insite T.M.

VOLUME I

INTEL SOFTWARE INDEX

AND TECHNOLOGY EXCHANGE

PROGRAM LIBRARY MANUAL

intel®

insite T.M.

INTEL USER'S LIBRARY PROGRAM MANUAL

APRIL 1978

TABLE OF CONTENTS

INTRODUCTION

PROGRAM LIBRARY DEFINITION

Welcome to Insite$^{T.M.}$, Intel's software index and technology exchange. We think you will be impressed and enthusiastic about your new library of programs. Within the contents of Insite$^{T.M.}$ you will find a diversified collection of programs, subroutines, procedures, and macros written by the users of Intel's microprocessors. Thanks to your contributions and continued participation in this Library, we are able to maintain and facilitate this unrestricted interchange of non-proprietary programs among our subscribers.

Each individual subscriber is responsible for making his own independent analysis to determine which programs in the Library are applicable to his requirements. The programs in the Library have been collected and published for your convenience and are *not guaranteed nor supported by Intel.* The verification of each program will be left up to the individual user.

PROGRAM LIBRARY MANUAL

This Program Library Manual includes submittal forms and source listings for those programs three pages and under. Longer programs are represented solely by an abstract which indicates the function of the routine, the required hardware and software and memory requirements. Should you find that you are interested in one of the longer programs, the listing and paper tape may be purchased from Intel at reproduction cost (see Section 3, Ordering Procedure). The Program Library Manual is divided into five major areas of program classification. At the beginning of each section you will find a listing of the contents of that section. At the end of the manual is an alphabetical index of all programs in the Library.

SOURCE TAPES

Source paper tapes and listings of most programs in the Program Library Manual are available for purchase at a minimum reproduction cost. For further information see Section 3, Ordering Procedure.

PROGRAM DISKETTES

Program Diskettes are available for most programs in your Program Library Manual. See Section 3 for prices and ordering procedure.

LIBRARY UPDATES

By filling out and mailing the REGISTRATION CARD in the front of your Library you will be placed on the Membership List and will receive the bi-monthly Program Library Update Package. This package will include all new programs which have been accepted into the Library over the preceding two months. During your subscription year you will not only receive the Library's current programs, but also six update packages. However, unless you mail in your Registration Card you will not receive these updates.

QUALITY ASSURANCE

As it is impossible to test each program submitted to the User's Library, the verification of each program will be left up to users. Included in this section are several "Program Certification and Review Forms". The purpose of this form is to determine whether or not a program functions as its author claims. This form should not be used as a vehicle for program enhancement or elaboration. We appreciate your responses which let us know the accuracy of library programs.

PROGRAM REVISION

Program revisions submitted by the original author are handled in a manner identical to original program submission. That is, the author should submit a completed "Microcomputer User's Library Submittal Form" along with the entire package or relevant documentation. The program which has been revised should be referred to in a cover letter by the reference number which appears in the upper right-hand corner of the original Library Submittal Form.

One member may not revise another member's program. Correctable errors should be documented on a "Program Certification and Review Form" and sent to the User's Library Manager for dissemination to the original authors to allow them to submit a revision.

Enhancement or reprogramming of a program already in the User's Library should be submitted as a new package of documentation and will be treated as an original submission.

3065 Bowers Avenue
Santa Clara, California 95051
Tel: (408) 987-8080* *
TWX: 910-338-0026
TELEX: 34-6372

MICROCOMPUTER AND MEMORY SYSTEM SALES AND MARKETING OFFICES

JANUARY 1978

U.S. AND CANADA SALES OFFICES

ALABAMA
Pen-Tech Associates, Inc
Holiday Office Center
3322 S. Memorial Pkwy.
Huntsville 35801
Tel: (205) 533-0090

Glen White Associates
7844 Horseshoe Trail
Huntsville 35802
Tel: (205) 883-9394

ARIZONA
BFA
4426 North Saddle Bag Trail
Scottsdale 85251
Tel: (602) 994-5400

Intel Corp.
8650 N. 35th Avenue
Phoenix 85021
Tel: (602) 242-7205

CALIFORNIA
Intel Corp.
15335 Morrison
Suite 345
Sherman Oaks 91403
(213) 986-9510
TWX: 910-495-2045

Intel Corp.*
990 E. Arques Ave.
Suite 112
Sunnyvale 94086
Tel: (408) 738-3870
TWX: 910-339-9279
TWX: 910-338-0255

Mac-I
2576 Shattuck
Suite 4B
Berkeley 94704
Tel: (415) 843-7625

Mac-I
P.O. Box 1420
Cupertino 95014
Tel: (408) 257-9880

Intel Corp.*
1651 East 4th Street
Suite 150
Santa Ana 92701
Tel: (714) 835-9642
TWX: 910-595-1114

Mac-I
P.O. Box 8763
Fountain Valley 92708
Tel: (714) 839-3341

Mac-I
20121 Ventura Blvd., Suite 240E
Woodland Hills 91364
Tel: (213) 347-5900

Earle Associates, Inc.
4805 Mercury Street
Suite L
San Diego 92111
Tel: (714) 278-5441

COLORADO
Intel Corp
6000 East Evans Ave.
Bldg. 1, Suite 260
Denver 80222
Tel: (303) 758-8086
TWX: 910-931-2289

Mountaintek
25921 Fern Gulch Road
P.O. Box 1355
Evergreen 80439
Tel: (303) 674-5255

CONNECTICUT
Computer Marketing Association
P.O. Box 172
Fair Haven Station
New Haven 06513
Tel: (203) 789-1013

Intel Corp.
Peacock Alley
1 Padanaram Road, Suite 146
Danbury 06810
Tel: (203) 792-8366
TWX: 710-456-1199

FLORIDA
Pen-Tech Associates, Inc.
201 S.E. 15th Terrace
Deerfield Beach 33431
Tel: (305) 421-4989

Pen-Tech Associates, Inc.
111 So. Maitland Ave.
Maitland 32751
Tel: (305) 645-3444

Intel Corp.
1001 N.W. 62nd Street, Suite 406
Ft. Lauderdale 33309
Tel: (305) 771-0600
TWX: 510-956-9407

Intel Corp.
5151 Adanson Street, Suite 105
Orlando 32804
Tel: (305) 628-2393
TWX: 810-853-9219

GEORGIA
Pen-Tech Associates, Inc.
Suite 305 C
2101 Powers Ferry Roads
Atlanta 30339
Tel: (404) 955-0293

ILLINOIS
Intel Corp.*
1000 Jorie Boulevard
Suite 224
Oakbrook 60521
Tel: (312) 325-9510
TWX: 910-651-5881

Dytek-Central, Inc.
121 So. Wilke Road
Suite 103
Arlington Heights 60005
Tel: (312) 394-3380
TWX: 910-687-2267

INDIANA
Electro Reps Inc
941 E. 66th Street
Indianapolis 46240
Tel: (317) 255-4147
TWX: 810-341-3217

Electro Reps Inc.
810 Valley-O-Pines Pkwy
Ft. Wayne 46825
Tel: (219) 637-6211

IOWA
Technical Representatives, Inc.
St. Andrews Building
1930 St. Andrews Drive N.E.
Cedar Rapids 52402
Tel: (319) 393-5510

