600       LPTIEN: MAPF[LPT-DATA] CYLEN[IOB-OUT] D[LPT-CONSAV] DEST[Q] $
01 0128                                                         ;Finish sending data in case of unpacked data
01 0129 13763 0106311700310605444035236555141600               D[CONST 1] ROT[18.] ALU[DORQ] DEST[IOD] SPEC[IOB-OUT] long $
01 0130                                                         ;Set micro-interrupt enable and status
01 0131 13764 0107311701402271141616202541145600               MAPF[LPT-CONTROL] CYLEN[IOB-OUT] CLR-DEV-FROM-INTR
01 0131                                                  FETCH-NEXT-INST  [ JUMP[MAIN] ]$
01 0132                                                         ;Finish enabling interrupts and start new instruction
01 0133
01 0134                                                 ;Interpret control characters
01 0135 13765 0116100040000000140136202556145600       LPTP7F: D[CONST 5] ALU[Q-D] DEST[Q] COND[OBUS<0] JUMP[LPTP7I] C550 $
01 0136                                                         ;Jump if less than ↑F
01 0137                                                 ;;;     D[CONST 20] ALU[Q-D] COND[-OBUS<0] JUMP[LPTP7I] C550 $
01 0138                                                 ;;;             ;Jump if greater than ↑W
01 0139 13766 0116110060000000140336202556145600               D[CONST 15] ALU[Q-D] COND[-OBUS<0] JUMP[LPTP7I] C550 $
01 0140                                                         ;Jump if greater than ↑S
01 0141                                                 .repeat 1 - f4sw [
01 0142                                                         D[LPT-DISP] MASK[18.] ALU[D-Q-1] SDISP CYLEN[DISP] $
01 0143                                                         ;Branch according to control code.  Table immediately
01 0144                                                         ;precedes interrupt location
01 0145                                                 ];.repeat 1 - f4sw
01 0146                                                 .repeat f4sw [  ;This distinction might go away someday if MUA-ROT is
01 0146                                                  ever
01 0147                                                         ;ALWAYS zero.  Sigh...
01 0148                                                 D[LPT-DISP] MASK[18.] ALU[D-Q-1] DEST[HOLD] NORM $
01 0149                                                 D[MEM] ROT[MUA-ROT] ODISP LONG $
01 0150                                                         ;Branch according to control code.  Table immediately
01 0151                                                         ;precedes interrupt location
01m0152                                                 ];[  ;This distinction might go away someday if MUA-ROT is ever
01m0152                                                         ;ALWAYS zero.  Sigh...
01m0152 13767 0106211700000605540455036603141600       D[LPT-DISP] MASK[18.] ALU[D-Q-1] DEST[HOLD] NORM $
01m0152 13770 0107311700000000001600072545155600       D[MEM] ROT[MUA-ROT] ODISP LONG $
01m0152                                                         ;Branch according to control code.  Table immediately
01m0152                                                         ;precedes interrupt location
01 0152                                                 ].repeat f4sw
01 0153                                                 ;Control character is ignored (sort of) but also clear column position(?)
01 0153
01 0154                                                 LPTP7I: MAPF[LPT-DATA] CYLEN[IOB-OUT]
01 0155 13771 0107311702000605541616236541141600       .REPEAT F4SW [ $ ]      [ $ ];Bug:  F4 can't do A-MEM write during
01 0155                                                  IOB-OUT cycle
01 0156 13772 0102411700002773141611202543145600                ALU[0] DEST[LPT-POS] JUMP[LPTP7E] $
01 0157
01 0158                                                 ;TAB expansion
01 0159 13773 0107301700000605541616236627141600       LPTHT:  D[LPT-POS] DEST[Q] NORM $
01 0160                                                         ;Get ready to expand tables
01 0161 13774 0112001700310605541616236543141600       LPTHT1: ALU[Q+1] DEST[Q] SPEC[IOB-OUT] NORM $
01 0162                                                         ;Advance to next column and send a space
01 0163 13775 0106410022002772740076202471145600               MAPF[LPT-DATA] C1000 D[MASK 3] ALU[D&Q] COND[OBUS=0] JUMP[LPTP7S]
01 0163                                                  $
01 0164                                                         ;Jump if done with tab, else read status from LPT
01 0165 13776 0102311040002774541611202541145600               ALU[Q] DEST[LPT-POS] COND[INTRPT] JUMP[LPTIEN] C1000 $
01 0166                                                         ;Give interface time for synchronizer to see busy
```

```
01m0172                                                 ];[
01m0172 13777 010731170000060554161623654316 8000              NORM $  ;Wait some more.  Cycle lengthes don't fit mnemonics
01m0172 14000 010731170000060554161623654316 6000              NORM $
01 0172                                                 ].REPEAT F4SW
01 0173 14001 010731170030060554161623654114 6000              SPEC[IOB-IN] C1000 $
01 0174                                                                 ;Take a little more time, then check status
01 0175                                                         MAPF[LPT-STATUS] CYLEN[IOB-IN]
01 0176 14002 010731002100277077611620257114 6000                               D[IOD] ROT[1 + 30.] MASK[4] COND[OBUS=0]
01 0176                                                  JUMP[LPTHT1] $
01 0177                                                                 ;Repeat if LPT is still ready for data
01 0178 14003 010731170000027745416162025431 6000              JUMP[LPTIEN] NORM $
01 0179                                                                 ;Not ready.  Enable micro-interrupts and dismiss
01 0180
01 0181                                                 ;Paper moving operation
01 0182 14004 010601170031277621411520255714 6000      LPTPAP: D[CONST 4] ROT[6] ALU[D+Q] DEST[IOD] SPEC[IOB-OUT] JUMP[LPTP7I] $
01 0182
01 0183                                                                 ;Send out paper command and go clear column control
01 0184
01 0185                                                 ;Slew specific amount of paper
01 0186                                                 LPTSL1: D[CONST 21] ROT[35. - 24.] ALU[DORQ] DEST[LPT-CONSAV]
01 0187 14005 010631170000027730264302025571 6000                              JUMP[LPTP7E] NORM $
01 0188                                                                 ;Turn on alternate bit and slew paper bit
01 0189                                                                 ;Get next byte or interrupt.
01 0190                                                 ;Return here after getting byte.
01 0191 14006 010241170000060554161123654374 6000      LPTSL2: ALU[0] DEST[LPT-POS] SHORT $
01 0192                                                                 ;Make sure column position is cleared, first!
01 0193 14007 011621004000000014035620255614 5600              D[CONST 16] ALU[D-Q] COND[OBUS<0] JUMP[LPTSL3] C550 $
01 0194                                                                 ;Jump if this takes more than one Printronix command
01 0195 14010 010601170031060540705523655714 1600              D[CONST 42] ROT[3] ALU[D+Q] DEST[IOD] SPEC[IOB-OUT] NORM $
01 0196                                                                 ;Make into paper slewing command for Printronix
01 0197 14011 010730170200277154161620260514 5600              MAPF[LPT-DATA] CYLEN[IOB-OUT] D[LPT-CONSAV] DEST[Q] JUMP[LPTSLE]
01 0197                                                  $
01 0198                                                                 ;Finish commands
01 0199                                                 ;More than one Printronix command is required.
01 0200 14012 010731170000060554160443662374 7000      LPTSL3: D[LPT-ROT] DEST[ROTR] SHORT $
01 0201                                                                 ;Setup rotation to decrement lines slewed
01 0202 14013 010730170000060554161623661774 1600              D[LPT-WORD] DEST[Q] SHORT $
01 0203                                                                 ;Get word
01 0204 14014 010731370000060554023623655774 1600              D[CONST 11] DEST[AR] SHORT $
01 0205                                                                 ;Thank you very much, P. Petit
01 0206 14015 011611170000060552161063654314 1600              D[AR] ROT[R] ALU[Q-D] DEST[LPT-WORD] $
01 0207                                                                 ;Decrement number of lines slewed
01 0208 14016 010731170031277440703520255714 5600              D[CONST (410 / 8.)] ROT[3] DEST[IOD] SPEC[IOB-OUT] NORM
01 0208                                                  JUMP[LPTIEN] $
01 0209                                                                 ;Set command to slew 11 (octal) lines of paper and enable
01 0209
01 0210                                                                 ;interrupts.  We will come back to LPTSL2 with the same
01 0210                                                  byte
01 0211                                                                 ;we were working on before, but decremented from
01 0211                                                  original.
01 0212
01 0213                                                 ;27  ?
01 0214                                                 ;;;     D[CONST (10. - 1)] DEST[Q] JUMP[LPTPAP] NORM $
01 0215                                                                 ;Set forms channel and send paper command
01 0216                                                 ;26  ?
01 0217                                                 ;;;     D[CONST (9. - 1)] DEST[Q] JUMP[LPTPAP] NORM $
01 0218                                                                 ;Set forms channel and send paper command
01 0219                                                 ;25  ?
01 0220                                                 ;;;     D[CONST (8. - 1)] DEST[Q] JUMP[LPTPAP] NORM $
01 0221                                                                 ;Set forms channel and send paper command
01 0222                                                 ;24  DC4
01 0223 14017 010730170000300114013620255714 5600              D[CONST (6 - 1)] DEST[Q] JUMP[LPTPAP] NORM $
01 0224                                                                 ;Set forms channel and send paper command
01 0225                                                 ;23  DC3
01 0226 14020 010730170000300114011620255714 5600              D[CONST (5 - 1)] DEST[Q] JUMP[LPTPAP] NORM $
01 0227                                                                 ;Set forms channel and send paper command
01 0228                                                 ;22  DC2
01 0229 14021 010730170000300114007620255714 5600              D[CONST (4 - 1)] DEST[Q] JUMP[LPTPAP] NORM $
01 0230                                                                 ;Set forms channel and send paper command
01 0231                                                 ;21  DC1
01 0232 14022 010730170000300114005620255714 5600              D[CONST (3 - 1)] DEST[Q] JUMP[LPTPAP] NORM $
01 0233                                                                 ;Set forms channel and send paper command
01 0234                                                 ;20  DLE
01 0235 14023 010730170000300114003620255714 5600              D[CONST (2 - 1)] DEST[Q] JUMP[LPTPAP] NORM $
01 0236                                                                 ;Set forms channel and send paper command
01 0237                                                 ;17  SO
01 0238 14024 010731170000277314161520254314 5600              D[AR] DEST[IOD] JUMP[LPTP7E] NORM $
01 0239                                                                 ;Ignore for now
01 0240                                                 ;16  SI
01 0241 14025 010731170000277314161520254314 5600              D[AR] DEST[IOD] JUMP[LPTP7E] NORM $
01 0242                                                                 ;Ignore for now
01 0243                                                 ;15  CR
01 0244 14026 010731170031277634161520254314 5600              D[AR] DEST[IOD] SPEC[IOB-OUT] JUMP[LPTP7I] NORM $
01 0245                                                                 ;Send CR directly to printer.  Clear column counter
01 0246                                                 ;14  FF
01 0247                                                 ;;;     D[CONST (1 - 1)] DEST[Q] JUMP[LPTPAP] NORM $
01 0248                                                 ;;;             ;Set forms channel and send paper command
01 0249 14027 010731170031277634161520254314 5600              D[AR] DEST[IOD] SPEC[IOB-OUT] JUMP[LPTP7I] NORM $
01 0250                                                                 ;Send FF directly to printer.  Clear column counter
01 0251                                                 ;13  VT
01 0252 14030 010730170000300114015620255714 5600              D[CONST (7 - 1)] DEST[Q] JUMP[LPTPAP] NORM $
01 0253                                                                 ;Set forms channel and send paper command
01 0254                                                 ;12  LF
01 0255 14031 010731170031277634161520254314 5600              D[AR] DEST[IOD] SPEC[IOB-OUT] JUMP[LPTP7I] NORM $
01 0256                                                                 ;Send LF directly to printer.  Clear column counter
01 0257                                                 ;11  TAB
01 0258 14032 010731170000277674101520255714 5600              D[CONST 40] DEST[IOD] JUMP[LPTHT] NORM $
01 0259                                                                 ;Put a <space> onto IO bus and start counting them as
01 0259                                                  they
01 0260                                                                 ;are sent
01 0261                                                 ;10  BS
01 0262 14033 010731170000277314161620254314 5600              JUMP[LPTP7E] NORM $
01 0263                                                                 ;Ignore <backspace> for now
01 0264                                                 ;7   (BELL) Slew N lines
01 0265 14034 010730170000300134161620260714 5600              D[LPT-CONSAV] DEST[Q] JUMP[LPTSL1] NORM $
01 0266                                                 ;6
01 0267 14035 010731170000277314161620254314 5600              JUMP[LPTP7E] NORM $
01 0268                                                                 ;Ignore for now
01 0269                                                 ;5   (?) Plot Mode on Printronix
01 0270 14036 010731170031277634161520254314 5600              D[AR] DEST[IOD] SPEC[IOB-OUT] JUMP[LPTP7I] NORM $
```

```
01 0275                                                 ;
01 0276                                                 ;       LPT Interrupt Code
01 0277                                                 ;
01 0278                                                 ;-------------------------------------------------------------------------
01 0278                                                 ------
01 0279                                                 ;This code is immediately preceded by the control character decode table
01 0279
01 0280 14037 010731170031060554045523660714160000     LPTINT: D[LPT-CONSAV] MASK[18.] DEST[IOD] SPEC[IOB-OUT] NORM $
01 0281                                                         ;Turn off micro-interrupts
01 0282                                                 MAPF[LPT-CONTROL] CYLEN[IOB-OUT]
01 0283 14040 010731006100276714161620262114560000                     D[LPT-ROT] COND[-OBUS<0] JUMP[LPTP7A] $
01 0284                                                         ;Finish output
01 0285                                                         ;Check for more characters to process
01 0286 14041 010730170000060554161623660714160000     D[LPT-CONSAV] DEST[Q] NORM $
01 0287                                                         ;Get status word
01 0288 14042 010650170000060541405623655714160000     D[CONST 2] ROT[6] ALU[-D&Q] DEST[Q] NORM $
01 0289                                                         ;Turn off BUSY
01 0290 14043 010631170000060541403023655714160000     D[CONST 1] ROT[6] ALU[DORQ] DEST[LPT-CONSAV] NORM $
01 0291                                                         ;Turn on DONE
01 0292 14044 010731200000203434007620260614560000     LPTNT2: D[LPT-CONSAV] MASK[3] DEST[AR] COND[-OBUS=0] JUMP[PIGEN] C550 $
01 0293                                                         ;Set PI channel and take macro-interrupt if enabled.
01 0294 14045 010731170040227114161620254314560000     CLR-DEV-FROM-INTR FETCH-NEXT-INST  [ JUMP[MAIN] ]NORM $
01 0295                                                         ;Start fetching instruction and dismiss interrupt
01 0296
01 0297
01 0298                                                 ;-------------------------------------------------------------------------
01 0298                                                 ------
01 0299                                                 ;Reset LPT: Set interrupt and dispatch addresses
01 0300                                                 ;-------------------------------------------------------------------------
01 0300                                                 ------
01 0301 14046 010241170000060554161023654314160000     LPTRST: ALU[0] DEST[LPT-CONSAV] NORM $
01 0302 14047 010731170031060554161523660714160000             D[LPT-CONSAV] DEST[IOD] SPEC[IOB-OUT] NORM $
01 0303                                                         ;Clear out interface
01 0304                                                 .REPEAT 1 - F4SW [
01 0305                                                         D[CONST (LPT-DISPATCH / 100)] ROT[30] DEST[Q] SPEC[IOB-OUT]
01 0306                                                                 MAPF[LPT-CONTROL] CYLEN[IOB-OUT] $
01 0307                                                         ;Finish setting control register in hardware
01 0308                                                         ;Construct dispatch address: high 6 bits.
01 0309                                                         ;Start clear data register in hardware
01 0310                                                         D[CONST (LPT-DISPATCH \ 100)] ROT[22] ALU[DORQ] DEST[Q]
01 0311                                                                 MAPF[LPT-DATA] CYLEN[IOB-OUT] $
01 0312                                                         ;Finish clear data register in hardware
01 0313                                                         ;Low 6 bits of dispatch address
01 0314                                                         D[CONST (LPTINT / 100)] ROT[6] ALU[DORQ] DEST[Q] NORM $
01 0315                                                         ;Construct interrupt address: high 6 bits
01 0316                                                         D[CONST (LPTINT \ 100)] ALU[DORQ] DEST[LPT-DISP] NORM$
01 0317                                                         ;Finish contructing interrupt address and store away.
01 0318                                                 ];.REPEAT 1 - F4SW
01 0319                                                 .REPEAT F4SW [
01 0320                                                         D[LIT ((LPT-DISPATCH * 1000000) + LPTINT)] DEST[LPT-DISP] NORM $
01 0320
01m0321                                                 ];[
01m0321 14050 010731170027302030077003655314160000             D[LIT ((LPT-DISPATCH * 1000000) + LPTINT)] DEST[LPT-DISP] NORM $
01m0321
01 0321                                                 ].REPEAT F4SW
01 0322 14051 010241170000060554161123654314160000             ALU[0] DEST[LPT-POS] NORM $
01 0323                                                         ;Clear column position
01 0324 14052 010721170000060554001104255714160000             ALU[-1] DEST[LPT-ROT] NORM POPJ $
01 0325                                                         ;No bytes left in this word
01 0326                                                         ;We're done.
01 0327
01 0328                                                 .REPEAT 1 - STANSW [
01 0329                                                 ;General CONS2/CONS0 termination
01 0330                                                 XXCONS0:
01 0331                                                 .REPEAT 1 - F4SW [
01 0332                                                         D[IR] MASK[Z2] ALU[D&Q] COND[OBUS=0] JUMP[MAIN] C550 $
01 0333                                                         DOSKIP $
01 0334                                                 ];.REPEAT 1 - F4SW
01 0335                                                 .REPEAT F4SW [
01 0336                                                         D[IR] MASK[Z2] ALU[D&Q] CONDSKP[-ALU=0] $
01 0337                                                 ];.REPEAT F4SW
01 0338
01 0339                                                 XXCONSZ:
01 0340                                                 .REPEAT 1 - F4SW [
01 0341                                                         D[IR] MASK[Z2] ALU[D&Q] COND[-OBUS=0] JUMP[MAIN] C550 $
01 0342                                                         DOSKIP $
01 0343                                                 ];.REPEAT 1 - F4SW
01 0344                                                 .REPEAT F4SW [
01 0345                                                         D[IR] MASK[Z2] ALU[D&Q] CONDSKP[ALU=0] $
01 0346                                                 ];.REPEAT F4SW
01 0347                                                 ];.REPEAT 1 - STANSW
```

```
08 1451                                                    ;LPT code (same source for FZ or F4)
08 1451                                                 ]].repeat waitsfix
08 145Z
```

```
09 1453                                                 ;Disk
09 1454
09 1455                                                 .REPEAT DSK [
09 1456
09 1457                                                 ;DISK CTRL IOTS-- 140 - 174
09 1458
09 1459                                                   .opcode[715]  ;DISK CTRL IOTS-- OPCODES 715 TO 724
09 1460
09 1461                                                 .REPEAT 1 - WAITSFIX [
09 1462
09 1463                                                         D[CONST 10] DEST[DEV-ADR] NORM JUMP[DSKIN0] $
09 1464                                                         D[CONST 10] DEST[DEV-ADR] NORM JUMP[DSKIN0] $
09 1465
09 1466                                                         D[CONST 10] DEST[DEV-ADR] NORM JUMP[DSKIN1] $
09 1467                                                         D[CONST 10] DEST[DEV-ADR] NORM JUMP[DSKIN1] $
09 1468
09 1469                                                         D[CONST 10] DEST[DEV-ADR] NORM JUMP[DSKIN2] $
09 1470                                                         D[CONST 10] DEST[DEV-ADR] NORM JUMP[DSKIN2] $
09 1471
09 1472                                                         D[CONST 10] DEST[DEV-ADR] NORM JUMP[DSKIN3] $
09 1473                                                         D[CONST 10] DEST[DEV-ADR] NORM JUMP[DSKIN3] $
09 1474
09 1475                                                         DF/WT D[MEM] DEST[IOD AR] NORM JUMP[DSKO4] $
09 1476                                                         ALU[MEMAC]   DEST[IOD AR] NORM JUMP[DSKO4] $
09 1477
09 1478                                                         DF/WT D[MEM] DEST[IOD] NORM JUMP[DSKO5] $
09 1479                                                         ALU[MEMAC] DEST[IOD] NORM JUMP[DSKO5] $
09 1480
09 1481                                                         DF/WT D[MEM] DEST[IOD] NORM JUMP[DSKO6] $
09 1482                                                         ALU[MEMAC] DEST[IOD] NORM JUMP[DSKO6] $
09 1483
09 1484                                                         DF/WT D[MEM] DEST[IOD] NORM JUMP[DSKO7] $
09 1485                                                         ALU[MEMAC] DEST[IOD] NORM JUMP[DSKO7] $
09 1486
09 1487                                                   .reloc
09 1488                                                   .even
09 1489                                                 ;Read from corresponding register in disk controller.  If not in Exec
09 1489                                                  mode
09 1490                                                 ;and not in IOT User mode, then trap as UUOs.
09 1491                                                 DSKIN0: START-IN $
09 1492                                                          MAPF[0] D[IOD] CS00 DEST[AR] MA<20 LBJUMP[SXEOI] $
09 1493                                                 DSKIN1: START-IN $
09 1494                                                          MAPF[1] D[IOD] CS00 DEST[AR] MA<20 LBJUMP[SXEOI] $
09 1495                                                 DSKIN2: START-IN $
09 1496                                                          MAPF[2] D[IOD] CS00 DEST[AR] MA<20 LBJUMP[SXEOI] $
09 1497                                                 DSKIN3: START-IN $
09 1498                                                          MAPF[3] D[IOD] CS00 DEST[AR] MA<20 LBJUMP[SXEOI] $
09 1499
09 1500                                                 DSKO4:   D[CONST 10] DEST[DEV-ADR] NORM COND[USER] JUMP[ILLUUO] $
09 1501                                                          START-OUT $
09 1502                                                          MAPF[4] D[AR] DEST[D%AMEM2] CS00 JUMP[MAIN] $
09 1503                                                 DSKO5:   D[CONST 10] DEST[DEV-ADR] NORM COND[USER] JUMP[ILLUUO] $
09 1504                                                          START-OUT $
09 1505                                                          MAPF[5] CS00 JUMP[MAIN] $
09 1506                                                 DSKO6:   D[CONST 10] DEST[DEV-ADR] NORM COND[USER] JUMP[ILLUUO] $
09 1507                                                          START-OUT $
09 1508                                                          MAPF[6] CS00 JUMP[MAIN] $
09 1509                                                 DSKO7:   D[CONST 10] DEST[DEV-ADR] NORM COND[USER] JUMP[ILLUUO] $
09 1510                                                          START-OUT $
09 1511                                                          MAPF[7] CS00 JUMP[MAIN] $
09 1512
09 1513                                                 ];.REPEAT 1 - WAITSFIX
09 1514                                                 .REPEAT WAITSFIX [
09 1515                                                         START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
09 1515                                                  LBJUMP[DSKIN0] $
09 1516                                                         START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
09 1516                                                  LBJUMP[DSKIN0] $
09 1517
09 1518                                                         START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
09 1518                                                  LBJUMP[DSKIN1] $
09 1519                                                         START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
09 1519                                                  LBJUMP[DSKIN1] $
09 1520
09 1521                                                         START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
09 1521                                                  LBJUMP[DSKIN2] $
09 1522                                                         START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
09 1522                                                  LBJUMP[DSKIN2] $
09 1523
09 1524                                                         START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
09 1524                                                  LBJUMP[DSKIN3] $
09 1525                                                         START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
09 1525                                                  LBJUMP[DSKIN3] $
09 1526
09 1527                                                         DF/WT D[MEM] DEST[IOD AR] NORM COND[-USER] LBJUMP[DSKO4] $
09 1528                                                         ALU[MEMAC]   DEST[IOD AR] NORM COND[-USER] LBJUMP[DSKO4] $
09 1529
09 1530                                                         DF/WT D[MEM] DEST[IOD] NORM COND[-USER] LBJUMP[DSKO5] $
09 1531                                                         ALU[MEMAC] DEST[IOD] NORM COND[-USER] LBJUMP[DSKO5] $
09 1532
09 1533                                                         DF/WT D[MEM] DEST[IOD] NORM COND[-USER] LBJUMP[DSKO6] $
09 1534                                                         ALU[MEMAC] DEST[IOD] NORM COND[-USER] LBJUMP[DSKO6] $
09 1535
09 1536                                                         DF/WT D[MEM] DEST[IOD] NORM COND[-USER] LBJUMP[DSKO7] $
09 1537                                                         ALU[MEMAC] DEST[IOD] NORM COND[-USER] LBJUMP[DSKO7] $
09 1538
09 1539                                                   .reloc
09 1540                                                   .even
09 1541                                                 ;Read from corresponding register in disk controller.  If not in Exec
09 1541                                                  mode
09 1542                                                 ;and not in IOT User mode, then trap as UUOs.
09 1543                                                 DSKIN0: MAPF[0] START-IN UIOTCK[ILLUUO] $
09 1544                                                          MAPF[0] D[IOD] CS00 DEST[AR] MA<20 LBJUMP[SXEOI] $
09 1545                                                 DSKIN1: MAPF[1] START-IN UIOTCK[ILLUUO] $
09 1546                                                          MAPF[1] D[IOD] CS00 DEST[AR] MA<20 LBJUMP[SXEOI] $
09 1547                                                 DSKIN2: MAPF[2] START-IN UIOTCK[ILLUUO] $
09 1548                                                          MAPF[2] D[IOD] CS00 DEST[AR] MA<20 LBJUMP[SXEOI] $
09 1549                                                 DSKIN3: MAPF[3] START-IN UIOTCK[ILLUUO] $
09 1550                                                          MAPF[3] D[IOD] CS00 DEST[AR] MA<20 LBJUMP[SXEOI] $
09 1551
09 1552                                                         .PAIR
                                                        DSKO4:  UIOTCK[ILLUUO] $
```

```
09 1558                                          DSK0S:  UIOTCK[ILLUUO] $
09 1559                                                  START-OUT D[CONST 10] DEST[DEV-ADDRESS] NORM $
09 1560                                                   MAPF[5] C500 JUMP[MAIN] $
09 1561
09 1562                                                  .PAIR
09 1563                                          DSK06:  UIOTCK[ILLUUO] $
09 1564                                                  START-OUT D[CONST 10] DEST[DEV-ADDRESS] NORM $
09 1565                                                   MAPF[6] C500 JUMP[MAIN] $
09 1566
09 1567                                                  .PAIR
09 1568                                          DSK07:  UIOTCK[ILLUUO] $
09 1569                                                  START-OUT D[CONST 10] DEST[DEV-ADDRESS] NORM $
09 1570                                                   MAPF[7] C500 JUMP[MAIN] $
09 1571                                          ];.REPEAT WAITSFIX
09 1572
09 1573                                          DSKRST:  SPEC[IOB-OUT] ALU[0] DEST[IOD] NORM $
09 1574                                                    ;SET DSK CTRL COMMAND REGISTER TO 0 (DISABLES INTS).
09 1575                                                  MAPF[4] SPEC[IOB-OUT] D[CONST 2] DEST[IOD]  CYLEN[IOB-OUT] $
09 1576                                                    ;NOW RESET THE CONTROLLER.
09 1577                                                  MAPF[7] ALU[0] DEST[D%AMEM1]  CYLEN[IOB-OUT]  $
09 1578                                                    ;ALSO CLEAR THE PI CHANNEL ASSIGNMENT.
09 1579                                                  D[LABEL DSKINT] DEST[D%AMEM0] POPJ $
09 1580
09 1581                                          ILLUUO: JUMP[MUUO] $
09 1582
```

```
10 1583                                         ; DISK Stuff
10 1584
10 1585                                         ;a-mem useage:
10 1586                                         ;       0       dispatch addr for interrupts
10 1587                                         ;       1       pi chn (33: 35) and intrpt waiting flag (32)
10 1588                                         ;       2       copy of last cmd sent to controller (by opcode 721)
10 1589
10 1590                                           .opcode[740]  ;Disk pseudo-iot dispatch table entries...
10 1591                                                 JUMP[DCNOA] $
10 1592                                                 JUMP[DCNOA] $
10 1593
10 1594                                                 JUMP[DCNIA] $
10 1595                                                 JUMP[DCNIA] $
10 1596
10 1597                                                 JUMP[DCNSOA] $
10 1598                                                 JUMP[DCNSOA] $
10 1599
10 1600                                                 JUMP[DCNSZA] $
10 1601                                                 JUMP[DCNSZA] $
10 1602
10 1603                                           .reloc
10 1604                                         ; dcono -- 740
10 1605                                         DCNOA:
10 1606                                         .REPEAT 1 - WAITSFIX [
10 1607                                                 d[const 10] dest[dev-adr] short $
10 1608                                         ];.REPEAT 1 - WAITSFIX
10 1609                                         .REPEAT  WAITSFIX [
10 1610                                                 d[const 10] dest[dev-adr] norm cond[user] pushj[ckiotu] $
10 1611                                         ];.REPEAT  WAITSFIX
10 1612                                                 d[ir] mask[3] dest[D%AMEM1] norm jump[dcno1]$
10 1613                                         ; dconi -- 741
10 1614                                         DCNIA:  d[const 10] dest[dev-adr] norm pushj[dcni1] $ ;get bits.
10 1615                                                 d[ar] SMAC $ ;store them and return.
10 1616                                         ; dconso -- 742
10 1617                                         DCNSOA: d[const 10] dest[dev-adr] norm pushj[dcni1] $ ;get bits.
10 1618                                                 d[mask 22] alu[d&q] c550 cond[-obus=0] lbjump[dskp1] $
10 1619                                         ; dconsz -- 743
10 1620                                         DCNSZA: d[const 10] dest[dev-adr] norm pushj[dcni1] $ ;get bits.
10 1621                                                 d[mask 22] alu[d&q] c550 cond[obus=0] lbjump[dskp1] $
10 1622
10 1623                                         dcno1:  d[ir] mask[3]  c500 cond[obus=0] jump[main] $
10 1624                                                  ; if assigned pi channel is not 0, then
10 1625                                                  ;  enable interrupt on "not active", by
10 1626                                                  ;  re-loading last cmd with 10 bit on.
10 1627                                                 D[CONST 20] DEST[Q] NORM $
10 1628                                                 D[IR] ALU[D&Q] C550 OBUS=0 JUMP[DCNO2] $
10 1629                                                 D[CONST 40] DEST[Q] NORM JUMP[DCNO3] $
10 1630                                         DCNO2:  d[const 10] dest[q] short $
10 1631                                         DCNO3:  d[D%AMEM2] alu[dorq] dest[iod] spec[iob-out] norm $
10 1632                                                 mapf[4] cylen[iob-out] jump[main] $
10 1633
10 1634                                         dcni1:  d[D%AMEM1] dest[q ar] norm $ ;get intrpt flag and pi chn
10 1635                                         .REPEAT 1 - WAITSFIX [
10 1636                                                 d[ir] alu[d&q] dest[q] norm popj $ ;this is for conso, z
10 1637                                         ];.REPEAT 1 - WAITSFIX
10 1638                                         .REPEAT WAITSFIX [
10 1639                                                 d[ir] alu[d&q] dest[q] norm cond[-user] popj $ ;this is for
10 1639                                          conso, z
10 1640                                         ;       \ /
10 1641                                         ;Subroutine to trap if not in EXEC or IOT-User mode
10 1642                                         CKIOTU: UIOTCK[MUUO] $
10 1643                                                 POPJ $
10 1644
10 1645                                         ];.REPEAT WAITSFIX
10 1646
10 1647                                                 ;interrupts from disk (dev 10) come here.
10 1648                                         DSKINT: D[CONST 50] ALU[NOTD] DEST[Q] NORM  $
10 1649                                                 d[D%AMEM2] ALU[D&Q] dest[iod] spec[iob-out] norm  $
10 1650                                                   ;clear interrrupt enable bit (amem[2] has last cmd)
10 1651                                                 mapf[4] d[D%AMEM1] dest[q ar] c550 cond[obus=0] jump[ddis] $
10 1652                                                 d[const 10] alu[dorq] dest[D%AMEM1] norm jump[pigen] $
10 1653                                                   ;set flag and request intrpt.
10 1654                                         ddis:   clr-dev-from-intr norm jump[main] $
10 1655
10 1656                                         DSKWT1: D[CONST 10] DEST[DEV-ADR AR] NORM JUMP[DSKWT4] $
10 1657                                         DSKWT3: D[MEM] ROT[31.] C550 COND[OBUS<0] JUMP[DSKWDN] $
10 1658                                                 D[CONST 1] ROT[LLOAD-ROT + 6] LLOAD NORM $
10 1659                                                 C550 LOOP[.] $
10 1660                                         DSKWT4: START-IN NORM $
10 1661                                                 MAPF[0] D[IOD] DEST[HOLD] C800  JUMP[DSKWT3] $
10 1662                                         DSKWDN: D[AR] ALU[D-1] DEST[AR] C550 COND[-OBUS=0] JUMP[DSKWT4] $
10 1663                                                 JUMP[MAIN] $
10 1664
```