KANSAS
Technical Representatives, Inc.
8245 Nieman Road, Suite 100
Lenexa 66214
Tel: (913) 888-0212, 3, & 4
TWX: 910-749-6412

MARYLAND
Glen White Associates
57 West Timonium Road
Timonium 21093
Tel: (301) 252-7742

Intel Corp.*
57 West Timonium Road
Suite 307
Timonium 21093
Tel: (301) 252-7742
TWX: 710-232-1807

Mesa Inc.
11900 Parklawn Drive
Rockville 20852
Tel: Wash. (301) 881-8430
Balto. (301) 792-0021

MASSACHUSETTS
Intel Corp.*
187 Billerica Road, Suite 14A
Chelmsford 01824
Tel: (617) 256-6567
TWX: 710-343-6333

Computer Marketing Associates
257 Crescent Street
Waltham 02154
Tel: (617) 894-7000

MICHIGAN
Intel Corp.
26500 Northwestern Hwy.
Suite 401
Southfield 48075
Tel: (313) 353-0920
TWX: 910-420-1212
TELEX: 2 31143

Lowry & Associates, Inc.
135 W. North Street
Suite 4
Brighton 48116
Tel: (313) 227-7067

MINNESOTA
Intel Corp
8200 Normandale Avenue
Suite 422
Bloomington 55437
Tel: (612) 835-6722
TWX: 910-576-2867

MISSOURI
Technical Representatives, Inc.
Trade Center Bldg.
300 Brookes Drive, Suite 108
Hazelwood 63042
Tel: (314) 731-5200
TWX: 910-762-0618

NEW JERSEY
Intel Corp.
2 Kilmer Road
Edison 08817
Tel: (201) 985-9100
TWX: 710-480-6238

NEW MEXICO
BFA Corporation
312 West Parker Road
Las Cruces 88001
Tel: (505) 523-0601
TWX: 910-983-0543

BFA Corporation
3705 Westerfield, N.E.
Albuquerque 87111
Tel: (505) 292-1212
TWX: 910-989-1157

NEW YORK
Intel Corp.*
350 Vanderbilt Motor Pkwy.
Suite 402
Hauppauge 11787
Tel: (516) 231-3300
TWX: 510-227-6236

NEW YORK (cont.)
Intel Corp.
474 Thurston Road
Rochester 14619
Tel: (716) 328-7340
TWX: 510-253-3841

T-Squared
4054 Newcourt Avenue
Syracuse 13206
Tel: (315) 463-8592
TWX: 710-541-0554

T-Squared
642 Kreag Road
P.O. Box W
Pittsford 14534
Tel: (716) 248-5005
TELEX: 97-8289

Intel Corp.
85 Market Street
Poughkeepsie 12601
Tel: (914) 473-2303
TWX: 510-248-0060

Measurement Technology, Inc.
159 Northern Boulevard
Great Neck 11021
Tel: (516) 482-3500

NORTH CAROLINA
Pen-Tech Associates, Inc.
P.O. Box 5382
Highpoint 27262
Tel: (919) 454-3316

Glen White Associates
3700 Computer Drive
Suite 330
Raleigh 27609
Tel: (919) 787-7016

OHIO
Intel Corp.*
8312 North Main Street
Dayton 45415
Tel: (513) 890-5350
TELEX: 288-004

Intel Corp.*
Chagrin-Brainard Bldg
28001 Chagrin Blvd.
Cleveland 44122
Tel: (216) 464-2736

Lowry & Associates, Inc.
42 East Rahn Road
Suite 100
Dayton 45429
Tel: (513) 435-4795

Lowry & Associates, Inc.
24200 Chagrin Blvd.
Suite 146
Cleveland 44122
Tel: (216) 464-8113

OREGON
ES Chase Company
7100 S.W. Hampton Street
Suite 121
Portland 97223
Tel: (503) 620-9616

PENNSYLVANIA
Intel Corp.*
520 Pennsylvania Ave.
Fort Washington 19034
Tel: (215) 542-9444
TWX: 510-661-2077

Q.E.D. Electronics
300 N. York Road
Hatboro 19040
Tel: (215) 674-9600

PENNSYLVANIA (cont.)
Lowry & Associates, Inc.
Three Parkway Center
Suite 201
Pittsburgh 15220
Tel: (412) 922-5110

TENNESSEE
Glen White Associates
Rt. #12, Norwood S.D.
Jonesboro 37659
Tel: (615) 477-8850

Glen White Associates
2523 Howard Road
Germantown 38138
Tel: (901) 754-0483

Glen White Associates
6446 Ridge Lake Road
Hixon 37343
Tel: (615) 842-7799

TEXAS
Intel Corp.
6776 S.W. Freeway
Suite 550
Houston 77074
Tel: (713) 784-3400

Mycrosystems Marketing Inc.
13777 N. Central Expressway
Suite 405
Dallas 75243
Tel: (214) 238-7157
TWX: 910-867-4763

Mycrosystems Marketing Inc.
6610 Harwin Avenue, Suite 125
Houston 77036
Tel: (713) 783-2900

Intel Corp.*
2925 L.B.J. Freeway
Suite 100
Dallas 75234
Tel: (214) 241-9521
TWX: 910-860-5487

UTAH
Mountaintek
3788 Brockbank Drive
Salt Lake City 84117
Tel: (801) 266-9617

VIRGINIA
Glen White Associates
P.O. Box 1104
Lynchburg 24505
Tel: (804) 384-6920

Glen White Associates
Rt. #1, Box 322
Colonial Beach 22443
Tel: (804) 224-4871

WASHINGTON
E.S. Chase Co.
P.O. Box 80903
Seattle 98108
Tel: (206) 762-4824
TWX: 910-444-2298

CANADA
Intel Corp.
70 Chamberlain Ave.
Ottawa, Ontario K1S 1V9
Tel: (613) 232-8576
TELEX: 053-4419

Multilek, Inc.*
15 Grenfell Crescent
Ottawa, Ontario K2G 0G3
Tel: (613) 825-4553
TELEX: 053-4585

EUROPEAN MARKETING OFFICES

BELGIUM
Intel International*
Rue du Moulin à Papier
51-Boite 1
B-1160 Brussels
Tel: (02) 660 30 10
TELEX: 24814

FRANCE
Intel Corporation, S.A.R.L.*
74, Rue D'Arcueil
Silic 223
94528 Rungis Cedex
Tel: (01) 687 22 21
TELEX: 270475

SCANDINAVIA
Intel Scandinavia A.S.*
Lyngbyvej 32 2nd Floor
DK-2100 Copenhagen East
Denmark
Tel: (01) 18 20 00
TELEX: 19567

Intel Sweden AB*
Box 20092
S-16120 Bromma
Sweden
Tel: (08) 98 53 90
TELEX: 12261

ENGLAND
Intel Corporation (U.K.) Ltd.*
Broadfield House
4 Between Towns Road
Cowley, Oxford OX4 3NB
Tel: (0865) 77 14 31
TELEX: 837203

Intel Corporation (U.K.) Ltd.
46-50 Beam Street
Nantwich, Cheshire CW5 5LJ
Tel: (0270) 62 65 60
TELEX: 36620

GERMANY
Intel Semiconductor GmbH*
Seidlstrasse 27
8000 Muenchen 2
Tel: (089) 55 81 41
TELEX: 523 177