```
11 1665
11m1666                                                 ] [
11m1666
11m1666                                                 ;DISK CTRL IOTS-- 140 - 174
11m1666
11m1666                                                   .opcode[715]   [xlist
11m1666                                                    list  ];DISK CTRL IOTS-- OPCODES 715 TO 724
11m1666
11m1666                                                 .REPEAT 1 - WAITSFIX [
11m1666
11m1666                                                         D[CONST 10] DEST[DEV-ADR] NORM JUMP[DSKIN0] $
11m1666                                                         D[CONST 10] DEST[DEV-ADR] NORM JUMP[DSKIN0] $
11m1666
11m1666                                                         D[CONST 10] DEST[DEV-ADR] NORM JUMP[DSKIN1] $
11m1666                                                         D[CONST 10] DEST[DEV-ADR] NORM JUMP[DSKIN1] $
11m1666
11m1666                                                         D[CONST 10] DEST[DEV-ADR] NORM JUMP[DSKIN2] $
11m1666                                                         D[CONST 10] DEST[DEV-ADR] NORM JUMP[DSKIN2] $
11m1666
11m1666                                                         D[CONST 10] DEST[DEV-ADR] NORM JUMP[DSKIN3] $
11m1666                                                         D[CONST 10] DEST[DEV-ADR] NORM JUMP[DSKIN3] $
11m1666
11m1666                                                         DF/WT D[MEM] DEST[IOD AR] NORM JUMP[DSK04] $
11m1666                                                         ALU[MEMAC]   DEST[IOD AR] NORM JUMP[DSK04] $
11m1666
11m1666                                                         DF/WT D[MEM] DEST[IOD] NORM JUMP[DSK05] $
11m1666                                                         ALU[MEMAC] DEST[IOD] NORM JUMP[DSK05] $
11m1666
11m1666                                                         DF/WT D[MEM] DEST[IOD] NORM JUMP[DSK06] $
11m1666                                                         ALU[MEMAC] DEST[IOD] NORM JUMP[DSK06] $
11m1666
11m1666                                                         DF/WT D[MEM] DEST[IOD] NORM JUMP[DSK07] $
11m1666                                                         ALU[MEMAC] DEST[IOD] NORM JUMP[DSK07] $
11m1666
11m1666                                                   .reloc
11m1666                                                   .even
11m1666                                                 ;Read from corresponding register in disk controller.  If not in Exec
11m1666                                                  mode
11m1666                                                 ;and not in IOT User mode, then trap as UUOs.
11m1666                                                 DSKIN0: START-IN $
11m1666                                                          MAPF[0] D[IOD] C500 DEST[AR] MA<20 LBJUMP[SXEOI] $
11m1666                                                 DSKIN1: START-IN $
11m1666                                                          MAPF[1] D[IOD] C500 DEST[AR] MA<20 LBJUMP[SXEOI] $
11m1666                                                 DSKIN2: START-IN $
11m1666                                                          MAPF[2] D[IOD] C500 DEST[AR] MA<20 LBJUMP[SXEOI] $
11m1666                                                 DSKIN3: START-IN $
11m1666                                                          MAPF[3] D[IOD] C500 DEST[AR] MA<20 LBJUMP[SXEOI] $
11m1666
11m1666                                                 DSK04:    D[CONST 10] DEST[DEV-ADR] NORM COND[USER] JUMP[ILLUUO] $
11m1666                                                           START-OUT $
11m1666                                                           MAPF[4] D[AR] DEST[D%AMEMZ] C500 JUMP[MAIN] $
11m1666                                                 DSK05:    D[CONST 10] DEST[DEV-ADR] NORM COND[USER] JUMP[ILLUUO] $
11m1666                                                           START-OUT $
11m1666                                                           MAPF[5] C500 JUMP[MAIN] $
11m1666                                                 DSK06:    D[CONST 10] DEST[DEV-ADR] NORM COND[USER] JUMP[ILLUUO] $
11m1666                                                           START-OUT $
11m1666                                                           MAPF[6] C500 JUMP[MAIN] $
11m1666                                                 DSK07:    D[CONST 10] DEST[DEV-ADR] NORM COND[USER] JUMP[ILLUUO] $
11m1666                                                           START-OUT $
11m1666                                                           MAPF[7] C500 JUMP[MAIN] $
11m1666
11m1666                                                 ];.REPEAT 1 - WAITSFIX
11m1666                                                 .REPEAT WAITSFIX [
11m1666                                                         START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
11m1666                                                  LBJUMP[DSKIN0] $
11m1666                                                         START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
11m1666                                                  LBJUMP[DSKIN0] $
11m1666
11m1666                                                         START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
11m1666                                                  LBJUMP[DSKIN1] $
11m1666                                                         START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
11m1666                                                  LBJUMP[DSKIN1] $
11m1666
11m1666                                                         START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
11m1666                                                  LBJUMP[DSKIN2] $
11m1666                                                         START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
11m1666                                                  LBJUMP[DSKIN2] $
11m1666
11m1666                                                         START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
11m1666                                                  LBJUMP[DSKIN3] $
11m1666                                                         START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
11m1666                                                  LBJUMP[DSKIN3] $
11m1666
11m1666                                                         DF/WT D[MEM] DEST[IOD AR] NORM COND[-USER] LBJUMP[DSK04] $
11m1666                                                         ALU[MEMAC]   DEST[IOD AR] NORM COND[-USER] LBJUMP[DSK04] $
11m1666
11m1666                                                         DF/WT D[MEM] DEST[IOD] NORM COND[-USER] LBJUMP[DSK05] $
11m1666                                                         ALU[MEMAC] DEST[IOD] NORM COND[-USER] LBJUMP[DSK05] $
11m1666
11m1666                                                         DF/WT D[MEM] DEST[IOD] NORM COND[-USER] LBJUMP[DSK06] $
11m1666                                                         ALU[MEMAC] DEST[IOD] NORM COND[-USER] LBJUMP[DSK06] $
11m1666
11m1666                                                         DF/WT D[MEM] DEST[IOD] NORM COND[-USER] LBJUMP[DSK07] $
11m1666                                                         ALU[MEMAC] DEST[IOD] NORM COND[-USER] LBJUMP[DSK07] $
11m1666
11m1666                                                   .reloc
11m1666                                                   .even
11m1666                                                 ;Read from corresponding register in disk controller.  If not in Exec
11m1666                                                  mode
11m1666                                                 ;and not in IOT User mode, then trap as UUOs.
11m1666                                                 DSKIN0: MAPF[0] START-IN UIOTCK[ILLUUO] $
11m1666                                                          MAPF[0] D[IOD] C500 DEST[AR] MA<20 LBJUMP[SXEOI] $
11m1666                                                 DSKIN1: MAPF[1] START-IN UIOTCK[ILLUUO] $
11m1666                                                          MAPF[1] D[IOD] C500 DEST[AR] MA<20 LBJUMP[SXEOI] $
11m1666                                                 DSKIN2: MAPF[2] START-IN UIOTCK[ILLUUO] $
11m1666                                                          MAPF[2] D[IOD] C500 DEST[AR] MA<20 LBJUMP[SXEOI] $
11m1666                                                 DSKIN3: MAPF[3] START-IN UIOTCK[ILLUUO] $
11m1666                                                          MAPF[3] D[IOD] C500 DEST[AR] MA<20 LBJUMP[SXEOI] $
11m1666
11m1666                                                         .PAIR
11m1666                                                 DSK04:  UIOTCK[ILLUUO] $
```

```
11m1666                                                 DSKO5:  UIOTCK[ILLUUO] $
11m1666                                                         START-OUT D[CONST 10] DEST[DEV-ADDRESS] NORM $
11m1666                                                          MAPF[5] C500 JUMP[MAIN] $
11m1666
11m1666                                                         .PAIR
11m1666                                                 DSKO6:  UIOTCK[ILLUUO] $
11m1666                                                         START-OUT D[CONST 10] DEST[DEV-ADDRESS] NORM $
11m1666                                                          MAPF[6] C500 JUMP[MAIN] $
11m1666
11m1666                                                         .PAIR
11m1666                                                 DSKO7:  UIOTCK[ILLUUO] $
11m1666                                                         START-OUT D[CONST 10] DEST[DEV-ADDRESS] NORM $
11m1666                                                          MAPF[7] C500 JUMP[MAIN] $
11m1666                                                 ];[
11m1666 07632 010731126030300001402046125571457000             START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
11m1666                                                  LBJUMP[DSKIN0] $
11m1666 07633 010731126030300001402046125571457000             START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
11m1666                                                  LBJUMP[DSKIN0] $
11m1666
11m1666 07634 010731126030300001402046125571457000             START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
11m1666                                                  LBJUMP[DSKIN1] $
11m1666 07635 010731126030300001402046125571457000             START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
11m1666                                                  LBJUMP[DSKIN1] $
11m1666
11m1666 07636 010731126030300001402046125571457000             START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
11m1666                                                  LBJUMP[DSKIN2] $
11m1666 07637 010731126030300001402046125571457000             START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
11m1666                                                  LBJUMP[DSKIN2] $
11m1666
11m1666 07640 010731126030300001402046125571457000             START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
11m1666                                                  LBJUMP[DSKIN3] $
11m1666 07641 010731126030300001402046125571457000             START-IN D[CONST 10] DEST[DEV-ADDRESS] NORM COND[-USER]
11m1666                                                  LBJUMP[DSKIN3] $
11m1666
11m1666 07642 030731326000000001416152125471356000             DF/WT D[MEM] DEST[IOD AR] NORM COND[-USER] LBJUMP[DSKO4] $
11m1666 07643 010331326000000001416152105431456000             ALU[MEMAC]   DEST[IOD AR] NORM COND[-USER] LBJUMP[DSKO4] $
11m1666
11m1666 07644 030731126000000001416152125471356000             DF/WT D[MEM] DEST[IOD] NORM COND[-USER] LBJUMP[DSKO5] $
11m1666 07645 010331126000000001416152105431456000             ALU[MEMAC] DEST[IOD] NORM COND[-USER] LBJUMP[DSKO5] $
11m1666
11m1666 07646 030731126000000001416152125471356000             DF/WT D[MEM] DEST[IOD] NORM COND[-USER] LBJUMP[DSKO6] $
11m1666 07647 010331126000000001416152105431456000             ALU[MEMAC] DEST[IOD] NORM COND[-USER] LBJUMP[DSKO6] $
11m1666
11m1666 07650 030731126000000001416152125471356000             DF/WT D[MEM] DEST[IOD] NORM COND[-USER] LBJUMP[DSKO7] $
11m1666 07651 010331126000000001416152105431456000             ALU[MEMAC] DEST[IOD] NORM COND[-USER] LBJUMP[DSKO7] $
11m1666
11m1666                                                  .reloc
11m1666                                                 [.USE[HILOC]
11m1666                                                 [ xlist
11m1667                                                   list ]                                ]  .even
11m1667                                                 [:. \ 2 + .
11m1668                                                   ];Read from corresponding register in disk controller.  If not in Exec
11m1668                                                  mode
11m1668                                                 ;and not in IOT User mode, then trap as UUOs.
11m1668                                                 DSKIN0: MAPF[0] START-IN UIOTCK[ILLUUO]
11m1668 14054 010731006030000001561620246114560000 [       D[PC-FLAGS] ROT[6] COND[-OBUS<0] JUMP[ILLUUO] LONG $
11m1668                                                 ]$
11m1668 14055 010731310000227054161621257214560000              MAPF[0] D[IOD] C500 DEST[AR] MA<20 LBJUMP[SXEOI] $
11m1668                                                 DSKIN1: MAPF[1] START-IN UIOTCK[ILLUUO]
11m1668 14056 010731006130000001561620246114560000 [       D[PC-FLAGS] ROT[6] COND[-OBUS<0] JUMP[ILLUUO] LONG $
11m1668                                                 ]$
11m1668 14057 010731310100227054161621257214560000              MAPF[1] D[IOD] C500 DEST[AR] MA<20 LBJUMP[SXEOI] $
11m1668                                                 DSKIN2: MAPF[2] START-IN UIOTCK[ILLUUO]
11m1668 14060 010731006230000001561620246114560000 [       D[PC-FLAGS] ROT[6] COND[-OBUS<0] JUMP[ILLUUO] LONG $
11m1668                                                 ]$
11m1668 14061 010731310200227054161621257214560000              MAPF[2] D[IOD] C500 DEST[AR] MA<20 LBJUMP[SXEOI] $
11m1668                                                 DSKIN3: MAPF[3] START-IN UIOTCK[ILLUUO]
11m1668 14062 010731006330000001561620246114560000 [       D[PC-FLAGS] ROT[6] COND[-OBUS<0] JUMP[ILLUUO] LONG $
11m1668                                                 ]$
11m1668 14063 010731310300227054161621257214560000              MAPF[3] D[IOD] C500 DEST[AR] MA<20 LBJUMP[SXEOI] $
11m1668
11m1668                                                         .PAIR
11m1668                                                 [:. \ 2 + .
11m1669                                                   JDSKO4:       UIOTCK[ILLUUO]
11m1669 14064 010731006000000000156162024611456000 [       D[PC-FLAGS] ROT[6] COND[-OBUS<0] JUMP[ILLUUO] LONG $
11m1669                                                 ]$
11m1669 14065 010731170031060554020463655714170000             START-OUT D[CONST 10] DEST[DEV-ADDRESS] NORM $
11m1669 14066 010731170400227114161040254214560000              MAPF[4] D[AR] DEST[D%AMEM2] C500 JUMP[MAIN] $
11m1669
11m1669                                                         .PAIR
11m1669                                                 [:. \ 2 + .
11m1670                                                   JDSKO5:       UIOTCK[ILLUUO]
11m1670 14070 010731006000000000156162024611456000 [       D[PC-FLAGS] ROT[6] COND[-OBUS<0] JUMP[ILLUUO] LONG $
11m1670                                                 ]$
11m1670 14071 010731170031060554020463655714170000             START-OUT D[CONST 10] DEST[DEV-ADDRESS] NORM $
11m1670 14072 010731170500227114161620254214560000              MAPF[5] C500 JUMP[MAIN] $
11m1670
11m1670                                                         .PAIR
11m1670                                                 [:. \ 2 + .
11m1671                                                   JDSKO6:       UIOTCK[ILLUUO]
11m1671 14074 010731006000000000156162024611456000 [       D[PC-FLAGS] ROT[6] COND[-OBUS<0] JUMP[ILLUUO] LONG $
11m1671                                                 ]$
11m1671 14075 010731170031060554020463655714170000             START-OUT D[CONST 10] DEST[DEV-ADDRESS] NORM $
11m1671 14076 010731170600227114161620254214560000              MAPF[6] C500 JUMP[MAIN] $
11m1671
11m1671                                                         .PAIR
11m1671                                                 [:. \ 2 + .
11m1672                                                   JDSKO7:       UIOTCK[ILLUUO]
11m1672 14100 010731006000000000156162024611456000 [       D[PC-FLAGS] ROT[6] COND[-OBUS<0] JUMP[ILLUUO] LONG $
11m1672                                                 ]$
11m1672 14101 010731170031060554020463655714170000             START-OUT D[CONST 10] DEST[DEV-ADDRESS] NORM $
11m1672 14102 010731170700227114161620254214560000              MAPF[7] C500 JUMP[MAIN] $
11 1666                                                 ].REPEAT WAITSFIX
11 1667
11 1668 14103 010241170031060554161523654314160000 DSKRST:  SPEC[IOB-OUT] ALU[0] DEST[IOD] NORM $
11 1669                                                          ;SET DSK CTRL COMMAND REGISTER TO 0 (DISABLES INTS).
11 1670 14104 010731170431060554005523655514160000             MAPF[4] SPEC[IOB-OUT] D[CONST 2] DEST[IOD]  CYLEN[IOB-OUT] $
11 1671                                                          ;NOW RESET THE CONTROLLER.
11 1672 14105 010241170700060554161023654114160000             MAPF[7] ALU[0] DEST[D%AMEM1]  CYLEN[IOB-OUT]  $
11 1673                                                          ;ALSO CLEAR THE PI CHANNEL ASSIGNMENT.
```

```
11 1679
11 1680                                                 ;a-mem useage:
11 1681                                                 ;       0       dispatch addr for interrupts
11 1682                                                 ;       1       pi chn (33: 35) and intrpt waiting flag (32)
11 1683                                                 ;       2       copy of last cmd sent to controller (by opcode 721)
11 1684
11m1685                                                   .opcode[740]   [xlist
11 1685                                                    list  ];Disk pseudo-iot dispatch table entries...
11 1686 07700 010731170000000014161620254314560000             JUMP[DCNOA] $
11 1687 07701 010731170000000014161620254314560000             JUMP[DCNOA] $
11 1688
11 1689 07702 010731170000000014161620254314560000             JUMP[DCNIA] $
11 1690 07703 010731170000000014161620254314560000             JUMP[DCNIA] $
11 1691
11 1692 07704 010731170000000014161620254314560000             JUMP[DCNSOA] $
11 1693 07705 010731170000000014161620254314560000             JUMP[DCNSOA] $
11 1694
11 1695 07706 010731170000000014161620254314560000             JUMP[DCNSZA] $
11 1696 07707 010731170000000014161620254314560000             JUMP[DCNSZA] $
11 1697
11m1698                                                   .reloc
11m1698                                                  [.USE[HILOC]
11m1698                                                  [ xlist                                         . #
11 1699                                                    list ]                                         ]; dcono -- 740
11 1700                                                 DCNOA:
11 1701                                                 .REPEAT 1 - WAITSFIX [
11 1702                                                         d[const 10] dest[dev-adr] short $
11 1703                                                 ];.REPEAT 1 - WAITSFIX
11 1704                                                 .REPEAT  WAITSFIX [
11 1705                                                         d[const 10] dest[dev-adr] norm cond[user] pushj[ckiotu] $
11m1706                                                 ];[
11m1706 14110 010731124040000001402046225571457000             d[const 10] dest[dev-adr] norm cond[user] pushj[ckiotu] $
11 1706                                                 ].REPEAT  WAITSFIX
11 1707 14111 010731170000000001400702025771456000             d[ir] mask[3] dest[D%AMEM1] norm jump[dcno1]$
11 1708                                                 ; dconi -- 741
11 1709 14112 010731170040000001402046225571457000    DCNIA:   d[const 10] dest[dev-adr] norm pushj[dcni1] $ ;get bits.
11 1710 14113 010731310000226654161561254304560000             d[ar] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<20] LBJUMP[SEOI] ]$
11 1710                                                  ;store them and return.
11 1711                                                 ; dconso -- 742
11 1712 14114 010731170040000001402046225571457000    DCNSOA: d[const 10] dest[dev-adr] norm pushj[dcni1] $ ;get bits.
11 1713 14115 010641000000000001404562124721456000             d[mask 22] alu[d&q] c550 cond[-obus=0] lbjump[dskp1] $
11 1714                                                 ; dconsz -- 743
11 1715 14116 010731170040000001402046225571457000    DCNSZA: d[const 10] dest[dev-adr] norm pushj[dcni1] $ ;get bits.
11 1716 14117 010641002000000001404562124721456000             d[mask 22] alu[d&q] c550 cond[obus=0] lbjump[dskp1] $
11 1717
11 1718 14120 010731002000227114007620257614560000    dcno1:   d[ir] mask[3]  c500 cond[obus=0] jump[main] $
11 1719                                                          ; if assigned pi channel is not 0, then
11 1720                                                          ;  enable interrupt on "not active", by
11 1721                                                          ;  re-loading last cmd with 10 bit on.
11 1722 14121 010730170000060554041623655714160000             D[CONST 20] DEST[Q] NORM $
11 1723 14122 010641002000000001416162025761456000             D[IR] ALU[D&Q] C550 OBUS=0 JUMP[DCNO2] $
11 1724 14123 010730170000000014101620255714560000             D[CONST 40] DEST[Q] NORM JUMP[DCNO3] $
11 1725 14124 010730170000060554021623655774160000    DCNO2:   d[const 10] dest[q] short $
11 1726 14125 010631170031060554161523661314160000    DCNO3:   d[D%AMEM2] alu[dorq] dest[iod] spec[iob-out] norm $
11 1727 14126 010731170400227114161620254114560000             mapf[4] cylen[iob-out] jump[main] $
11 1728
11 1729 14127 010730370000060554161623660714160000    dcni1:   d[D%AMEM1] dest[q ar] norm $ ;get intrpt flag and pi chn
11 1730                                                 .REPEAT 1 - WAITSFIX [
11 1731                                                         d[ir] alu[d&q] dest[q] norm popj $ ;this is for conso, z
11 1732                                                 ];.REPEAT 1 - WAITSFIX
11 1733                                                 .REPEAT WAITSFIX [
11 1734                                                         d[ir] alu[d&q] dest[q] norm cond[-user] popj $ ;this is for
11 1734                                                  conso, z
11 1735                                                 ;       \ /
11 1736                                                 ;Subroutine to trap if not in EXEC or IOT-User mode
11 1737                                                 CKIOTU: UIOTCK[MUUO] $
11 1738                                                         POPJ $
11 1739
11m1740                                                 ];[
11m1740 14130 010640126000060554161624257714160000             d[ir] alu[d&q] dest[q] norm cond[-user] popj $ ;this is for
11m1740                                                  conso, z
11m1740                                                 ;       \ /
11m1740                                                 ;Subroutine to trap if not in EXEC or IOT-User mode
11m1740                                                 CKIOTU: UIOTCK[MUUO]
11m1740 14131 010731006000102101561620246114560000    [        D[PC-FLAGS] ROT[6] COND[-OBUS<0] JUMP[MUUO] LONG $
11m1740                                                 ]$
11m1740 14132 010731170000060554161624254314160000             POPJ $
11m1740
11 1740                                                 ].REPEAT WAITSFIX
11 1741
11 1742                                                         ;interrupts from disk (dev 10) come here.
11 1743 14133 010770170000060554121623655714160000    DSKINT:  D[CONST 50] ALU[NOTD] DEST[Q] NORM  $
11 1744 14134 010641170031060554161523661314160000             d[D%AMEM2] ALU[D&Q] dest[iod] spec[iob-out] norm  $
11 1745                                                           ;clear interrrupt enable bit (amem[2] has last cmd)
11 1746 14135 010730202400000001416162026061456000             mapf[4] d[D%AMEM1] dest[q ar] c550 cond[obus=0] jump[ddis] $
11 1747 14136 010631170000020343402102025571456000             d[const 10] alu[dorq] dest[D%AMEM1] norm jump[pigen] $
11 1748                                                           ;set flag and request intrpt.
11 1749 14137 010731170040227114161620254314560000    ddis:    clr-dev-from-intr norm jump[main] $
11 1750
11 1751 14140 010731370040000001402046025571457000    DSKWT1: D[CONST 10] DEST[DEV-ADR AR] NORM JUMP[DSKWT4] $
11 1752 14141 010731004000000000776162025461556000    DSKWT3: D[MEM] ROT[31.] C550 COND[OBUS<0] JUMP[DSKWDN] $
11 1753 14142 010731170010060541402003655714170000             D[CONST 1] ROT[LLOAD-ROT + 6] LLOAD NORM $
11 1754 14143 010731070000303074161620254214560000             C550 LOOP[.] $
11 1755 14144 010731170030060554161623654314160000    DSKWT4: START-IN NORM $
11 1756 14145 010731170000303034161500257114560000             MAPF[0] D[IOD] DEST[HOLD] C800  JUMP[DSKWT3] $
11 1757 14146 010721200000303114161620254214560000    DSKWDN: D[AR] ALU[D-1] DEST[AR] C550 COND[-OBUS=0] JUMP[DSKWT4] $
11 1758 14147 010731170000227114161620254314560000             JUMP[MAIN] $
11 1759
11 1760
11 1666                                                 ];DSK
11 1667
11 1668
```

```
12 1669                                         ;TYMNET INTERFACE.
12 1670
12 1671       16700     TYMORG = 16700  ;Where to assemble fixed part of TYMNet code.
12 1672
12 1673                                         ;A-MEM Usage.
12 1674                                         ; 0: Standard.
12 1675                                         ; 1: State of output register... see below
12 1676                                         ; 2: Output coroutine adr (lo 6 bits)
12 1677                                         ; 3: Input coroutine adr (lo 6 bits)
12 1678                                         ; 4: Current 36-bit word being output (for State 1 only)
12 1679                                         ; 5: Current 36-bit word being assembled on input (State 1)
12 1680                                         ; 6: IOWD for packet currently being read
12 1681                                         ; 7: Input reg at intrpt
12 1682
12 1683                                         ;MAIN MEMORY LOCATIONS:
12 1684                                         ; 2000/ KEY:    0               ;FLAGS
12 1685                                         ; 2200/ IRING: BLOCK 200        ;INPUT (TO HOST) RING
12 1686                                         ; 2400/ ORING: BLOCK 400        ;OUTPUT (FROM HOST) RING
12 1687       2005      HIRP = 2005     ;HOST IRING PTR (TAKER)
12 1688       2006      BIRP = 2006     ;BASE IRING PTR (PUTTER)
12 1689       2011      HORP = 2011     ;HOST ORING PTR (PUTTER)
12 1690       2012      BORP = 2012     ;BASE ORING PTR (TAKER)
12 1691
12 1692                                         ;DR11C FLAG NAMES.  (FIELD VALUES).
12 1693                                         ;THE ACTUAL DEFINITIONS ARE IN F4DEF.SLO
12 1694                                         ;TNODIBN=2      ;OUTPUT DONE ROTATE AMOUNT (INPUT) = REQ B.
12 1695                                         ;TNIRIBN=3      ;INPUT READY ROTATE AMOUNT (INPUT) = REQ A.
12 1696                                         ;TNODOBN=6      ;OUTPUT DONE ROTATE AMOUNT (OUTPUT) = CSR1.
12 1697                                         ;TNODOBNPO=7
12 1698                                         ;TNIROBN=7      ;INPUT READY ROTATE AMOUNT (OUTPUT) = CSR0.
12 1699                                         ;TNIROBNPO=10
12 1700
12 1701                                           .repeat 1 - tymnet [
12 1702                                         TYMRST: D[CONST 1] DEST[Q] JUMP[DEVGCL] $
12 1703
12 1704                                                 ILLIOT[744 747]
12 1705                                                   ;These opcodes don't exist if TYMNET code not assembled.
12 1706                                           .reloc
12m1707                                            ];[
12m1707 14150 0107301700000000014003620255714560000    TYMRST: D[CONST 1] DEST[Q] JUMP[DEVGCL] $
12m1707
12m1707                                                 ILLIOT[744 747]
12m1707                                         [
12m1707                                           .opcode[744]
12m1707                                         [xlist
12m1708                                           list  ]   .repeat (1 + 747 - 744) [
12m1708                                                 COND[-USER] LBJUMP[PANIOT] $
12m1708                                                 COND[-USER] LBJUMP[PANIOT] $
12m1708                                               ]
12m1708                                         [
12m1708 07710 0107311260002542541616212543145600000            COND[-USER] LBJUMP[PANIOT] $
12m1708 07711 0107311260002542541616212543145600000            COND[-USER] LBJUMP[PANIOT] $
12m1708                                               ][
12m1708 07712 0107311260002542541616212543145600000            COND[-USER] LBJUMP[PANIOT] $
12m1708 07713 0107311260002542541616212543145600000            COND[-USER] LBJUMP[PANIOT] $
12m1708                                               ][
12m1708 07714 0107311260002542541616212543145600000            COND[-USER] LBJUMP[PANIOT] $
12m1708 07715 0107311260002542541616212543145600000            COND[-USER] LBJUMP[PANIOT] $
12m1708                                               ][
12m1708 07716 0107311260002542541616212543145600000            COND[-USER] LBJUMP[PANIOT] $
12m1708 07717 0107311260002542541616212543145600000            COND[-USER] LBJUMP[PANIOT] $
12 1708                                               ] ]       ;These opcodes don't exist if TYMNET code not
12 1708                                         assembled.
12 1709                                           .reloc
12m1710                                         [.USE[HILOC]
12m1710                                         [ xlist
12 1707                                           list ]                                ]    ] 1 - tymnet
12 1708
12 1709
12 1710
12 1711                                         .repeat TYMNET [
12 1712
12 1713
12 1714
12 1715                                         TYMRST:   ;Reset Tymnet interface
12 1716                                                 D[CONST 1] DEST[Q] PUSHJ[DEVGCL] $
12 1717                                                 D[LABEL TYMINT] DEST[D%AMEM0] $
12 1718                                                 JUMP[TYMRS1] NORM $
12 1719                                                   ;Set initial coroutine addrs, intrpt enbls.
12 1720
12 1721
12 1722                                         ;IO subselects (for Dev. 6) for talking to the DR-11 in the Tymnet Base:
12 1722
12 1723                                          ;Output:
12 1724                                         ];.REPEAT TYMNET
12 1725       15        TYM.LD = 15     ;Load output register, both data and flags.
12 1726                                         .REPEAT TYMNET [
12 1727                                         TNIRIFS = 16    ;Clear the REQ B EDGE flag (which means Base has sent us
12 1727                                          data)
12 1728                                         TNODIFS = 17    ;Clear the REQ B EDGE flag (which means Base has taken
12 1728                                          data)
12 1729                                          ;Input:
12 1730                                         TYM.RD = 5      ;Read input data and flags.
12 1731
12 1732
12 1733
12 1734                                         .define TNOCEC[] [ ;Send 0 byte if Escape Word (-1) just sent.
12 1735                                                 D[D%AMEM4] ROT[34] MASK[20] DEST[Q] NORM $
12 1736                                                 D[MASK 20] ALU[D#Q] COND[-OBUS=0] JUMP[. + 3] C550 $
12 1737                                                 ALU[0] DEST[Q] PUSHJ[TNSEND] NORM $
12 1738                                                 D[CONST[. & 77 + 1]] DEST[D%AMEM2] JUMP[TNI0] NORM $
12 1739                                           ]
12 1740                                         .define GBORP[] [
12 1741                                                 D[LIT BORP] DEST[Q] $
12 1742                                           ]
12 1743                                         .define GHORP[] [
12 1744                                                 D[CONST 11] DEST[Q] PUSHJ[TNRCW] NORM $
12 1745                                           ]
12 1746                                         .define GBIRP[] [
12 1747                                                 D[CONST 6] DEST[Q] PUSHJ[TNRCW] NORM $
```

```
12 1753
12 1754                                         ;HERE TO SERVICE INPUT READY OR OUTPUT DONE INTERRUPT BY
12 1755                                         ; TYMNET INTERFACE.
12 1756
12 1757                                         TYMINT: START-IN D[CONST 6] DEST[DEV-ADDRESS] $
12 1758                                                 MAPF[TYM.RD] D[IOD] DEST[Q D%AMEM7] CY-IO-IN
12 1758                                          PUSHJ[MEMADR-23BIT-ABS] $
12 1759                                                         ;Turn off memory mapping, and save input reg in A-MEM[7].
12 1759
12 1760                                                 D[CONST 1] ROT[TNODIBN] ALU[D&Q] COND[OBUS=0] JUMP[TNI1] C550 $
12 1761                                                         ;Output done bit on ?
12 1762                                                 START-OUT D[D%AMEM2] DEST[Q] $
12 1763                                                         ;Yes. Clear it, and get dispatch index.
12 1764                                                 MAPF[TNODIFS] ALU[Q] COND[OBUS<0] JUMP[TNI0]  C550 $
12 1765                                                         ;Test for initial (no disp.) state.
12 1766                                                 D[LIT TYMORG] ALU[DORQ] DEST[HOLD] $
12 1767                                                 D[MEM] ROT[MUA-ROT] LONG  ODISP  $
12 1768                                                         ;Form dispatch address.
12 1769                                         TNI0:   D[D%AMEM7] DEST[Q] NORM $
12 1770                                                         ;Recover contents of input register at time of intrpt.
12 1771                                         TNI1:   D[CONST 1] ROT[TNIRIBN] ALU[D&Q] COND[OBUS=0] JUMP[TNI2] C550 $
12 1772                                                         ;Input ready bit on ?
12 1773                                                 START-OUT D[D%AMEM7] ROT[34] MASK[20] DEST[AR] $
12 1774                                                         ;Yes. Clear it, and extract data bits from input
12 1774                                          register.
12 1775                                                 MAPF[TNIRIFS]
12 1776                                                  D[D%AMEM3] DEST[Q] COND[OBUS18] JUMP[TNI2] C500 $
12 1777                                                   ;Get dispatch index, test for initial (no disp.) state.
12 1778                                                 D[LIT TYMORG] ALU[DORQ] DEST[HOLD] $
12 1779                                                 D[MEM] ROT[MUA-ROT] LONG  ODISP  $
12 1780                                                         ;Dispatch.
12 1781                                         TNI2:   MAP-ENABLE JUMP[MAIN] $
12 1782                                                  ;Return from interrupt.
12 1783
12 1784                                         ;VARIOUS SUBROUTINES.
12 1785
12 1786                                           ;Q loaded with 16. bit word to send, shifted left by 8.
12 1787                                         TNSEND: ALU[Q] DEST[AR] NORM $
12 1788                                                 D[AR] ROT[34] DEST[Q] $
12 1789                                                 D[MASK 20] ALU[D&Q] DEST[Q AR] NORM  $
12 1790
12 1791                                         .repeat 0 [ ;This is a debugging kluge !
12 1792                                           D[MA] DEST[55] $
12 1793                                           D[LIT 103077] DEST[MA] $
12 1794                                           DF/WT D[MEM] ALU[D+1] DEST[Q] $
12 1795                                           D[MASK 6] ALU[D&Q] DEST[MEMSTO Q] $
12 1796                                           D[LIT 103100] ALU[D+Q] DEST[MA] $
12 1797                                           D[AR] DEST[MEMSTO] $
12 1798                                           D[55] DEST[MA] $
12 1799                                          ]
12 1800                                                 D[AR] ROT[8] DEST[Q] SHORT $
12 1801                                                 D[MASK 16.] ROT[8] ALU[D&Q] DEST[AR] NORM $
12 1802                                                 D[MASK 8] DEST[Q] PUSHJ[DEV6ST1] NORM $
12 1803                                             ;Now clear, then set the Input Ready flag to DR11C.
12 1804                                                 D[MASK 43] ROT[TNIROBNPO] DEST[Q] PUSHJ[DEV6CL] NORM $
12 1805                                                 D[CONST 1] ROT[TNIROBN] DEST[AR] NORM JUMP[DEV6ST2] $
12 1806
12 1807
12 1808                                           ;No output to do.  Do not change TNIROBN, set A-MEM flag for
12 1809                                           ; clock routine.
12 1810                                         TNORE:  ALU[-1] DEST[D%AMEM2] JUMP[TNI0] NORM $
12 1811
12 1812                                           ;Read word from base-host communication area, location 20xx;
12 1813                                           ; Q has xx.
12 1814                                         TNRCW:  D[LIT 2000] ALU[DORQ] DEST[MA] NORM $
12 1815                                                 DFRQ POPJ $
12 1816
12 1817                                           ;Here when input ring full.  Set clock flag and return without
12 1818                                           ; setting TNODOBN.
12 1819                                         TNIRF:  ALU[-1] DEST[D%AMEM3] JUMP[TNI2] NORM $
12 1820
12 1821                                           ;Clear then set TNODOBN.
12 1822                                         TNCASODOF:
12 1823                                                 D[MASK 43] ROT[TNODOBNPO] DEST[Q] PUSHJ[DEV6CL] NORM $
12 1824                                                 D[CONST 1] ROT[TNODOBN] DEST[AR] NORM JUMP[DEV6ST2] $
12 1825
12 1826
12 1827
12 1828                                           .org[TYMORG]
12 1829
12 1830                                         ;********** OUTPUT DONE CO-ROUTINE.
12 1831
12 1832
12 1833                                           ;A-MEM[4] = Rotated 36-bit word with -11 word just sent in 12-27.
12 1834                                           ;State 0: Fetch next 36-bit word or quit if ring empty.
12 1835                                         TNOC0:  D[LIT BORP] DEST[MA] $
12 1836                                                 DF/WT D[MEM] DEST[D%AMEM17 AR] NORM $
12 1837                                         TNOC0A: GHORP
12 1838                                                 D[AR] DEST[Q] NORM $
12 1839                                                 D[MEM] ALU[D-Q] COND[OBUS=0] JUMP[TNORE] C600 $
12 1840                                                 D[CONST 24] ROT[6] ALU[D+Q] DEST[MA] NORM $
12 1841                                                 DFRQ $
12 1842                                                 D[MEM] ROT[30] DEST[D%AMEM4 Q] PUSHJ[TNSEND] NORM $
12 1843                                                 D[CONST[. & 77 + 1]] DEST[D%AMEM2] JUMP[TNI0] NORM $
12 1844
12 1845                                           ;Clock restarts coroutine here.
12 1846                                         TNOC2:  TNOCEC
12 1847                                           ;State 1: Send low order half of 36-bit word.
12 1848                                                 D[D%AMEM4] ROT[20] DEST[Q] NORM $
12 1849                                                 ALU[Q] DEST[D%AMEM4] PUSHJ[TNSEND] NORM $
12 1850                                                 D[CONST[. & 77 + 1]] DEST[D%AMEM2] JUMP[TNI0] NORM $
12 1851                                                 TNOCEC
12 1852                                                 D[LIT BORP] DEST[MA] $
12 1853                                                 DF/WT D[MEM] DEST[D%AMEM17] $
12 1854                                                 D[MEM] ALU[D+1] DEST[Q] NORM $
12 1855                                                 D[MASK 8] ALU[D&Q] DEST[AR MEMSTO] JUMP[TNOC0A] $
12 1856
12 1857
12 1858                                         ;********** INPUT READY CO-ROUTINE.
12 1859
```