Intel Semiconductor GmbH
Abraham Lincoln Strasse 30
6200 Wiesbaden 1
Tel: (06121) 74855
TELEX: 04186183

Intel Semiconductor GmbH
Ernsthaldenstrasse 17
D-7000 Stuttgart 80
Tel: (0711) 7351506
TELEX: 7255346

Intel Semiconductor GmbH
Wiesenweg 26
D-6272 Niedernhausen
Tel: (06127) 2314

ORIENT MARKETING OFFICES

JAPAN
Intel Japan Corporation*
Flower Hill-Shinmachi East Bldg.
1-23-9, Shinmachi, Setagaya-ku
Tokyo 154
Tel: (03) 426-9261
TELEX: 781-28426

TAIWAN
Taiwan Automation Co.*
2nd Floor, 224
Nanking East Road
Section 3
Taipei
Tel: (02) 7710940-3
TELEX: 11942 TAIAUTO

HONG KONG
China Electronics
Sea Bird House, 9th Floor
22-28 Wyndham Street
Hong Kong

KOREA
Koram Digital
Sam Yung Bldg. #303
71-2 Bukchang - Dong Chung-Ku
Seoul 100

INTERNATIONAL DISTRIBUTORS

ARGENTINA
S.I.E.S.A.
Av. Pte. Roque Saenz Pena 1142 9B
1035 Buenos Aires
Tel: 35-6784

AUSTRALIA
A. J. Ferguson (Adelaide) PTY, Ltd.
44 Prospect Rd.
Prospect 5082
South Australia 17005
Tel: 269-1244
TELEX: 82635

A. J. Ferguson Electronics
34 Herbert Street
West Ryde, N.S.W. 2114
Tel: AC8 269-1244
TELEX: 82635

Warburton-Frankie (Sydney) Pty. Ltd.
199 Parramatta Road
Auburn, N.S.W. 2114
Tel: 648-1711, 648-1381
TELEX: WARFRAN AA 22265

Warburton-Frankie Industries (Melbourne) Pty. Ltd.
220 Park Street
South Melbourne, Victoria 3205
Tel: 699-4989
TELEX: WARFRAN AA 31370

AUSTRIA
Bacher Elektronische Gerate GmbH
Meidlinger Hauptstrasse 78
A 1120 Vienna
Tel: (0222) 83 63 96
TELEX: (01) 1532

BELGIUM
Ineico Belgium S.A.
Avenue Val Duchesse, 3
B-1160 Brussels
Tel: (02) 660 00 12
TELEX: 25441

DENMARK
Lyngso Komponent A/S
Osmarken 4
2860 Soborg
Tel: (01) 67 00 77
TELEX: 22990

Scandinavian Semiconductor Supply A/S
Vannasgade 18
DK-2200 Copenhagen N
Tel: (01) 83 50 90
TELEX: 19037

FINLAND
Oy Fintronic AB
Loennrotinkatu 35D
SF 00180
Helsinki 18
Tel: (90) 664 451
TELEX: 12426

FRANCE
Tekelec Airtronic
Cite des Bruyeres
Rue Carle Vernet
92310 Sevres
Tel: (1) 027 75 35
TELEX: 250997

GERMANY
Alfred Neye Enatachnik GmbH
Schillerstrasse 14
D-2085 Quickborn-Hamburg
Tel: (04106) 6121
TELEX: 02-13590

Electronic 2000 Vertriebs GmbH
Neumarkter Strasse 75
D-8000 Muenchen 80
Tel: (089) 434061
TELEX: 522561

Jermyn GmbH
Postfach 1146
D-6277 Kamberg
Tel: (06434) 6005
TELEX: 484426

INDIA
Electronics International
128 Mahatma Gandhi Road
Secunderabad
Tel: 73720
TELEX: 015-363
CABLE: GUNTICO

ISRAEL
Eastronics Ltd.*
11 Rozanis Street
P.O. Box 39300
Tel-Aviv
Tel: 475151
TELEX: 33638

ITALY
Eledra 3S S.P.A.*
Viale Elvezia, 18
20154 Milan,
Tel: (02) 3493041
TELEX: 39332

Eledra 3S S.P.A.*
Via Paolo Gaidano, 141 D
10137 Torino
TEL: (011) 30 97 097 - 30 97 114

Eledra 3S S.P.A.*
Via Giuseppe Valmarana, 63
00139 Rome, Italy
Tel: (06) 81 27 290 - 81 27 324
TELEX: 63051

JAPAN
Pan Electron
No. 1 Higashikata-Machi
Midori-Ku, Yokohama 226
Tel: (045) 471-8811
TELEX: 781-4773

Ryoyo Electric Corp.
Konwa Bldg.
1-12-22, Tsukiji, 1-Chome
Chuo-Ku, Tokyo 104
Tel: (03) 543-7711

Nippon Micro Computer Co. Ltd.
Mutsumi Bldg. 4-5-21 Kojimachi
Chiyoda-ku, Tokyo 102
Tel: (03) 230-0041

NETHERLANDS
C.N. Rood BV
Cort Vender
Lindenstraat, 13
Postbus 42
Rijswijk 2H2100
Tel: 070-996360
TELEX: 31238

Ineico Nederland
AFD Elektronic
Joan Muyskenweg 22
NL-1006 Amsterdam
Tel: (020) 934824
TELEX: 14622

NEW ZEALAND
W. K. McLean Ltd.
103-5 Felton Matthew Avenue
Glenn Innes, Auckland
Tel: 587-037
TELEX: NZ2763

NORWAY
Nordisk Elektronik (Norge) A/S
Mustads Vei 1
N-Oslo 2
Tel: (02) 55 38 93
TELEX: 16963

PORTUGAL
Ditram
Componentes E Electronica LDA
Av. Miguel Bombarda, 133
Lisboa 1
Tel: 119 45 313

SOUTH AFRICA
Electronic Building Elements
P.O. Box 4609
Pretoria
Tel: 78 92 21
TELEX: 30181

SPAIN
Interface
Ronda San Pedro 22
Barcelona 10
Tel: 301 78 51

SWEDEN
Nordisk Electronik AB
Fack
S-10380 Stockholm 7
Tel: (08) 248340
TELEX: 10547

SWITZERLAND
Industrade AG
Gemsenstrasse 2
Postcheck 80 - 21190
CH-8021 Zurich
Tel: (01) 60 22 30
TELEX: 56788

UNITED KINGDOM
Rapid Recall, Ltd.
11-15 Betterton Street
Drury Lane
London WC2H 9BS
Tel: (01) 379-6741
TELEX: 28752

G.E.C. Semiconductors Ltd.
East Lane
Wembley HA9 7PP
Middlesex
Tel: (01) 904-9303
TELEX: 923429

Jermyn Industries
Vestry Estate
Sevenoaks, Kent
Tel: (0732) 5[illegible]14
TELEX: 9514[illegible]

**Sintrom Electronics Ltd.*
Arkwright Road
Reading, Berkshire RG20LS
Tel: (0734) 85464
TELEX: 847395

**Note New Telephone Number

***Field Application Location**

MICROCOMPUTER AND MEMORY COMPONENT SALES AND MARKETING OFFICES

3065 Bowers Avenue
Santa Clara, California 95051
Tel: (408) 987-8080**
TWX: 910-338-0026
TELEX: 34-6372

JANUARY 1978

U.S. AND CANADIAN SALES OFFICES

ALABAMA
Glen White Associates
7844 Horseshoe Trail
Huntsville 35802
Tel: (205) 883-9394

ARIZONA
Intel Corp.
8650 N. 35th Avenue
Phoenix 85021
Tel: (602) 242-7205

CALIFORNIA
Intel Corp.
15335 Morrison
Suite 345
Sherman Oaks 91403
(213) 986-9510
TWX: 910-495-2045

Intel Corp.*
990 E. Arques Ave.
Suite 112
Sunnyvale 94086
Tel: (408) 738-3870
TWX: 910-339-9279
TWX: 910-338-0255

Mac-l
2576 Shattuck
Suite 4B
Berkeley 94704
Tel: (415) 843-7625

Mac-l
P.O. Box 1420
Cupertino 95014
Tel: (408) 257-9880

Earle Associates, Inc.
4805 Mercury Street
Suite L
San Diego 92111
Tel: (714) 278-5441
TWX: 910-335-1585

Mac-l
P.O. Box 8763
Fountain Valley 92708
Tel: (714) 839-3341

Intel Corp.*
1651 East 4th Street
Suite 150
Santa Ana 92701
Tel: (714) 835-9642
TWX: 910-595-1114

COLORADO
Intel Corp.
6000 East Evans Ave.
Bldg. 1, Suite 260
Denver 80222
Tel: (303) 758-8086
TWX: 910-931-2289

CONNECTICUT
Intel Corp.
Peacock Alley
1 Padanaram Road, Suite 146
Danbury 06810
Tel: (203) 792-8366
TWX: 710-456-1199

FLORIDA
Intel Corp.
1001 N.W. 62nd Street, Suite 406
Ft. Lauderdale 33309
Tel: (305) 771-0600
TWX: 510-956-9407

Intel Corp.
5151 Adanson Street, Suite 105
Orlando 32804
Tel: (305) 628-2393
TWX: 810-853-9219

ILLINOIS
Intel Corp.
1000 Jorie Boulevard
Suite 224
Oakbrook 60521
Tel: (312) 325-9510
TWX: 910-651-5881

INDIANA
Electro Reps Inc.
941 E. 86th Street
Indianapolis 46240
Tel: (317) 255-4147
TWX: 810-341-3217

Electro Reps Inc.
810 Valley-O-Pines Pkwy.
Ft. Wayne 46825
Tel: (219) 637-6211

IOWA
Technical Representatives, Inc.
St. Andrews Building
1930 St. Andrews Drive N.E.
Cedar Rapids 52402
Tel: (319) 393-5510

KANSAS
Technical Representatives, Inc.
8245 Nieman Road, Suite 105
Lenexa 66214
Tel: (913) 888-0212, 3, & 4
TWX: 910-749-6412

MARYLAND
Glen White Associates
57 West Timonium Road
Timonium 21093
Tel: (301) 252-6360

Intel Corp.*
57 West Timonium Road
Suite 307
Timonium 21093
Tel: (301) 252-7742
TWX: 710-232-1807

MASSACHUSETTS
Intel Corp.*
187 Billerica Road, Suite 14A
Chelmsford 01824
Tel: (617) 256-6567
TWX: 710-343-6333

MICHIGAN
Intel Corp.
26500 Northwestern Hwy.
Suite 401
Southfield 48075
Tel: (313) 353-0920
TWX: 910-420-1212
TELEX: 2 31143

MINNESOTA
Intel Corp.
8200 Normandale Avenue
Suite 422
Bloomington 55437
Tel: (612) 835-6722
TWX: 910-576-2867

MISSOURI
Technical Representatives, Inc.
Trade Center Bldg.
300 Brookes Drive, Suite 108
Hazelwood 63042
Tel: (314) 731-5200
TWX: 910-762-0618

NEW JERSEY
Intel Corp.
2 Kilmer Road
Edison 08817
Tel: (201) 985-9100
TWX: 710-480-6238

NEW YORK
Intel Corp.*
350 Vanderbilt Motor Pkwy.
Suite 402
Hauppauge 11787
Tel: (516) 231-3300
TWX: 510-227-6236

Intel Corp.
474 Thurston Road
Rochester, N.Y. 14619
Tel: (716) 328-7340
TWX: 510-253-3841

T-Squared
4054 Newcourt Ave.
Syracuse 13206
Tel: (315) 463-8592
TWX: 710 541-0554

T-Squared
642 Kreag Road
P.O. Box W
Pittsford 14534
Tel: (716) 248-5005
TELEX: 97-8289

NEW YORK (cont.)
Intel Corp.
85 Market Street
Poughkeepsie, New York 12601
Tel: (914) 473-2303
TWX: 510-248-0060

NORTH CAROLINA
Glen White Associates
3700 Computer Dr., Suite 330
Raleigh 27609
Tel: (919) 787-7016

OHIO
Intel Corp.*
8312 North Main Street
Dayton 45415
Tel: (513) 890-5350
TELEX: 288-004

Intel Corp.*
Chagrin-Brainard Bldg.
28001 Chagrin Blvd.
Cleveland 44122
Tel: (216) 464-2736

OREGON
E.S. Chase Company
7100 S.W. Hampton Street
Suite 121
Portland 97223
Tel: (503) 620-9616

PENNSYLVANIA
Intel Corp.*
520 Pennsylvania Ave.
Fort Washington 19034
Tel: (215) 542-9444
TWX: 510-661-2077

TENNESSEE
Glen White Associates
Rt. #12, Norwood S.D.
Jonesboro 37659
Tel: (615) 477-8850

Glen White Associates
2523 Howard Road
Germantown 38138
Tel: (901) 754-0483

Glen White Associates
6446 Ridge Lake Road
Hixon 37343
Tel: (615) 842-7799

TEXAS
Intel Corp.
6776 S.W. Freeway
Suite 550
Houston 77074
Tel: (713) 784-3400

Mycrosystems Marketing Inc.
13777 N. Central Expressway
Suite 405
Dallas 75243
Tel: (214) 238-7157
TWX: 910-867-4763

Mycrosystems Marketing Inc.
6610 Harwin Avenue, Suite 125
Houston 77036
Tel: (713) 783-2900

Intel Corp.*
2925 L.B.J. Freeway
Suite 100
Dallas 75234
Tel: (214) 241-9521
TWX: 910-860-5487

VIRGINIA
Glen White Associates
P.O. Box 1104
Lynchburg 24505
Tel: (804) 384-6920

Glen White Associates
Rt. #1, Box 322
Colonial Beach 22443
Tel: (804) 224-4871

WASHINGTON
E.S. Chase Co.
P.O. Box 80903
Seattle 98108
Tel: (206) 762-4824
Twx: 910-444-2298

CANADA
Intel Corp.
70 Chamberlain Ave.
Ottawa, Ontario K1S 1V9
Tel: (613) 232-8576
TELEX 053-4419

Multitek Inc.*
15 Grenfell Crescent
Ottawa, Ontario K2G 0G3
Tel: (613) 825-4553
TELEX 053-4585