```
12 1865                                                 D[AR] DEST[Q D%AMEM5] NORM $
12 1866                                                 D[CONST 1] ROT[17] ALU[D&Q] COND[OBUS=0] JUMP[TNIC0A] C550 $
12 1867                                               ;DATA PACKET MESSAGE.
12 1868                                                 D[AR] ROT[34] MASK[7] DEST[Q] SHORT $
12 1869                                                 D[CONST 5] ALU[D+Q] DEST[AR] SHORT $
12 1870                                                 D[AR] ROT[42] MASK[18.] DEST[D%AMEM6] NORM $
12 1871                                                 D[D%AMEM5] DEST[AR] SHORT $
12 1872                                                ;Read msg.  A-MEM[6]:=# 36-bit words in message.
12 1873                                                ; AR:=first -11 word in message.
12 1874                                               TNICRM:
12 1875                                                 GBIRP
12 1876                                                 D[D%AMEM6] ROT[18.] ALU[0-D] DEST[Q] NORM $
12 1877                                                 D[MEM] MASK[18.] ALU[QORD] DEST[D%AMEM6] NORM $
12 1878                                               ;STATE 1: HO 16 BITS OF WORD ARRIVES.
12 1879                                               TNIC1:  D[AR] ROT[24] DEST[D%AMEM5] NORM $
12 1880                                                 GHIRP
12 1881                                                 D[D%AMEM6] ALU[D+1] DEST[Q] NORM $
12 1882                                                 D[MASK 7] ALU[D&Q] DEST[Q] SHORT $
12 1883                                                 D[MEM] ALU[D-Q] COND[OBUS=0] JUMP[TNIRF] C600 $
12 1884                                                 PUSHJ[TNCASDDOF] NORM $
12 1885                                                 D[CONST[. & 77 + 1]] DEST[D%AMEM3] JUMP[TNI2] NORM $
12 1886                                               ;STATE 2: LO 16 BITS OF WORD ARRIVES.
12 1887                                               ; ALSO STARTED HERE BY CLOCK ROUTINE.
12 1888                                               TNIC2:  D[D%AMEM6] MASK[7] DEST[Q] NORM JUMP[TNIC2A] $
12 1889
12 1890
12 1891                                                .reloc
12 1892
12 1893                                               TNIC2A: D[CONST 22] ROT[6] ALU[D+Q] DEST[MA] NORM $
12 1894                                                 D[D%AMEM5] DEST[Q] NORM $
12 1895                                                 D[AR] ROT[4] ALU[DORQ] DEST[MEMSTO] NORM $
12 1896                                                 PUSHJ[TNCASDDOF] NORM $
12 1897                                                 D[D%AMEM6] ALU[D+1] DEST[Q] $
12 1898                                                 D[CONST 1] ROT[18.] ALU[D+Q] DEST[D%AMEM6] COND[-OBUS<0]
12 1898                                                JUMP[TNICDN] C600 $
12 1899                                                 D[CONST[?? & TNIC1]] DEST[D%AMEM3] JUMP[TNI2] NORM $
12 1900
12 1901                                               TNICDN: D[CONST 6] DEST[Q] SHORT $
12 1902                                                 D[CONST 20] ROT[6] ALU[DORQ] DEST[MA] SHORT $
12 1903                                                 D[D%AMEM6] MASK[7] DEST[MEMSTO] NORM $
12 1904                                                 D[CONST[TNIC0 & 77]] DEST[D%AMEM3] JUMP[TNI2] $
12 1905
12 1906                                                ;1 wd or 2 wd or escape message.
12 1907                                               TNIC0A: D[CONST 1] ROT[16] ALU[D&Q] COND[-OBUS=0] JUMP[TNIC0B] C550 $
12 1908                                                ;One word packet.
12 1909                                                 D[CONST 1] DEST[D%AMEM6] JUMP[TNICRM] NORM $
12 1910                                               TNIC0B: D[CONST 1] ROT[16] ALU[D#Q] DEST[Q AR] SHORT $
12 1911                                                 D[MASK 6] ROT[8] ALU[D&Q] DEST[Q] SHORT $
12 1912                                                 D[MASK 6] ROT[8] ALU[D#Q] COND[OBUS=0] JUMP[TNIC0C] C550 $
12 1913                                                ;2 word packet.
12 1914                                                 D[CONST 2] DEST[D%AMEM6] JUMP[TNICRM] NORM $
12 1915                                                ;Escape sequence.  Better not happen.
12 1916                                               TNIC0C: JUMP[.] $
12 1917
12 1918                                               ;CLOCK ROUTINE, CALLED EVERY TICK OF 60HZ CLOCK.
12 1919
12 1920                                               TYMCLK: D[CONST 6] DEST[DEV-ADR] PUSHJ[MEMADR-23BIT-ABS] $
12 1921                                                  ;Select device, enter 23-bit absolute memory address mode.
12 1922                                                 D[D%AMEM3] COND[OBUS<0] PUSHJ[TNSI] C550 $
12 1923                                                 D[D%AMEM2] COND[OBUS<0] PUSHJ[TNSO] C550 $
12 1924                                                 JUMP[MEMADR-18BIT] $
12 1925                                                  ;Restore to 18-bit address mode and return.
12 1926                                               TNSI:   GHIRP
12 1927                                                 D[D%AMEM6] ALU[D+1] DEST[Q] NORM $
12 1928                                                 D[MASK 7] ALU[D&Q] DEST[Q] SHORT $
12 1929                                                 D[MEM] ALU[D-Q] COND[OBUS=0] POPJ C600 $
12 1930                                                 D[CONST[TNIC2 & 77]] DEST[D%AMEM3] JUMP[TNCASDDOF] NORM $
12 1931
12 1932                                               TNSO:   D[LIT BORP] DEST[MA] $
12 1933                                                 DFRQ D[LIT HORP] DEST[MA] $
12 1934                                                 DFRQ D[MEM] MASK[17.] DEST[Q D%AMEM17] NORM $
12 1935                                                 D[MEM] MASK[17.] ALU[D-Q] COND[OBUS=0] POPJ C600 $
12 1936                                                 D[LIT 2400]  ALU[D+Q] DEST[MA] NORM $
12 1937                                                 DF/WT D[MEM] ROT[30] DEST[D%AMEM4 Q] PUSHJ[TNSEND] NORM $
12 1938                                                 D[CONST[TNOC2 & 77]] DEST[D%AMEM2] POPJ NORM $
12 1939
12 1940                                               ;RESET TYMNET INTERFACE.  OUTPUT COROUTINE GOES INTO NULL STATE.
12 1941                                               ; INPUT COROUTINE GOES INTO STATE 0, INTRPTS GET ENABLED.
12 1942                                               TYMRS1: D[LABEL TYMINT] DEST[D%AMEM0] $
12 1943                                                  ;Set up intrpt. dispatch address.
12 1944                                                 D[CONST 3] ROT[2] DEST[AR] CYLEN[IOB-OUT] SPEC[IOB-OUT] $
12 1945                                                  ;Get const to enb. intrpts; IOB-OUT clears flags.
12 1946                                               TYMRS2: MAPF[TNODIFS] START-OUT C600 $
12 1947                                                 MAPF[TNIRIFS] D[MASK 2] ALU[NOTD] ROT[2] DEST[Q] PUSHJ[DEV6ST1]
12 1947                                                C600 $
12 1948
12 1949                                               ;NOW RESET PTRS TO MEAN RINGS EMPTY FOR INPUT AND OUTPUT WITHOUT REGARD
12 1950                                               ;TO PREVIOUS STATE FOR NOW
12 1951
12 1952                                                 MAP-DISABLE $
12 1953                                                 D[LIT HIRP] DEST[MA] NORM $
12 1954                                                 ALU[0] DEST[MEMSTO] NORM $
12 1955                                                 D[LIT BIRP] DEST[MA] NORM $
12 1956                                                 ALU[0] DEST[MEMSTO] NORM $
12 1957                                                 D[LIT HORP] DEST[MA] NORM $
12 1958                                                 ALU[0] DEST[MEMSTO] NORM $
12 1959                                                 D[LIT BORP] DEST[MA] NORM $
12 1960                                                 ALU[0] DEST[MEMSTO] NORM $
12 1961
12 1962                                                 ALU[-1] DEST[D%AMEM2] NORM $
12 1963                                                 MAP-ENABLE D[CONST[TNIC0 & 77]] DEST[D%AMEM3] POPJ NORM $
12 1964
12 1965                                               ;;;TYMAREA:     ;SAVE LOCATION
12 1966
12 1967                                                .opcode[744]    ;RESET TYMNET
12 1968
12 1969                                                 D[CONST 6] DEST[DEV-ADR] NORM JUMP[TYMNRESET] $
12 1970                                                 D[CONST 6] DEST[DEV-ADR] NORM JUMP[TYMNRESET] $
12 1971
12 1972                                                 D[CONST 6] DEST[DEV-ADR] NORM JUMP[TYMNOUT] $
```

```
12 1978                                                 D[CONST 6] DEST[DEV-ADR] NORM JUMP[FSDIAG] $
12 1979                                                 D[CONST 6] DEST[DEV-ADR] NORM JUMP[FSDIAG] $
12 1980
12 1981                                               .reloc
12 1982
12 1983                                               TYMNRESET:
12 1984                                                 SPEC[IOB-OUT] D[MA] ROT[2] DEST[AR] NORM PUSHJ[TYMRS2] $
12 1985                                                         ;Reset co-routines, enb. intrpts from eff. addr. 34,35
12 1986                                                 JUMP[MAIN] $
12 1987
12 1988                                               TYMNOUT:SPEC[IOB-OUT] NORM $
12 1989                                                         ;Clear "RDY FOR DATA" from PDPf1.
12 1990                                                 MAPF[TNODIFS] D[IR] ROT[8.] DEST[Q] NORM PUSHJ[TNSEND] $
12 1991                                                         ;Send eff. addr. 20-35 to PDP11.
12 1992                                                 JUMP[MAIN] $
12 1993
12 1994                                               TYMNIN: SPEC[IOB-IN] NORM $
12 1995                                                 MAPF[TYM.RD] D[IOD] ROT[27. + 1] DEST[AR] C800 $
12 1996                                                    ;Align input word with 16 data bits at RIGHT end.
12 1997                                                 D[AR] ROT[4] CS50 COND[SIGNOFF] JUMP[. + 2] $
12 1998                                                    ;Don't clear "INPUT RDY" flag unless it's on.
12 1999                                                  SPEC[IOB-OUT] SHORT $
12 2000                                                    ;It's on.
12 2001                                                 MAPF[TNIRIFS] D[AR] SMAC $
12 2002
12 2003                                               FSDIAG:   ;Special output code for F5 diagnostic interface hardware.
12 2004                                                 DFRQ D[IR] ALU[NOTD] DEST[AR] NORM $
12 2005                                                         ;Fetch C(E), set AR=-IR
12 2006                                                 D[AR] ROT[13.] MASK[2] DEST[AR] NORM $
12 2007                                                         ;AR=-AC MASK 2
12 2008                                                 D[AR] ROT[6] DEST[Q] NORM $
12 2009                                                         ;Q=INVERTED 2 LSB'S OF AC SHIFTED INTO STROBES
12 2010                                                 D[MEM] ROT[8.] ALU[DORQ] DEST[HOLD] NORM $
12 2011                                                         ;DATA SHIFTED WITH STROBES INSERTED
12 2012                                                 D[MEM] DEST[AR] NORM $
12 2013                                                         ;DATA=(E) (LSH 8) (STB ACTIVE LOW)
12 2014                                                 D[MASK 18.] ROT[18. + 6] DEST[Q] PUSHJ[DEV6ST1] NORM $
12 2015                                                         ;LEAVE BITS 0-11. AND 30.-35. UNCHANGED
12 2016                                                 D[MASK 2] ROT[6] DEST[Q] NORM $
12 2017                                                         ;MASK TO FORCE STB'S HIGH
12 2018                                                 D[MEM] ALU[DORQ] DEST[AR] NORM $
12 2019                                                         ;DATA=(E) (LSH 8) (STB'S FORCED INACTIVE HIGH)
12 2020                                                 D[MASK 18.] ROT[18. + 6] DEST[Q] PUSHJ[DEV6ST1] NORM $
12 2021                                                         ;LEAVE BITS 0-11. AND 30.-35. UNCHANGED
12 2022                                                 ALU[0] DEST[DEV-ADR] JUMP[MAIN] $
12 2023
12 2024
12 2025                                               ];tymnet
12 2026
12 2027
```

```
13 2028                                                 ;Device 6, which is mainly Tymnet (DR-11) interface, but also 50Hz Clock.
13 2028
13 2029
13 2030                                                 DEV6CL:   ;Clear some Dev 6 bits (see comment below).
13 2031 14151 010241370000000001416162025431456000             ALU[0] DEST[AR] NORM JUMP[DEV6ST] $
13 2032
13 2033                                                 DEV6ST:     ;Set some of the bits in Device 6's output register, a copy
13 2034                                                             ;of which is maintained in A-MEM(1).
13 2035                                                             ;Enter with mask for UNCHANGED bits in in Q, data in AR.
13 2036                                                             ;On return, Q has new value of AMEM(1).
13 2037 14152 010731170040060554014463655714170000             D[CONST 6] DEST[DEV-ADR] NORM $
13 2038                                                 DEV6ST1:    ;Enter here if you've already set DEVADR:=6
13 2039 14153 010640170000060554161623660714160000             D[D%AMEM1] ALU[D&Q] DEST[Q] NORM $
13 2040                                                             ;Get AMEM(1), mask it.
13 2041 14154 010630170031060554161523654314160000     DEV6STZ: D[AR] ALU[DORQ] DEST[Q IOD] SPEC[IOB-OUT] NORM $
13 2042                                                             ;OR in new data.
13 2043 14155 010231171500060554161024254114160000             MAPF[TYM.LD] ALU[Q] DEST[D%AMEM1] CYLEN[IOB-OUT] POPJ $
13 2044
13 2045
```