EUROPEAN MARKETING OFFICES

BELGIUM
Intel International*
Rue du Moulin à Papier
51-Boite 1
B-1160 Brussels
Tel: (02) 660 30 10
TELEX: 24814

FRANCE
Intel Corporation, S.A.R.L.*
74, Rue D'Arcueil
Silic 223
94528 Rungis Cedex
Tel. (01) 687 22 21
TELEX: 270475

SCANDINAVIA
Intel Scandinavia A/S*
Lyngbyvej 32 2nd Floor
DK-2100 Copenhagen East
Denmark
Tel: (01) 18 20 00
TELEX: 19567

Intel Sweden AB*
Box 20092
S-16120 Bromma
Sweden
Tel. (08) 98 53 90
TELEX: 12261

ENGLAND
Intel Corporation (U.K.) Ltd.*
Broadfield House
4 Between Towns Road
Cowley, Oxford OX4 3NB
Tel. (0865) 77 14 31
TELEX 837203

Intel Corporation (U.K.) Ltd.
46-50 Beam Street
Nantwich, Cheshire CW5 5LJ
Tel. (0270) 62 65 60
TELEX 36620

GERMANY
Intel Semiconductor GmbH*
Seidlstrasse 27
8000 Muenchen 2
Tel (089) 55 81 41
TELEX 523 177

Intel Semiconductor GmbH
Abraham Lincoln Strasse 30
6200 Wiesbaden 1
Tel. (06121) 74855
TELEX 04186183

Intel Semiconductor GmbH
Ernsthaldenstrasse 17
D-7000 Stuttgart 80
Tel. (0711) 7351506
TELEX 7255346

Intel Semiconductor GmbH
Wiesenweg 26
D-6272 Niedernhausen
Tel. (06127) 2314

ORIENT MARKETING OFFICES

JAPAN
Intel Japan Corporation*
Flower Hill-Shinmachi East Bldg.
1-23-9, Shinmachi, Setagaya-ku
Tokyo 154
Tel. (03) 426-9261
TELEX: 781-28426

TAIWAN
Taiwan Automation Co.*
2nd Floor, 224
Nanking East Road
Section 3
Taipei
Tel: (02) 7710940-3
TELEX: 11942 TAIAUTO

HONG KONG
China Electronics
Sea Bird House, 9th Floor
22-28 Wyndham Street
Hong Kong

KOREA
Koram Digital
Sam Yung Bldg. #303
71-2 Bukchang - Dong Chung-Ku
Seoul 100

INTERNATIONAL DISTRIBUTORS

ARGENTINA
S.I.E.S.A.
Av. Pte. Roque Saenz Pena 1142 9B
1035 Buenos Aires
Tel: 35-6784

AUSTRALIA
A. J. Ferguson (Adelaide) PTY, Ltd.
44 Prospect Rd.
Prospect 5082
South Australia 17005
Tel: 269-1244
TELEX: 82635

A. J. Ferguson Electronics
34 Herbert Street
West Ryde, N.S.W. 2114
Tel: AC8 269-1244
TELEX: 82635

Warburton-Frankie (Sydney) Pty. Ltd.
199 Parramatta Road
Auburn, N.S.W. 2114
Tel: 648-1711, 648-1381
TELEX: WARFRAN AA 22265

Warburton-Frankie Industries
(Melbourne) Pty. Ltd.
220 Park Street
South Melbourne, Victoria 3205
Tel: 699-4999
TELEX: WARFRAN AA 31370

AUSTRIA
Bacher Elektronische Gerate GmbH
Meidlinger Hauptstrasse 78
A 1120 Vienna
Tel: (0222) 83 63 96
TELEX: (01) 1532

BELGIUM
Inelco Belgium S.A.
Avenue Val Duchesse, 3
B-1160 Brussels
Tel: (02) 660 00 12
TELEX: 25441

DENMARK
Lyngso Komponent A/S
Ostmarken 4
2860 Soborg
Tel: (01) 67 00 77
TELEX: 22990

Scandinavian Semiconductor
Supply A/S
Nannasgade 18
DK-2200 Copenhagen N
Tel: (01) 83 50 90
TELEX: 19037

FINLAND
Oy Fintronic AB
Loennrotinkatu 35D
SF 00180
Helsinki 18
Tel: (90) 664 451
TELEX: 12426

FRANCE
Tekelec Airtronic
Cite des Bruyeres
Rue Carle Vernet
92310 Sevres
Tel: (1) 027 75 35
TELEX: 250997

GERMANY
Alfred Neye Enatechnik GmbH
Schillerstrasse 14
D-2085 Quickborn-Hamburg
Tel. (04106) 6121
TELEX: 02-13590

Electronic 2000 Vertriebs GmbH
Neumarkter Strasse 75
D-8000 Muenchen 80
Tel: (089) 434061
TELEX: 522561

Jermyn GmbH
Postfach 1146
D-6277 Kamberg
Tel: (06434) 6005
TELEX: 484426

INDIA
Electronics International
128 Mahatma Gandhi Road
Secunderabad
Tel: 73720
TELEX: 015-363
CABLE: GUNTICO

ISRAEL
Eastronics Ltd.*
11 Rozanis Street
P.O. Box 39300
Tel-Aviv
Tel: 475151
TELEX: 33638

ITALY
Eledra 3S S.P.A.*
Viale Elvezia, 18
20154 Milan
Tel. (02) 3493041
TELEX: 39332

Eledra 3S S.P.A.*
Via Paolo Gaidano, 141 D
10137 Torino
TEL. (011) 30 97 097 - 30 97 114

Eledra 3S S.P.A.*
Via Giuseppe Valmarana, 63
00139 Rome, Italy
Tel. (06) 81 27 290 - 81 27 324
TELEX: 63051

JAPAN
Pan Electron
No. 1 Higashikata-Machi
Midori-Ku, Yokohama 226
Tel. (045) 471-8811
TELEX: 781-4773

JAPAN (cont.)
Ryoyo Electric Corp.
Konwa Bldg.
1-12-22, Tsukiji, 1-Chome
Chuo-Ku, Tokyo 104
Tel: (03) 543-7711

Nippon Micro Computer Co. Ltd.
Mutsumi Bldg. 4-5-21 Kojimachi
Chiyoda-ku, Tokyo 102
Tel. (03) 230-0041

NETHERLANDS
Inelco Nederland
AFD Elektronic
Joan Muyskenweg 22
NL-1006 Amsterdam
Tel: (020) 934824
TELEX: 14622

NEW ZEALAND
W. K. McLean Ltd.
103-5 Felton Matthew Avenue
Glenn Innes, Auckland
Tel. 587-037
TELEX: NZ2763

NORWAY
Nordisk Elektronik (Norge) A/S
Mustads Vei 1
N-Oslo 2
Tel: (02) 55 38 93
TELEX: 16963

PORTUGAL
Ditram
Componentes E Electronica LDA
Av. Miguel Bombarda, 133
Lisboa 1
Tel: 119 45 313

SOUTH AFRICA
Electronic Building Elements
P.O. Box 4609
Pretoria
Tel: 78 92 21
TELEX: 30181

SPAIN
Interface
Ronda San Pedro 22
Barcelona 10
Tel: 301 78 51

SWEDEN
Nordisk Electronik AB
Fack
S-10380 Stockholm 7
Tel (08) 248340
TELEX: 10547

SWITZERLAND
Industrade AG
Gemsenstrasse 2
Postcheck 80 - 21190
CH-8021 Zurich
Tel. (01) 60 22 30
TELEX: 56788

UNITED KINGDOM
Rapid Recall, Ltd.
11-15 Betterton Street
Drury Lane
London WC2H 9BS
Tel: (01) 379-6741
TELEX: 28752

G.E.C. Semiconductors Ltd.
East Lane
Wembley HA9 7PP
Middlesex
Tel: (01) 904-9303
TELEX: 923429

Jermyn Industries
Vestry Estate
Sevenoaks, Kent.
Tel: (0732) 50144
TELEX: 95142

**Note New Telephone Number

*Field Application Location

3065 Bowers Avenue
Santa Clara, California 95051
Tel: (408) 987-8080**
TWX: 910-338-0026
TELEX: 34-6372

U.S. AND CANADIAN DISTRIBUTORS

U.S. AND CANADIAN DISTRIBUTORS

JANUARY 1978

ALABAMA

†*Hamilton/Avnet Electronics
805 Oser Drive NW
Huntsville 35805
Tel: (205) 533-1170

Pioneer
1207 Putman Drive NW
Huntsville 35805
Tel: (205) 837-9300

ARIZONA

†*Hamilton/Avnet Electronics
8155 North 21st Street
Phoenix 85021
Tel: (602) 275-7851

Liberty/Arizona
3130 N. 27th Avenue
Phoenix 85017
Tel: (602) 257-1272
TELEX: 910-951-4282

CALIFORNIA

†*Avnet Electronics
350 McCormick Avenue
Costa Mesa 92626
Tel: (714) 754-6111
Tel: (213) 558-2345

†*Hamilton/Avnet Electronics
575 E. Middlefield Road
Mountain View 94040
Tel: (415) 961-8600

†*Hamilton/Avnet Electronics
8917 Complex Drive
San Diego 92123
Tel: (714) 279-2421

†*Hamilton Electro Sales
10912 W. Washington Boulevard
Culver City 90230
Tel: (213) 558-2121

†Cramer/San Francisco
720 Palomar Avenue
Sunnyvale 94086
Tel: (408) 739-3011

Cramer/Los Angeles
17201 Daimler Street
Irvine 92714
Tel: (714) 979-3000

†Liberty Electronics
124 Maryland Street
El Segundo 90245
Tel: (213) 322-8100
Tel: (714) 638-7601
TWX: 910-348-7140

†Liberty/San Diego
8284 Mercury Court
San Diego 92111
Tel: (714) 565-9171
TELEX: 910-335-1590

†Elmar Electronics
2288 Charleston Road
Mountain View 94040
Tel: (415) 961-3611
TELEX: 910-379-6437

COLORADO

†Elmar/Denver
6777 E. 50th Avenue
Commerce City 80022
Tel: (303) 287-9611
TWX: 910-936-0770

†*Hamilton/Avnet Electronics
5921 No. Broadway
Denver 80216
Tel: (303) 545-1212

CONNECTICUT

†Cramer/Connecticut
35 Dodge Avenue
North Haven 06473
Tel: (203) 239-5641

†*Hamilton/Avnet Electronics
643 Danbury Road
Georgetown 06829
Tel: (203) 762-0361

Harvey Electronics
112 Main Street
Norwalk 06851
Tel: (203) 853-1515

FLORIDA

Cramer/E.W. Hollywood
4035 No. 29th Avenue
Hollywood 33020
Tel: (305) 921-7878

†*Hamilton/Avnet Electronics
6800 Northwest 20th Ave.
Ft. Lauderdale 33309
Tel: (305) 971-2900

Cramer/EW Orlando
345 No. Graham Ave.
Orlando 32814
Tel: (305) 894-1511

Pioneer
6220 S. Orange Blossom Trail
Suite 412
Orlando 32809
Tel: (305) 859-3600

GEORGIA

†Cramer
6456 Warren Drive
Norcross 30071
Tel: (404) 448-9050

**Note New Telephone Number

GEORGIA (cont.)

†*Hamilton/Avnet Electronics
6700 I 85, Access Road, #11
Norcross 30071
Tel: (404) 448-0800

ILLINOIS

†Cramer/Chicago
1911 So. Busse Rd.
Mt. Prospect 60056
Tel: (312) 593-8230

†*Hamilton/Avnet Electronics
3901 No. 25th Ave.
Schiller Park 60176
Tel. (312) 678-6310

Pioneer/Chicago
1551 Carmen Drive
Elk Grove Village 60006
Tel: (312) 437-9680

INDIANA

†Pioneer/Indiana
6408 Castleplace Drive
Indianapolis 46250
Tel: (317) 849-7300

Sheridan Sales
8790 Purdue Road
Indianapolis 46268
Tel: (317) 297-3146

KANSAS

†*Hamilton/Avnet Electronics
9219 Quivira Road
Overland Park 66215
Tel: (913) 888-8900

MARYLAND

†Cramer/EW Washington
16021 Industrial Drive
Gaithersburg 20760
Tel: (301) 948-0110

†*Hamilton Avnet
7235 Standard Drive
Hanover 21076
Tel: (301) 796-5000

†Pioneer/Washington
9100 Gaither Road
Gaithersburg 20760
Tel: (301) 948-0710
TWX: 710-828-0545

MASSACHUSETTS

†Cramer Electronics Inc.
85 Wells Avenue
Newton 02159
Tel: (617) 969-7700

†*Hamilton/Avnet Electronics
100 E. Commerce Way
Woburn 01801
Tel: (617) 933-8000

MICHIGAN

†Sheridan Sales Co.
24543 Indoplex Circle
Farmington Hills 48024
Tel: (313) 477-3800

†Pioneer/Michigan
13485 Stamford
Livonia 48150
Tel: (313) 525-1800

†*Hamilton/Avnet Electronics
32487 Schoolcraft Road
Livonia 48150
Tel: (313) 522-4700
TWX: 810-242-8775

MINNESOTA

†Industrial Components
5280 West 74th Street
Minneapolis 55435
Tel: (612) 831-2666

†Cramer/Bonn
7275 Bush Lake Road
Edina 55435
Tel: (612) 835-7811

†*Hamilton/Avnet Electronics
7683 Washington Avenue So.
Edina 55435
Tel: (612) 941-3801

MISSOURI

†*Hamilton/Avnet Electronics
396 Brookes Lane
Hazelwood 63042
Tel: (314) 731-1144

Sheridan Sales
110 S. Hwy. 67, Suite 10
Florissant 63031
Tel: (314) 837-5200

NEW JERSEY

Cramer/Pennsylvania, Inc.
12 Springdale Road
Cherry Hill Industrial Center
Cherry Hill 08034
Tel: (609) 424-5993
TWX: 710-896-0908

†*Hamilton/Avnet Electronics
218 Little Falls Road
Cedar Grove 07009
Tel: (201) 239-0800
TWX: 710-994-5787

Cramer/New Jersey
1 Cardinal Drive
Little Falls 07424
Tel: (201) 785-4300

NEW JERSEY (cont.)