```
14 2046                                                 ;Interval Timer code.
14 2047
14 2048                                                 ;A-MEM Useage:
14 2049                                                 ;       0       dispatch addr for interrupts
14 2050                                                 ;       1       pi chn (33: 35) and intrpt waiting flag (32)
14 2051                                                 ;       3       INTERVAL TIMER REGISTER
14 2052                                                 ;       4       Maximum # of usecs allowable for a Timer Int to get
14 2052                                                  serviced...
14 2053                                                 ;                this is reset to infinity, but can be made smaller as a
14 2053                                                 ;                bugtrap to catch cases of the cpu not servicing I/O
14 2054                                                  intrpts
14 2054                                                 ;                in reasonable times.  TIMINT goes to HLTLOC with 17 in
14 2055                                                  AR
14 2055                                                 ;                if the hardware timer register contains a # larger than
14 2056                                                  this
14 2056                                                 ;                when A-MEM[3] overflows.
14 2057
14 2058
14 2059       11                TIM.DO = 11
14 2060       1         TIM.DI = 1
14 2061       10                TIM.ENB = 10
14 2062       5         TIM.DEV = 5
14 2063
14 2064                                                    .REPEAT 1 - TIMER [
14 2065                                                 TIMINT: START-OUT D[CONST 0] DEST[IOD] NORM $
14 2066                                                            ;Clear the TIMER OVERFLOW flag, disable timer intrpts.
14 2067                                                         MAPF[TIM.ENB]   CLR-DEV-FROM-INTR JUMP[MAIN] $
14 2068                                                            ;Flush interrupt.
14 2069                                                      ]
14 2070
14 2071
14 2072                                                  .REPEAT TIMER [
14 2073
14 2074                                                 TIMINT:
14 2075                                                         START-OUT D[CONST 1] DEST[IOD] NORM $
14 2076                                                            ;Clear the TIMER OVERFLOW FLAG.
14 2077                                                         MAPF[TIM.ENB] D[D%AMEM3] DEST[Q] C800 $
14 2078                                                            ;Get TIMER REGISTER.
14 2079                                                         D[CONST 1] ROT[12.] ALU[D+Q] DEST[Q D%AMEM3] CS50 -CRY0
14 2079                                                  JUMP[MAIN] $
14 2080                                                            ;Increment the count by 2↑12, exit if no overflow.
14 2081                                                         PUSHJ[TDTI] NORM $
14 2082                                                            ;Get contents of hardware timer reg.
14 2083                                                         D[D%AMEM4] ALU[D-Q] LONG COND[SIGNOFF] JUMP[. + 2] $
14 2084                                                            ;Jump if it has not been too long since it overflowed.
14 2085                                                          D[CONST 17] DEST[AR] JUMP[HLTLOC] $
14 2086                                                            ;Halt indicating "I/O intrpt service took too long".
14 2087
14 2088                                                         D[D%AMEM1] MASK[3] DEST[AR] CS50 COND[OBUS=0] JUMP[MAIN] $
14 2089                                                            ;Timer overflowed. ignore if no PI CHANNEL assigned.
14 2090                                                         D[D%AMEM1] DEST[Q] NORM $
14 2091                                                            ;Get full contents of A-MEM[1].
14 2092                                                         D[CONST 10] ALU[DORQ] DEST[D%AMEM1] NORM JUMP[PIGEN] $
14 2093                                                            ;Set the "interrupting" status bit and generate a PI.
14 2094
14m2095                                                    ] [
14m2095
14m2095                                                 TIMINT:
14m2095 14156 010731170031060554003523655714116000             START-OUT D[CONST 1] DEST[IOD] NORM $
14m2095                                                            ;Clear the TIMER OVERFLOW FLAG.
14m2095 14157 010730171000060554161623661514116000             MAPF[TIM.ENB] D[D%AMEM3] DEST[Q] C800 $
14m2095                                                            ;Get TIMER REGISTER.
14m2095 14160 010600032000227103003060255614560000             D[CONST 1] ROT[12.] ALU[D+Q] DEST[Q D%AMEM3] CS50 -CRY0
14m2095                                                  JUMP[MAIN] $
14m2095                                                            ;Increment the count by 2↑12, exit if no overflow.
14m2095 14161 010731170000000014161622254314560000             PUSHJ[TDTI] NORM $
14m2095                                                            ;Get contents of hardware timer reg.
14m2095 14162 011621006000303514161620262114560000             D[D%AMEM4] ALU[D-Q] LONG COND[SIGNOFF] JUMP[. + 2] $
14m2095                                                            ;Jump if it has not been too long since it overflowed.
14m2095 14163 010731370000224234037620255714560000              D[CONST 17] DEST[AR] JUMP[HLTLOC] $
14m2095                                                            ;Halt indicating "I/O intrpt service took too long".
14m2095
14m2095 14164 010731202000227114007620260614560000             D[D%AMEM1] MASK[3] DEST[AR] CS50 COND[OBUS=0] JUMP[MAIN] $
14m2095                                                            ;Timer overflowed. ignore if no PI CHANNEL assigned.
14m2095 14165 010730170000060554161623660714116000             D[D%AMEM1] DEST[Q] NORM $
14m2095                                                            ;Get full contents of A-MEM[1].
14m2095 14166 010631170000203434021020255714560000             D[CONST 10] ALU[DORQ] DEST[D%AMEM1] NORM JUMP[PIGEN] $
14m2095                                                            ;Set the "interrupting" status bit and generate a PI.
14m2095
14 2095                                                    ];end REPEAT TIMER
14 2096
14m2097                                                    .opcode[760]   [xlist
14 2097                                                     list  ];TIMER PSEUDO-IOT DISPATCH TABLE ENTRIES.
14 2098
14 2099                                                 .REPEAT 1 - WAITS [
14 2100
14 2101                                                         JUMP[TCNOA] $
14 2102                                                         JUMP[TCNOA] $
14 2103
14 2104                                                         JUMP[TCNIA] $
14 2105                                                         JUMP[TCNIA] $
14 2106
14 2107                                                         JUMP[TCNSOA] $
14 2108                                                         JUMP[TCNSOA] $
14 2109
14 2110                                                         JUMP[TCNSZA] $
14 2111                                                         JUMP[TCNSZA] $
14 2112
14 2113                                                         JUMP[TDTAOA] $
14 2114                                                         JUMP[TDTAOA] $
14 2115
14 2116                                                         D[CONST TIM.DEV] DEST[DEV-ADR] JUMP[TDTAIA] $
14 2117                                                         D[CONST TIM.DEV] DEST[DEV-ADR] JUMP[TDTAIA] $
14 2118
14 2119                                                    .reloc
14 2120                                                 ];.REPEAT 1 - WAITS
14 2121                                                 .REPEAT WAITS [
14 2122                                                         JUMP[TIMIOT] NORM $
14 2123                                                         JUMP[TIMIOT] NORM $
14 2124                                                         .reloc
```

```
14 2130                                                         ;machine has already done indexing/indirection and bits
14 2131                                                         ;13:17 are guaranteed zero
14 2132                                                 D[LABEL TIMER-DISPATCH] ALU[D+Q] DEST[AR] NORM $
14 2133                                                 D[AR] ROT[MUA-ROT] CDISP LONG $
14 2134                                                         ;Dispatch of type of IOT
14 2135
14 2136                                                 ;Allocate space for dispatch table
14 2137                                                         .PAIR   ;Limit address to 12 bits in width
14 2138                                                 TIMER-DISPATCH:
14 2139                                                 :. + 20
14 2140                                                         .ORG [TIMER-DISPATCH]
14 2141
14 2142                                                         ILGIOT $                ;BLKO
14 2143                                                         NORM $
14 2144
14m2145                                                 ];[
14m2145 07740 010731170000000001416162025431456000             JUMP[TIMIOT] NORM $
14m2145 07741 010731170000000001416162025431456000             JUMP[TIMIOT] NORM $
14m2145                                                         .reloc
14m2145                                                  [.USE[HILOC]
14m2145                                                  [ xlist
14m2146                                                    list ]                               ]
14m2146                                                 TIMIOT: CLR-DEV-FROM-INTR D[IR] ROT[12. + 1 + 1] MASK[4] DEST[Q]
14m2146 14167 010730124040302623411622257714456000                     COND[USER] PUSHJ[CKIOTU] $
14m2146                                                         ;Treat device 370 and 374 as identical for now...
14m2146                                                         ;Extract IOT decode * 2.  Note we can do this because the
14m2146
14m2146                                                         ;machine has already done indexing/indirection and bits
14m2146                                                         ;13:17 are guaranteed zero
14m2146 14170 010601370000000000000162365531416000             D[LABEL TIMER-DISPATCH] ALU[D+Q] DEST[AR] NORM $
14m2146 14171 010731170000000000160007254114456000             D[AR] ROT[MUA-ROT] CDISP LONG $
14m2146                                                         ;Dispatch of type of IOT
14m2146
14m2146                                                 ;Allocate space for dispatch table
14m2146                                                         .PAIR    [:. \ 2 + .
14m2146                                                    ];Limit address to 12 bits in width
14m2146                                                 TIMER-DISPATCH:
14m2146                                                 :. + 20
14m2146                                                         .ORG [TIMER-DISPATCH]
14m2146                                                  [ xlist
14m2147                                                    list ]
14m2147 14172 540731170000060554161623254204160000             ILGIOT $                ;BLKO
14m2147 14173 010731170000060554161623654314160000             NORM $
14m2147
14 2145                                                 ].REPEAT WAITS
14 2146
14 2147                                                 .repeat waitsfix [;This conditional is sole for ordering.
14 2148                                                 ; TDATAO -- 764  LOAD THE 36-BIT TIMER
14 2149                                                 TDTAOA: D[CONST TIM.DEV] DEST[DEV-ADR] NORM $
14 2150                                                         DFRQ ALU[MEMAC] DEST[HOLD]  JUMP[TDATO] $
14 2151
14 2152                                                 .REPEAT WAITS [
14 2153                                                         ILGIOT $                ;BLKI
14 2154                                                         NORM $
14 2155                                                 ];.REPEAT WAITS
14 2156
14 2157                                                 ; TDATAI  --  765    READ TIMER REGISTER
14 2158                                                 TDTAIA:
14 2159                                                 .REPEAT WAITSFIX [;Is there some reason this would lose?
14 2160                                                         START-IN D[CONST TIM.DEV] DEST[DEV-ADDRESS] NORM PUSHJ[TDTI] $
14 2161                                                                 ;get bits.
14 2162                                                 ];.REPEAT WAITSFIX
14 2163                                                 .REPEAT 1 - WAITSFIX [
14 2164                                                         START-IN NORM PUSHJ[TDTI] $ ;get bits.
14 2165                                                 ];.REPEAT 1 - WAITSFIX
14 2166                                                         D[AR] SMAC $ ;Store them and return.
14m2167                                                 ];[;This conditional is sole for ordering.
14m2167                                                 ; TDATAO -- 764  LOAD THE 36-BIT TIMER
14m2167 14174 010731170040060554012463655714417000     TDTAOA: D[CONST TIM.DEV] DEST[DEV-ADR] NORM $
14m2167 14175 010331170000000001416150005431256000             DFRQ ALU[MEMAC] DEST[HOLD]  JUMP[TDATO] $
14m2167
14m2167                                                 .REPEAT WAITS [
14m2167                                                         ILGIOT $                ;BLKI
14m2167                                                         NORM $
14m2167                                                 ];[
14m2167 14176 540731170000060554161623254204160000             ILGIOT $                ;BLKI
14m2167 14177 010731170000060554161623654314160000             NORM $
14 2167                                                 ].REPEAT WAITS
14 2168
14 2169                                                 ; TDATAI  --  765    READ TIMER REGISTER
14 2170                                                 TDTAIA:
14 2171                                                 .REPEAT WAITSFIX [;Is there some reason this would lose?
14 2172                                                         START-IN D[CONST TIM.DEV] DEST[DEV-ADDRESS] NORM PUSHJ[TDTI] $
14 2173                                                                 ;get bits.
14m2174                                                 ];[;Is there some reason this would lose?
14m2174 14200 010731170030000001401246225571457000             START-IN D[CONST TIM.DEV] DEST[DEV-ADDRESS] NORM PUSHJ[TDTI] $
14m2174                                                                 ;get bits.
14 2174                                                 ].REPEAT WAITSFIX
14 2175                                                 .REPEAT 1 - WAITSFIX [
14 2176                                                         START-IN NORM PUSHJ[TDTI] $ ;get bits.
14 2177                                                 ];.REPEAT 1 - WAITSFIX
14 2178 14201 010731310000226654161561254304560000             D[AR] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<20] LBJUMP[SEOI] ]$
14 2178                                                  ;Store them and return.
14 2167                                                 ].repeat waitsfix (ordering)
14 2168                                                 ; TCONO -- 760
14 2169 14202 010731170040060554012463655774170000     TCNOA:  D[CONST TIM.DEV] DEST[DEV-ADR] SHORT $
14 2170 14203 010730170000000001416162026071456000             D[D%AMEM1] DEST[Q] NORM JUMP[TCNO1]$
14 2171                                                 ; TCONI -- 761
14 2172 14204 010731170040000001401246225571457000     TCNIA:  D[CONST TIM.DEV] DEST[DEV-ADR] NORM PUSHJ[TCNI1] $ ;get bits.
14 2173 14205 010731310000226654161561254304560000             D[AR] SMAC  [ IFRQ DEST[MEMSTO AR] COND[MA<20] LBJUMP[SEOI] ]$
14 2173                                                  ;store them and return.
14 2174                                                 .repeat waitsfix [      ;This is just an ordering question.  Makes
14 2174                                                  dispatch work.
14 2175                                                 ; TCONSZ -- 763
14 2176                                                 TCNSZA: D[CONST TIM.DEV] DEST[DEV-ADR] NORM PUSHJ[TCNI1] $ ;get bits.
14 2177                                                         D[MASK 22] ALU[D&Q] C550 COND[OBUS=0] LBJUMP[DSKP1] $ ;Skip if
14 2177                                                  off.
14m2178                                                 ];[     ;This is just an ordering question.  Makes dispatch work.
14m2178                                                 ; TCONSZ -- 763
14m2178 14206 010731170040000001401246225571457000     TCNSZA: D[CONST TIM.DEV] DEST[DEV-ADR] NORM PUSHJ[TCNI1] $ ;get bits.
```