†Harvey Electronics
389 Passaic Avenue
Fairfield 07006
Tel: (201) 227-1262

†*Hamilton/Avnet Electronics
113 Gaither Drive
East Gate Industrial Park
Mt. Laurel 08057
Tel: (609) 234-2133
TWX: 710-897-1405

NEW MEXICO

†*Hamilton/Avnet Electronics
2524 Baylor Drive, S.E.
Albuquerque 87119
Tel: (505) 765-1500

NEW YORK

†Cramer/Rochester
3000 Winton Road South
Rochester 14623
Tel: (716) 275-0300

†*Hamilton/Avnet Electronics
167 Clay Road
Rochester 14623
Tel: (716) 442-7820

Cramer/Syracuse
6716 Joy Road
East Syracuse 13057
Tel: (315) 437-6671

†*Hamilton/Avnet Electronics
6500 Joy Road
E. Syracuse 13057
Tel: (315) 437-2641

†Cramer/Long Island
129 Oser Avenue
Hauppauge, L.I. 11787
Tel: (516) 231-5600
TWX: 510-227-9863

†*Hamilton/Avnet Electronics
70 State Street
Westbury, L.I. 11590
Tel: (516) 333-5800
TWX: 510-222-8237

†Harvey Electronics
60 Crossways Park West
Woodbury 11797
Tel: (516) 921-8700

NORTH CAROLINA

†Cramer Electronics
938 Burke Street
Winston-Salem 27102
Tel: (919) 725-8711

Pioneer/Carolina
2906 Baltic Avenue
Greensboro 27406
Tel: (919) 273-4441
TWX: 510-925-1114

Hamilton/Avnet Electronics, Inc.
2803 Industrial Drive
Raleigh 27609
Tel: (919) 829-8030

OHIO

†Sheridan Sales Co.
2501 Neff Road
Dayton 45414

†Cramer/Cleveland
5835 Harper Road
Cleveland 44139
Tel: (216) 248-8400

†*Hamilton/Avnet Electronics
118 Westpark Road
Dayton 45459
Tel: (513) 433-0610
TWX: 810-450-2531

†Pioneer/Dayton
1900 Troy Street
Dayton 45404
Tel: (513) 236-9900

†Sheridan Sales Co.
10 Knollcrest Drive
Cincinnati 45222
Tel: (513) 761-5432
TWX: 810-461-2670

†Pioneer/Cleveland
4800 E. 131st Street
Cleveland 44105
Tel: (216) 587-3600

†*Hamilton/Avnet Electronics
761 Beta Drive, Suite E
Cleveland 44143
Tel: (216) 461-1400

†Sheridan Sales Co.
23224 Commerce Park Road
Beachwood 44122
Tel: (216) 831-0130

OKLAHOMA

†Components Specialties, Inc.
7920 E. 40th Street
Tulsa 74145
Tel: (918) 664-2820

OREGON

†Almac/Stroum Electronics
4475 S.W. Scholls Ferry Rd.
Portland 97225
Tel: (503) 292-3534

PENNSYLVANIA

†Sheridan Sales Co.
1717 Penn Avenue, Suite 5009
Pittsburgh 15221
Tel: (412) 244-1640

Pioneer/Pittsburgh
560 Alpha Drive
Pittsburgh 15238
Tel: (412) 782-2300

PENNSYLVANIA (cont.)

Pioneer/Delaware
141 Gibraltar Road
Horsham 19044
Tel: (215) 674-4000
TWX: 510-665-6778

TENNESSEE

Sheridan Sales Co.
6900 Office Park Circle
Knoxville 37919
Tel: (615) 588-5836

TEXAS

Component Specialties Inc.
8330 Burnett Road, Suite 101
Austin 78758
Tel: (512) 459-3308

†Cramer Electronics
13740 Midway Road
Dallas 75240
Tel: (214) 661-9300

†*Hamilton/Avnet Electronics
4445 Sigma Road
Dallas 75240
Tel: (214) 661-8661

†*Hamilton/Avnet Electronics
3939 Ann Arbor
Houston 77063
Tel: (713) 780-1771

†Component Specialties, Inc.
10907 Shady Trail, Suite 101
Dallas 75220
Tel: (214) 357-6511

†Component Specialties, Inc.
8585 Commerce Park Drive, Suite 590
Houston 77036
Tel: (713) 771-7237

UTAH

†*Hamilton/Avnet Electronics
1585 West 2100 South
Salt Lake City, 84119
Tel: (801) 972-2800

WASHINGTON

†*Hamilton/Avnet Electronics
13407 Northrup Way
Bellevue 98005
Tel: (206) 746-8750

†Almac/Stroum Electronics
5811 Sixth Ave. South
Seattle 98108
Tel: (206) 763-2300

†Liberty Electronics
1750 132nd Avenue NE
Bellevue 98005
Tel: (206) 763-8200

WISCONSIN

†*Hamilton/Avnet
2975 Moorland Road
New Berlin 53151
Tel: (414) 784-4510

CANADA

ALBERTA

†L. A. Varah Ltd.
4742 14th Street N.E.
Calgary T2E 6LT
Tel: (403) 276-8818
Telex: 13 825 89 77

BRITISH COLUMBIA

†L.A. Varah Ltd.
2077 Alberta Street
Vancouver V5Y 1C4
Tel: (604) 873-3211
TWX: 610-929-1068
Telex: 04 53167

ONTARIO

L.A. Varah, Ltd.
505 Kenora Avenue
Hamilton L8E-3P2
Tel: (416) 561-9311
TELEX: 061-8349

†*Hamilton/Avnet Electronics
3688 Nashua Drive, Unit GH
Mississauga L4V IM5
Tel: (416) 677-7432
TWX: 610-492-8867

†*Hamilton/Avnet Electronics
1735 Courtwood Cresc.
Ottawa K2C 3J2
Tel: (613) 226-1700
TWX: 610 562-1906

†Zentronics
141 Catherine Street
Ottawa, Ontario K2P 1C3
Tel: (613) 238-6411

†Zentronics
99 Norfinch Dr.
Downsview, Ontario M3N 1W8
Tel: (416) 635-2822
Telex: 02-021694

QUEBEC

†*Hamilton/Avnet Electronics
2670 Paulus
St. Laurent H4S 1G2
Tel: (514) 331-6443
TWX: 610-421-3731

†Zentronics
8146 Montview Road
Town of Mount Royal, Montreal
Quebec H4P 2L7
Tel: (514) 735-5361
Telex: 05-827535

†Microcomputer System Technical Demonstrator Centers
*Microcomputer System Spares Order Point

DISKETTE AND PAPER TAPE ORDERING PROCEDURE

DISKETTE ORDERING PROCEDURE

The guidelines below apply to all diskette orders, with the exception of Resident Language Translators (Section 9). Program diskettes, unless otherwise noted on the author's program abstract, are available in source code only.

1. Three program minimum on all diskette orders.
2. A three program diskette is priced at $55.00; add $15.00 for each additional program.
3. Your check or money order is required with orders under $100.00.

Listings will not be included with diskette orders. However, when required, appropriate User's Guide will be sent. Diskette Program Order Forms are enclosed in your Insit$_e$ TM Manual, Section 3. Send your order to Intel Corporation, User's Library, 3065 Bowers Avenue, Santa Clara, CA 95051.

RESIDENT LANGUAGE TRANSLATORS

Due to program length, programs in Section 9 are priced separately. Insit$_e$ members can either order the programs individually or as a complete package. Program Diskettes are available in source code unless otherwise noted.