```
14 2181 14Z11 010641000000000001404562124721456000             D[MASK ZZ] ALU[D&Q] CS50 COND[-OBUS=0] LBJUMP[DSKP1] $ ;Skip if
14 2181                                                          on.
14 2182                                                        .repeat 1 - waitsfix [  ;This is just an ordering question.  FLUSH THIS
14 2182                                                          COPY
14 2183                                                        ; TCONSZ -- 763
14 2184                                                        TCNSZA: D[CONST TIM.DEV] DEST[DEV-ADR] NORM PUSHJ[TCNI1] $ ;get bits.
14 2185                                                                D[MASK ZZ] ALU[D&Q] CS50 COND[OBUS=0] LBJUMP[DSKP1] $ ;Skip if
14 2185                                                          off.
14 2186                                                        ];.repeat 1 - waitsfix
14 2187
14 2188                                                        .repeat 1 - waitsfix [;*** This conditional is sole for ordering.  FLUSH
14 2188                                                          THIS COPY
14 2189
14 2190                                                        ; TDATAO -- 764  LOAD THE 36-BIT TIMER
14 2191                                                        TDTAOA: D[CONST TIM.DEV] DEST[DEV-ADR] NORM $
14 2192                                                                DFRQ ALU[MEMAC] DEST[HOLD]  JUMP[TDATO] $
14 2193
14 2194                                                        ; TDATAI  --  765    READ TIMER REGISTER
14 2195                                                        TDTAIA:
14 2196                                                                START-IN NORM PUSHJ[TDTI] $ ;get bits.
14 2197                                                                D[AR] SMAC $ ;Store them and return.
14 2198                                                        ];.repeat 1 - waitsfix (*** ordering)
14 2199
14 2200
14 2201                                                        .REPEAT WAITS [
14 2202                                                          .reloc
14m2203                                                         ];[
14m2203                                                          .reloc
14m2203                                                         [.USE[HILOC]
14m2203                                                         [ xlist
14 2203                                                          list ]                                  ]]REPEAT WAITS
14 2204
14m2205                                                          .pair
14m2205                                                         [:. \ Z + .
14 2206 14Z12 540731170000060554161623254204165000              ]DSKP1:      NEOI $
14 2207 14Z13 540731172000073554161623254204165000                   SKPDSP[ALWAYS] $
14 2208
14 2209                                                         .REPEAT TIMER [
14 2210                                                          .pair
14 2211
14 2212                                                        TCNO1:  D[MASK 32.] ROT[4] ALU[D&Q] DEST[Q] NORM $
14 2213                                                                D[IR] MASK[3] ALU[DORQ] DEST[D%AMEM1] JUMP[MAIN] $
14 2214                                                                   ;Put new PI CHAN. in bits 33-35 of A-MEM[1].
14 2215
14 2216                                                        TCNI1:  D[D%AMEM1] MASK[4] DEST[Q AR] NORM $ ;get intrpt flag and pi chn
14 2216
14 2217                                                                D[IR] ALU[D&Q] DEST[Q] NORM POPJ $ ;this is for conso, z
14 2218
14 2219                                                        TDATO:  START-OUT D[MEM] MASK[12.] DEST[IOD] NORM $
14 2220                                                                   ;Get low 12 bits of new timer value...
14 2221                                                                   ;Place in hardware counter (the "TP TIMER").
14 2222                                                                MAPF[TIM.DO] START-OUT D[CONST 1] DEST[IOD] CS00 $
14 2223                                                                   ;Enable interrupts from timer.
14 2224                                                                MAPF[TIM.ENB] D[MASK 24.] ROT[12.] DEST[Q] CB00 $
14 2225                                                                D[MEM] ALU[D&Q] DEST[D%AMEM3] NORM JUMP[MAIN] $
14 2226                                                                   ;Put other 24 bits into A-MEM[3] (the "TIMER REGISTER").
14 2227
14 2228                                                        TDTI:   MAPF[TIM.DI] START-IN D[IOD] ROT[12.] MASK[12.] DEST[AR] CS00 $
14 2229                                                                   ;Read hardware timer.
14 2230                                                                MAPF[TIM.DI] D[IOD] ROT[12.] MASK[12.] DEST[Q] CS00 $
14 2231                                                                   ;Read it again.
14 2232                                                                D[AR] ALU[Q-D] DEST[AR] NORM $
14 2233                                                                START-IN D[AR] ROT[36. - 1] MASK[35.] CS00 COND[-ZERO] JUMP[TDTI]
14 2233                                                         $
14 2234                                                                   ;If it changed by more than one, try again (to ensure that we
14 2234                                                         don't
14 2235                                                                   ;get a garbaged value because of reading it while it is
14 2235                                                         changing).
14 2236                                                                D[D%AMEM3] ALU[DORQ] DEST[AR] NORM POPJ $
14 2237                                                                   ;Combine high-order bits from A-MEM[3] with
14 2238                                                                   ; low-order bits from hardware counter.
14 2239
14 2240
14 2241                                                        TIMRST: ALU[0] DEST[D%AMEM3] NORM $
14 2242                                                                D[LIT 377777000000] DEST[D%AMEM4] $
14 2243                                                                D[LIT TIMINT] DEST[D%AMEM0] POPJ $
14 2244
14m2245                                                          ] [
14m2245                                                          .pair
14m2245                                                         [:. \ Z + .
14m2246                                                          ]
14m2246 14Z14 010640170000060S41101623647314165000             TCNO1:  D[MASK 32.] ROT[4] ALU[D&Q] DEST[Q] NORM $
14m2246 14Z15 010631170000Z27114007020257714565000                     D[IR] MASK[3] ALU[DORQ] DEST[D%AMEM1] JUMP[MAIN] $
14m2246                                                                   ;Put new PI CHAN. in bits 33-35 of A-MEM[1].
14m2246
14m2246 14Z16 010730370000060S5401162366071416000             TCNI1:  D[D%AMEM1] MASK[4] DEST[Q AR] NORM $ ;get intrpt flag and pi chn
14m2246
14m2246 14Z17 010640170000060S5416162425771416000                     D[IR] ALU[D&Q] DEST[Q] NORM POPJ $ ;this is for conso, z
14m2246
14m2246 14Z20 010731170031060S5403152365471516000             TDATO:  START-OUT D[MEM] MASK[12.] DEST[IOD] NORM $
14m2246                                                                   ;Get low 12 bits of new timer value...
14m2246                                                                   ;Place in hardware counter (the "TP TIMER").
14m2246 14Z21 010731171131060S5400352365551416000                      MAPF[TIM.DO] START-OUT D[CONST 1] DEST[IOD] CS00 $
14m2246                                                                   ;Enable interrupts from timer.
14m2246 14Z22 010730171000060S4306162364711416000                      MAPF[TIM.ENB] D[MASK 24.] ROT[12.] DEST[Q] CB00 $
14m2246 14Z23 010641170000Z27114161060254715560000                     D[MEM] ALU[D&Q] DEST[D%AMEM3] NORM JUMP[MAIN] $
14m2246                                                                   ;Put other 24 bits into A-MEM[3] (the "TIMER REGISTER").
14m2246
14m2246 14Z24 010731370130060S4303162365711416000             TDTI:   MAPF[TIM.DI] START-IN D[IOD] ROT[12.] MASK[12.] DEST[AR] CS00 $
14m2246                                                                   ;Read hardware timer.
14m2246 14Z25 010730170100060S4303162365711416000                      MAPF[TIM.DI] D[IOD] ROT[12.] MASK[12.] DEST[Q] CS00 $
14m2246                                                                   ;Read it again.
14m2246 14Z26 011611370000060S5416162365431416000                      D[AR] ALU[Q-D] DEST[AR] NORM $
14m2246 14Z27 010731000030304510707620254114565000                     START-IN D[AR] ROT[36. - 1] MASK[35.] CS00 COND[-ZERO] JUMP[TDTI]
14m2246                                                         $
14m2246                                                                   ;If it changed by more than one, try again (to ensure that we
14m2246                                                         don't
14m2246                                                                   ;get a garbaged value because of reading it while it is
14m2246                                                         changing).
14m2246 14Z30 010631370000060S5416162426171416000                      D[D%AMEM3] ALU[DORQ] DEST[AR] NORM POPJ $
```

```
SLOE  March 23, 1984 21:19:45  file STRING:  --  of -- F41NNF

14mZ246 14232 010731170777776000001103655531416000              D[LIT 377777000000] DEST[D%AMEM4] $
14mZ246 14233 010731170000000003033500425531416000              D[LIT TIMINT] DEST[D%AMEM0] POPJ $
14mZ246
14 2245                                                        ];end REPEAT TIMER
14 2246
14 2247
14 2248                                                       .repeat 1 - TIMER [
14 2249
14 2250                                                        TCNO1:
14 2251                                                        TDATO: JUMP MAIN $
14 2252
14 2253                                                      TCNI1:
14 2254                                                      TDTI:   ALU[0] DEST[AR Q] NORM POPJ $
14 2255
14 2256                                                        ]  ;end repeat 1 - TIMER
14 2257
14 2258
```

```
SLOE  March 23, 1984 21:19:46  file DSK:F418S.SLO  --  of -- F41NNF

01 0031
01 0032                                                 .repeat JSYSDF [ .INSERT F4JSYS ]       ;MENTR&MRETN UCODE EMULATION
01 0033
01 0034                                                 .REPEAT 1 - WAITS [
01 0035                                                 .repeat DBOOTF [ .INSERT F4DBOT ]       ;DISK BOOTER
01 0036                                                 ];.REPEAT 1 - WAITS
01 0037                                                 .REPEAT WAITS [
01 0038                                                 DBOOTA: JUMP[17002] $           ;Use macro boot in PROM
01m0039                                                 ];[
01m0039 14234 010731170000360054161620254314560000     DBOOTA: JUMP[17002] $           ;Use macro boot in PROM
01 0039                                                 ].REPEAT WAITS
01 0040
01m0041                                                   .org[17777]    [ xlist
01 0041                                                    list ]               ;THIS IS TERMINATION FLAG FOR UCODE BOOTER
01 0042
01 0043 17777 010731170000Z24234161620254114560000           JUMP[HLTLOC] LONG $     ;XFER INSTR FOR UCODE BOOTER
01 0044                                                                           ;TURN OFF HISTORIES AND HALT
01 0045
01 0046
01 0047                                                 list-labels

labels
 31 2553  5356 &CA1          31 2554  5357 &CA2          31 2557  5362 &CAB1
 31 2555  5360 &CAI1         31 2556  5361 &CAM1         31 2588  5363 &CMM1          0   38     0 .
  6  447  4226 ADJBP          5  402  6212 ADJSP          5  404  4166 ADJSP1         5  404  4175 ADJSP2
  5  404  4201 ADJSPO         1  184 10070 APDINT         1  232 10112 APIOT          1  139 10063 APRC1
  1  141 10064 APRCZ          1  147 10066 APRCHK         1  187 10071 APRCI          1  188 10071 APRCII
  1  229 10110 APRCIN         1  229 10111 APRCIY         1  148 10067 APRCK1         1  101 10040 APRCO
  1  121 10053 APRCOZ         1  125 10055 APRCO3         1   28 10000 APRDSP         1  235 10117 APRRST
  2  194 11235 AROV           1   32 12735 AS-DISPATCH     1   68 12755 asblko         1   73 12756 ASDTO
  3  373 11340 ASEOI         13 1031  4744 ASH1           14 1136  5037 ASHC1          14 1155  5057 ASHC2
 14 1151  5053 ASHC4         14 1146  5050 ASHC6          14 1157  5063 ASHC7          14 1144  5046 ASHC8
 14 1149  5052 ASHC9         13 1049  4757 ASHDO          13 1054  4763 ASHDO1         13 1057  4765 ASHDOP
 13 1036  4750 ASHNEG         1   22 12732 ASIOT           1  102 12770 ASMAP          1  112 12774 ASMAP1
  1  150 13007 ASMAP3         1  155 13010 ASMAP4          1  129 13002 ASMAPS         1  158 13011 ASRST
 11  546 10326 BBNCND        13 1068  4777 BIGAS2         13 1039  4752 &IGASH         8  620  4316 BII
  8  627  4321 BIIH1          8  634  4325 BIIHZ          8  625  4320 BIIHA         17 1434  6522 BLT0
 17 1436  5120 BLT0A         17 1436  5122 BLT1           17 1436  5126 BLT1B         17 1436  5130 BLT1C
 17 1436  5147 BLT1CZ        17 1436  5145 BLT1CK         17 1436  5134 BLT1D         17 1436  5153 BLT1IZ
 17 1436  5151 BLT1IN        17 1436  5142 BLT1LP         17 1436  5155 BLT1Z         18 1570  5241 BLT99
 18 1440  5164 BLTA1         18 1537  5226 BLTA10         18 1494  5210 BLTA2         18 1530  5217 BLTA2A
 18 1530  5222 BLTA2B        18 1533  5225 BLTA4          18 1490  5206 BLTAL2        18 1459  5175 BLTAL3
 18 1490  5201 BLTFXZ        18 1530  5223 BLTL1          18 1538  5227 BLTL2         18 1570  5234 BLTLZA
 18 1491  5207 BLTXIT        17 1436  5161 BLTZLP         3  376 11341 CASEOI         3  773 12146 CHKINS
  3  773 12151 CKINS1         3  773 12156 CKINS2         3  773 12160 CKINS3        11 1740 14131 CKIOTU
  8 1479 13572 CLKCLR         8 1481 13573 CLKENB         8 1461 13565 CLKINT         8 1472 13570 CLKRDFLG
  8 1478 13572 CLKRST         7 1225 13516 CLRMEM         3  416 11356 CPOPJ        28 2307  5354 CSSKP1
  8 1529 13631 CTYCI          8 1533 13635 CTYCI1         8 1532 13634 CTYCIZ         8 1542 13645 CTYCI3
  8 1540 13643 CTYCI8         8 1537 13641 CTYCI9         8 1526 13627 CTYCO          8 1543 13646 CTYCO1
  8 1549 13652 CTYCO3         8 1552 13655 CTYCO4         8 1559 13662 CTYCO5         8 1579 13700 CTYCS
  8 1577 13676 CTYCZ          8 1506 13607 CTYDI          8 1513 13616 CTYDO          8 1444 13521 CTYDSP
  8 1489 13576 CTYIN1         8 1449 13561 CTYINT         8 1497 13602 CTYIOT         8 1569 13672 CTYRS1
  8 1563 13664 CTYRST         1   39 14234 DBOOTA         1   14 12165 DCA-DISPATCH     1   30 12177 DCACNI
  1   59 12175 DCACNO         1   64 12205 DCACO1         1   93 12222 DCACO2         1  171 12263 DCACO3
  1   98 12224 DCACO5         1  121 12235 DCACO6         1  142 12246 DCACO7         1  165 12260 DCACOJ
  1   36 12203 DCACSO         1   33 12201 DCACS2         1  274 12310 DCADI1         1  291 12317 DCADIA
  1  311 12320 DCADO1         1  269 12167 DCADTI         1  300 12173 DCADTO         1  372 12340 DCAINT
  1    4 12162 DCAIOT         1  391 12345 DCARST         1  235 12304 DCASGZ         1  347 12326 DCASLR
  1  182 12267 DCASRG         4  611 13153 DCG1           4  611 13150 DCGO           4  611 13170 DCGWR
  4  436 13125 DCINIT        11 1729 14127 dcni1         11 1709 14112 DCNIA         11 1718 14120 dcno1
 11 1725 14124 DCNOZ         11 1726 14125 DCNO3         11 1700 14110 DCNOA         11 1712 14114 DCNSOA
 11 1715 14116 DCNSZA        11 1749 14137 ddis          31 1954 11206 DECOD1        31 1953 11205 DECODE
 13 2030 14151 DEVGCL        13 2033 14152 DEVGST        13 2038 14153 DEVGST1       13 2041 14154 DEVGSTZ
  1   12  6262 DFN           2  153 11556 DFN0           2  155 11560 DFN1           2    2  7764 DIAGI
  2    3 12560 DIAGI2         2    3 12563 DIAGI3         2    4  7766 DIAGO          2    5 12564 DIAGOZ
  2    5 12567 DIAGO3         2    6 12610 DIAGSB        12  914  4560 DIV9          12  922  4572 DIVBA
 12  910  4565 DIVMA          1   90  4055 DMOVE1         1   91  4056 DMOVE2         1   94  4060 DMOVN1
  1   95  4061 DMOVN2         1   98  4062 DMOVN3         1  101  4064 DMOVN4         1  103  4066 DMOVN5
  1  119  4067 DMVEM1         1  120  4070 DMVEM2         1  123  4072 DMVNM1         1  126  4074 DMVNM3
 12  926  4577 DODDIV         12  946  4604 DODIV          12  950  4606 DODIV1        12  958  4612 DODIV2
 12  956  4611 DODIV3         12  992  4626 DODIV4         12 1001  4741 DODIV5        12  974  4617 DODIV7
 11  783  4370 DOIMUL         11  797  4375 DOMMUL         11  809  4401 DOMUL         11  814  4404 DOMUL0
 11  822  4516 DOMUL1         11  826  4520 DOMUL4         11  811  4402 DOMULF         7  568  4276 DPB1
 11 1658 14054 DSKIN0        11 1668 14056 DSKIN1        11 1658 14060 DSKINZ        11 1668 14062 DSKIN3
 11 1743 14133 DSKINT        11 1669 14064 DSKO4         11 1670 14070 DSKO5         11 1671 14074 DSKO6
 11 1672 14100 DSKO7         14 2206 14212 dskp1         11 1668 14103 DSKRST        11 1757 14146 DSKWDN
 11 1751 14140 DSKWT1        11 1752 14141 DSKWT3        11 1755 14144 DSKWT4        30 1905 11165 ECC-PROCEED
 30 1909 11166 ECC-RET       29 1851 11142 ECCDIE        30 1896 11161 ECCFX         30 1913 11170 ECCFX1
 29 1714 11060 ECCFX2        29 1811 11121 ECCHRD        29 1858 11143 ECCNXM        31 1923 11171 ECCPMA
 29 1707 11055 ECCTP1        29 1701 11052 ECCTRP        29 1802 11117 ECCUD1        29 1705 11105 ECCUDF
 29 1715 11060 ECCUTP        29 1778 11101 ECCXIT         4  463 10262 END-OF-PI-CODE  15 1197  5114 EXCH1
 15 1198  5115 EXCHA         16  866 10456 EXECL1        16  878 10464 EXECL2        16  860 10453 EXECLR
  2  371 11737 EXPUF          2  390 11742 FADSU1         2  389 11741 FADSUB         2  173 11600 FAOS1
  2  177 11602 FAOS2          2  179 11604 FAOS3          2  183 11610 FAOS4          2  174 11600 FAOS5
  2  535 12051 FD             2  206 11625 FD1            2  226 11642 FD2A           2  263 11666 FD2C
  2  261 11665 FD2D           2  266 11670 FD3            2  271 11674 FD4            2  586 12075 FDAD
 12  987  4624 FDODIV         2  521 12043 FDS            2  522 12044 FDS1           2  564 12064 FDSHFT
  2  205 11624 FDV1           2  213 11630 FDV2           2  265 11667 FDV3           2  270 11673 FDV4
  3  365 11334 FIX-SEOI       3  678 12126 FIXER          3  698 12134 FIXER1         3  701 12135 FIXERZ
  3  706 12137 FIXER3         3  624  6254 FIXR           3  629 12105 FIXR0          3  630 12106 FIXR1
  3  662 12121 FIXRZ          3  639 12112 FIXRN          3  710  6256 FLTR           3  716 12140 FLTR0
  3  717 12141 FLTR1          3  718 12142 FLTR2          2  480 12021 FM             2  487 12024 FM1
  2  494 12027 FM2            4  436 13107 FMNBW1         4  436 13106 FMNBWT         2  186 11612 FMP1
  2  190 11614 FMP2           2  192 11616 FMP3           2  196 11622 FMP4           2  187 11612 FMP5
  6 1313 13406 FMTDON         4  436 13115 FMTHNG         2  191 13040 FNTDSP         2  210 13065 FNTINT
  2  175 13031 FNTRD          2  188 13037 FNTRD1         2  198 13060 FNTRST         2  172 13030 FNTWR1
  2  163 13023 FNTWRA         2  162 13022 FNTWRB         2  149 13015 FNTWRT        17  915 10502 FORCE1
 17  898 10472 FORCEM         2  464 12012 FPLOW           2  423 11774 FPNAR          2  276 11677 FPNDIV
  2  392 11744 FS1            2  398 11747 FS2            2  403 11754 FS3            2  412 11767 FS4
  2  415 11771 FS5            1   16  6264 FSC            2  168 11574 FSCZ           2  162 11566 FSCZAP
  1   44 12675 GRN-DISPATCH    1   93 12720 GRNDTO         1   96 12721 GRNINT         1   34 12672 GRNIOT
  1  111 12727 GRNRST          1   85 12715 GRNSTS         2   99  4043 HALTED        32 2774  5376 HLL1
 32 2775  5377 HLL2          32 2899  5434 HLLE1         32 2902  5436 HLLE2         32 2904  5440 HLLE3
 32 2906  5442 HLLEM1        32 2908  5444 HLLEM2        32 2900  5435 HLLES1        32 2910  5446 HLLES2
 32 2912  5450 HLLES3        32 2776  5400 HLLM1         32 2777  5401 HLLM2         32 2859  5423 HLLO1
 32 2860  5424 HLLOS1        32 2822  5416 HLLZS1        32 2987  5502 HLR1          32 2988  5503 HLR2
 32 3121  5550 HLRE1         32 3128  5556 HLRE2         32 3122  5551 HLRE4         32 3123  5552 HLREM1
 32 3130  5560 HLREM2        32 3132  5562 HLREM3        32 3124  5553 HLREM4        32 3125  5554 HLRES1
 32 3134  5564 HLRES3        32 3126  5555 HLRES4        32 2989  5504 HLRM1         32 2990  5505 HLRM2
 32 3067  5523 HLRO1         32 3068  5524 HLRO2         32 3069  5525 HLROM1        32 3070  5526 HLROM2
 32 3071  5527 HLROS1        32 3072  5530 HLROS2        32 3082  5537 HLRS1         32 3083  5540 HLRS2
```

```
 6 1196 13345 INT-READ          4  611 13136 INT-STARTDC       6 1249 13363 INT-WRITE         6 1342 13421 INT-WRTDONE
 2  141 11213 IOINT             2  127  4047 ITRAP            2  140  4103 ITRAP1           4  611 13175 ITRCHK
19 1651  5264 JFCL1            19 1657  5271 JFCLZ           14 1115  5026 JFFO1           14 1117  5030 JFFOZ
14 1120  5033 JFFO3            14 1123  5036 JFFO4           14 1121  5034 JFFO5            2  150 11215 JMPAC
 1   73  4052 JMPFAIL          26 2211  5346 JRA1            26 2213  5350 JRAZ            19 1612  5242 JRST0
19 1616  5244 JRST1            19 1626  5252 JRSTZ           19 1637  5257 JRST21          19 1633  5256 JRSTZ2
19 1640  5260 JRST3            19 1620  5246 JRST4           19 1621  5247 JRST5           19 1643  5262 JRST9
26 2200  5343 JSA1             26 2190  5342 JSP1            26 2181  5337 JSR1             4  236  4132 JSYS1
 4  306  4157 JSYSZ             4  307  4161 JSYSZA           4  307  4164 JSYSZB           4  254  4137 JSYSA
 4  306  4151 JSYSAZ            4  306  4156 JSYSA3           4  306  4153 JSYSFX           3  623 12100 KAFIX1
 3  623 12103 KAFIXN           3  623 12101 KAFIXP           3  653  6244 KIFIX            3  658 12117 KIFIX0
 3  659 12120 KIFIX1           3  668 12123 KIFIXN           4  436 13073 KNYCLR           4  436 13101 KNYGOA
 4  629 13241 KNYMTP           4  436 13075 KNYRGO           4  436 13073 KNYRS1           7 1225 13501 KNYW1
 7 1225 13477 KNYWAIT          7  508  4250 LDB1            25 1524 10761 LOADA0          25 1530 10764 LOADA1
25 1532 10765 LOADAZ           25 1546 10773 LOADA3           25 1516 10755 LOADAR           1    2 13541 LPT-DISPATCH
 1    3 13702 LPTCOZ            1    3 13706 LPTCO3            1   38 13721 LPTDOZ           8 1444 13541 LPTDSP
 1  159 13773 LPTHT             1  161 13774 LPTHT1            1  127 13762 LPTIEN           1  280 14037 LPTINT
 1  292 14044 LPTNTZ            1   73 13734 LPTP7A            1   97 13744 LPTP7B            1  107 13751 LPTP7C
 1  109 13752 LPTP7D            1  113 13754 LPTP7E            1  135 13765 LPTP7F            1  154 13771 LPTP7I
 1  111 13753 LPTP7S            1  182 14004 LPTPAP           1   68 13732 LPTPK7            1  301 14046 LPTRST
 1  186 14005 LPTSL1            1  191 14006 LPTSLZ           1  200 14012 LPTSL3            1   99 13746 LPTSLE
 1   17 13713 LPTSTS           13 1101  5020 LSH0            13 1093  5012 LSH1            14 1181  5100 LSHC1
14 1187  5107 LSHCZ           14 1183  5102 LSHCA1           13 1097  5015 LSHDO            13 1103  5021 LSHNEG
13 1096  5014 LSHX              2  131  4077 LUUO             3  386 11343 MACSTO           21 1204 10635 MACTRZ
21 1145 10634 MACTRP            3  389 11344 MAIN            18  954 10517 MAP-RET          12  562 10336 MAPACB
12  575 10343 MAPAGE           22 1225 10644 MAPBKI           11  515 10315 MAPBKO           14  802 10427 MAPC17
11  526 10320 MAPCNI           11  532 10321 MAPCNO           12  587 10347 MAPCO0           13  694 10402 MAPCO1
13  724 10404 MAPCO3           14  765 10411 MAPCO4           14  766 10411 MAPCO5           14  773 10413 MAPCO6
14  795 10424 MAPCO7           12  625 10361 MAPCO8           12  627 10362 MAPCOC           10  458 10277 MAPCSO
10  490 10305 MAPCSZ           10  467 10301 MAPD10           10  498 10307 MAPDI1           10  497 10306 MAPDTI
10  466 10300 MAPDTO            2    5 12570 MAPFTB           23 1346 10654 MAPIF             9  432 10262 MAPIOT
18  985 10536 MAPLNG           24 1381 10670 MAPNTR           14  756 10406 MAPOFF           23 1351 10656 MAPRD
23 1359 10662 MAPRDZ           23 1357 10661 MAPRDA           23 1363 10664 MAPRMW           28 1670 11037 MAPRST
18  966 10525 MAPSHT           18  931 10506 MAPTRP           20 1111 10623 MAPUF1           20 1085 10606 MAPUL1
20 1071 10577 MAPULG           19 1045 10565 MAPUS1           19 1036 10560 MAPUSH           19 1059 10574 MAPUTI
19 1026 10556 MAPUTP           20 1122 10627 MAPUTW1          20 1132 10632 MAPUTW2          23 1366 10665 MAPWR
18  958 10521 MAPXFX            7 1225 13440 MBOOT            7 1225 13446 MBOOT1            7 1225 13452 MBOOTR
 7 1225 13466 MBTBY1            7 1225 13474 MBTBY2            7 1225 13464 MBTBYTE           7 1225 13476 MBTBYX
 7 1225 13505 MBTCHECK          7 1225 13511 MBTDNX            7 1225 13515 MBTERR            3  419 11357 MEMADR-18BIT
 3  422 11360 MEMADR-23BIT-ABS  3  361 11332 MEMST             3  374 11340 MEMSTA             3  377 11341 MEMSTCA
24 1406 10701 MFA1             24 1412 10704 MFA2            26 1556 10776 MFA3            26 1568 11003 MFA4
25 1566 10753 MFA5             26 1564 11002 MFA6            25 1558 10777 MFA7            24 1421 10710 MFB3
27 1597 11014 MFB4             27 1601 11016 MFB41           27 1596 11014 MFB5            24 1404 10700 MFHIEX
27 1584 11007 MFTR1            27 1592 11012 MFTR10          27 1613 11021 MFTR2           27 1620 11022 MFTR3
27 1626 11024 MFTR4            27 1656 11035 MFTR41          27 1627 11024 MFTR5           27 1632 11025 MFTR6
27 1638 11030 MFTR61           27 1640 11031 MFTR62          27 1647 11033 MFTR7           27 1650 11034 MFTR9
25 1468 10731 MFTY00           25 1472 10733 MFTY01          25 1466 10730 MFTYP0          25 1454 10725 MFTYP1
24 1397 10676 MFUS              3  342 11326 MLERR             3  332 11321 MLWAIT            3  334 11322 MLWRT
 9  713  4340 MOVM1             9  715  4342 MOVM11           9  717  4344 MOVM2            9  719  4346 MOVMS1
 9  721  4350 MOVMS2            9  675  4333 MOVS1            9  676  4334 MOVSM1           9  677  4335 MOVSM2
 9  678  4336 MOVSS1            9  442 10267 MPIOTD           11  795  4373 MSETOV          11  795  4374 MSTOV1
 3  347 11330 MSTRT1           27 1619 11022 MTRPAL           2  183  4124 MUJSR             2  177  4121 MUJSYS
10  770  4361 MUL1             10  773  4363 MUL2            10  776  4365 MUL3             2  144  4104 MUUO
 2  148  4105 MUUO1             0   38     0=NAF             4  436 13120 NCNTW1            4  436 13120 NCNTWT
 2  304 11711 NEGNOR            2  310 11714 NN1             2  327 11717 NORMOV            2  450 12006 NORND
 2  343 11724 NOVND             2  336 11722 NOVYES          2  288 11702 NRMLI2           31 2724  5371 ORCM1
31 2727  5374 ORCMB1           31 2728  5375 ORCMB2          31 2725  5372 ORCMM1          31 2726  5373 ORCMM2
 7 1225 13456 PZ                7 1225 13457 PZA             1    2 12620 PAN-DISPATCH      1   50 12652 PAN-INTR
 1    2 12640 PAN-IOB-READ      1    8 12644 PAN-IOB-WRITE     1   75 12662 PANIDS           1    2 12612 PANIOT
 1   87 12665 PANIST            1   86 12657 PANNC1           1   97 12670 PANRST           2  180 11231 PC<20
 2  191 11234 PC<20I            2  168 11223 PCTRAP           2  178 11230 PCTRPX          25 2164  5333 PDL-TRAP
25 2158  5331 PDLO              2  160  4112 PGRTRP           2  274 10131 PI-CHECK-RQS      3  371 10203 PI-DISMISS
 2  258 10123 PI-GET-CHN        4  456 10256 PI-RESET         4  446 10251 PICONISUB         4  397 10215 PICONO
 3  388 10210 PIDSM1            3  388 10211 PIDSM2           1   67 10020 PIDSP             4  444 10250 PIERR
 3  331 10161 PIGEN             3  346 10167 PIGEN1           3  355 10174 PIGENWT           2  267 10127 PIGETMASK
 2  314 10152 PIINTGO           2  263 10125 PIL1             2  316 10153 PIL11             4  418 10233 PIL3
 4  420 10235 PIL4              4  422 10237 PIL5             4  424 10241 PIL6              4  404 10222 PIL7
 4  407 10224 PIL8              4  415 10230 PIL9             2  157  4111 PIMUUO           25 2092  5302 POP1
25 2103  5310 POP2             25 2098  5305 POP3            25 2131  5317 POPJ0           25 2127  5316 POPJ1
25 2145  5324 POPJ2            25 2143  5323 POPJ3           25 2151  5327 POPJEC          25 2149  5326 POPJOV
25 2134  5320 POPJTR            2  298 11706 POSNOR          25 2073  5274 PUSH1           25 2078  5276 PUSH2
25 2081  5300 PUSH3            25 2114  5313 PUSHJ1           1   41  4053 RDAMEM           3  234 11244 RESET
 3  250 11254 RESET1           13 1077  5002 ROT1            14 1165  5064 ROTC1           14 1171  5073 ROTC2
14 1167  5066 ROTCA1           13 1084  5007 ROTDO           13 1083  5006 ROTNEG           3  268 11262 RSTCCL
 3  360 11332 SEDI              3  399 11346 SET-CONTEXT     33 3406  5615 SET-HALF          2  365 11734 SETFOV
 3  413 11354 SETMD1            3  402 11347 SETMODE         13 1042  4755 SETOV1           2  367 11735 SETOVX
 3  370 11336 SNEDI            31 2647  5364 STCM            31 2650  5367 STCMB           31 2651  5370 STCMBA
31 2648  5365 STCMM            31 2649  5366 STCMMA           4  611 13143 STDC1             4  611 13141 STRTDC
 3  384 11342 SXEDI             6 1426 13432 TAPCLR           6 1100 13266 TAPE-DISPATCH     6 1081 13247 TAPEBR
 6 1081 13246 TAPEFR            4  611 13217 TAPEMT           6 1162 13335 TAPENR            4  436 13070 TAPEOI
 6 1333 13415 TAPEPI            6 1142 13330 TAPERD           6 1083 13262 TAPERS            6 1117 13306 TAPERS0
 6 1130 13313 TAPERS1           4  629 13245 TAPERW           6 1171 13336 TAPERX            6 1214 13347 TAPEWR
 6 1298 13401 TAPEXFRDON        6 1270 13367 TAPINT           6 1306 13404 TAPINTENB         6 1290 13376 TAPINTWRT
 4  436 13071 TAPRST           14 2246 14216 TCNI1           14 2172 14204 TCNIA           14 2246 14214 TCNO1
14 2169 14202 TCNOA            14 2180 14210 TCNSOA          14 2178 14206 TCNSZA          14 2246 14220 TDATO
33 3326  5602 TDCX1            33 3328  5603 TDCX2           14 2246 14224 TDTI            33 3399  5612 TDOX2
14 2170 14200 TDTAIA           14 2167 14174 TDTAOA          33 3397  5611 TDOX1            33 3256  5573 TDZX1
33 3258  5574 TDZX2             4  639 13237 TERAS1           4  642 13240 TERASE            2  514 12042 TESMUL
14 2146 14172 TIMER-DISPATCH   14 2095 14156 TIMINT          14 2146 14167 TIMIOT          14 2246 14231 TIMRST
33 3294  5600 TLC1             33 3366  5607 TLO1            33 3225  5571 TLZ1             6 1190 13344 TNRP3
 6 1365 13300 TPCONI            6 1473 13434 TPCONIBTS        6 1417 13426 TPCONO2          6 1389 13304 TPCONSO
 6 1389 13302 TPCONSZ           4  611 13214 TPMAOK           6 1081 13252 TPSETDMP         6 1081 13250 TPSETIND
23 1336 10650 TRAP-ENTER        1   69  4051 TRAP1X          33 3292  5577 TRC1             4  611 13202 TRCHECK
 4  611 13203 TRCHKB            4  611 13206 TRDONE          4  611 13215 TREOF             4  611 13216 TRERR
33 3364  5606 TRO1             4  611 13177 TRPZ            33 3223  5570 TRZ1            33 3324  5601 TSC1
33 3330  5604 TSCE1            33 3332  5605 TSCN1           33 3190  5566 TSNE1           33 3191  5567 TSNN1
33 3395  5610 TSO1             33 3401  5613 TSOE1           33 3403  5614 TSON1           33 3254  5572 TSZ1
33 3260  5575 TSZE1            33 3262  5576 TSZN1            6 1257 13365 TWERR            4  636 13236 TWREOF
12 1707 14150 TYMRST            1    8  6260 UFA             2  148 11552 UFA1            2  454 12007 UFANOR
 3  216  4126 UMOVX            15  831 10441 USRCL1          1  388 12517 VCBRST           1  171 12440 VCCMD
22 1327 10645 USRRST            2  186  4125 UUOPJ           1  360 12503 VCGO             1  323 12472 VCGST
 1  425 12534 VCDONE             1    1 12353 VCDSP           1  147 12431 VCINT3           1  153 12434 VCIOT
 1  121 12423 VCINT              1  145 12430 VCINTZ          1  482 12554 VCODLY           1  166 12437 VCODSP
 1  390 12520 VCLP               1  442 12541 VCLST           1  203 12445 VCOND0           1  255 12462 VCONO1
 1  478 12551 VCOLP              1  201 12444 VCONO           1  311 12472 VCONSO           1  300 12472 VCONSZ
 1  257 12463 VCONO2             1  266 12464 VCONO4          1  336 12475 VCOUT            1  479 12552 VCOUT1
 1    3 12413 VCORG              1  327 12475 VCOUNT          1  287 12472 VCSTAT           1  450 12543 VCWRD
 1  184 12440 VCRST              1  283 12471 VCSET2          4  611 13174 WCDONE1          4  611 13173 WCPOPJ
 1  461 12545 VCWRD1             4  611 13147 WCDONE         24 1996 11531 XBLTLP          24 2029 11543 XBLTSA
24 2054 11551 XBLTDN           24 2052 11544 XBLTL1          23 1948 11511 XRYIND          23 1935 11504 XRYMOV
```