Reference Number	Description	Price
F1	Small	$ 35.00
F2	ML/80 Structured Assembler for 8080	$ 70.00
F3	Basic M	$ 35.00
F7	LLL Basic Interpreter	$ 35.00
F8	Octal Debugging Routine	
F9	4004/4040 Cross Assembler for Intellec 800 (Paper Tape and Diskette available in object code only)	$ 25.00
F10	8008 Cross Assembler for Intellec 800 (Paper Tape and Diskette available in object code only)	$ 25.00
F11	8080 Macro Assembler for Intellec 8/MOD80 (Source Listing available only)	$ 15.00
F12	8008 Macro Assembler for Intellec 8/MOD8 (Source Listing available only)	$ 15.00
F13	Sequential Pascal Compiler PAS80 (Available on diskette only)	$ 70.00
F14	RIA80	$ 25.00
F15	LLL/Chernack Basic Interpreter	$ 35.00

Package Price for all Resident Language Translators (Section 9).....$320.00

PAPER TAPE ORDERING PROCEDURE

The majority of Library Programs are available in source code. However, paper tapes for Resident Language Translators, Section 9, are only offered in object code. Program Order Forms are in this section of your Insite Program Library Manual. A prepaid $15.00 handling fee is charged for each program, which includes paper tape and listing. Send your order to Intel Corporation, User's Library, 3065 Bowers Avenue, Santa Clara, California 95051.

PROGRAM SUBMITTAL PROCEDURE

All programs submitted for our review must follow the guidelines listed below:

1. Programs must be written in standard Intel Assembly Language, PL/M or FORTRAN. These languages are documented in the following manuals:

   a. 8080/8085 Assembly Language Programming Manual #9800301C

   b. 8008 Assembly Language Programming Manual #98-019B

   c. 4004/4040 Assembly Language Programming Manual #98-025A

   d. 8008/8080 PL/M Programming Manual #98-108A

   e. PL/M 80 Programming Manual #98-268B

   f. MCS 48/UPI 41 Assembly Language Manual #98-255B

   g. FORTRAN-80 Programming Manual #9800481

2. Submitted programs must be error free. No consideration will be given to partial programs or duplication of existing programs. All accepted programs should assemble or compile correctly without syntax errors. An example of program operation must be included.

3. Programs must be supplied in both machine readable source form and machine generated listing along with a completed submittal form (copies of this form can be found in this section). On the back of the Library Submittal Form are detailed instructions for program submittal which should be closely adhered to. These documentation standards are maintained to assure the usability of each library program by other users.

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/4040 ☐ 8008 ☐ 8080 ☐ 8048 ☐ 8085 ☐ Other_____ (use additional sheets if necessary)

Program Title

Function

Required Hardware

Required Software

Input Parameters

Output Results

Registers Modified:	Programmer:
RAM Required:	Company:
ROM Required:	Address:
Maximum Subroutine Nesting Level:	City:
Assembler/Compiler Used:	State:

insite T.M.

PROGRAM CERTIFICATION AND REVIEW FORM

Please check all statements made by the submitting author before noting program discrepancies. Any comments relating to program improvement are welcome; however, program revisions or rewrites must be sent in as original submissions.

PROGRAM NAME:			REFERENCE NUMBER:		
Does the program work?	Yes ☐	No ☐	Was modification necessary?	Yes ☐	No ☐
Were the author's comments accurate?	Yes ☐	No ☐	Were usage instructions adequate?	Yes ☐	No ☐
Was the documentation sufficient?	Yes ☐	No ☐	Should the program be revised?	Yes ☐	No ☐

COMMENTS: (Please elaborate on deficiencies noted above and include any necessary modifications)

(Use additional sheets if necessary)

REVIEWED BY:	COMPANY:

insite T.M.

PROGRAM CERTIFICATION AND REVIEW FORM

Please check all statements made by the submitting author before noting program discrepancies. Any comments relating to program improvement are welcome; however, program revisions or rewrites must be sent in as original submissions.

PROGRAM NAME:			REFERENCE NUMBER:		
Does the program work?	Yes ☐	No ☐	Was modification necessary?	Yes ☐	No ☐
Were the author's comments accurate?	Yes ☐	No ☐	Were usage instructions adequate?	Yes ☐	No ☐
Was the documentation sufficient?	Yes ☐	No ☐	Should the program be revised?	Yes ☐	No ☐

COMMENTS: (Please elaborate on deficiencies noted above and include any necessary modifications)

(Use additional sheets if necessary)

REVIEWED BY:	COMPANY:

SECTION 4

OPERATING, TESTING AND DEBUGGING PROGRAMS

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AD1

☐ 4004 ☐ 4040 ☒ 8008 ☐ 8080 ☐ 3000

(use additional sheets if necessary)

Program Title

SAVE/RESTORE CPU STATES ON AN INTERRUPT

Function

Saves the CPU registers and flags in memory at the start of interrupt processing and restores the CPU registers and flags after the interrupt has been processed.

Required Hardware

A LIFO register (hardware external push down stack) to allow saving registers and flags without using L and H.

In the present program, SVAD (saveguard address) is the output address of that LIFO register and RSAD (restitution address) is the input.

Required Software

None

Input Parameters

None

Output Results

None

Registers Modified:	Assembler/Compiler Used:
C	MAC8
RAM Required:	**Programmer:**
6	Robert Arouete
ROM Required:	**Company:**
60_{10} bytes	A 2 M
Maximum Subroutine Nesting Level:	**Address:** 78 170 La Celle St. Cloud, FRANCE
0	

```
                ;REF. NO. AD1
                ;PROGRAM TITLE SAVE/RESTORE CPU STATE ON INTERRUPT
                ;
                ;
                ;
                ;

0100                    ORG 100H
                SV:
0100 D310               OUT SVAD
0102 78                 MOV A,B
0103 D310               OUT SVAD
0105 79                 MOV A,C
0106 D310               OUT SVAD
0108 7A                 MOV A,D
0109 D310               OUT SVAD
010B 7B                 MOV A,E
010C D310               OUT SVAD
010E 7C                 MOV A,H
010F D310               OUT SVAD
0111 7D                 MOV A,L
0112 D310               OUT SVAD
                SVF:
[illegible] 3E00        MVI A,0
[illegible] 47          MOV B,A
0117 CA2901             JZ S1I          ;DECODE Z FLAG
011A 3E08               MVI A,8
011C F22501             JP S2I          ;DECODE S FLAG
011F 0640               MVI B,64
0121 EA2901             JPE S1I         ;DECODE P FLAG WHEN S IS FALSE
0124 04                 INR B
                S2I:
0125 E22901             JPO S1I         ;DECODE P FLAG WHEN S IS TRUE
0128 04                 INR B
                S1I:
0129 1F                 RAR             ;DECODE CARRY FLAG
012A B0                 ORA B           ;GROUP ALL FLAGS IN ONE WORD
012B D310               OUT SVAD        ;SAVE FLAGS
012D C9                 RET
0010            SVAD    EQU 16
                RSP:
012E AF                 XRA A
012F DB04               IN RSAD
0131 17                 RAL             ;RESTORE CARRY FLAG
0132 47                 MOV B,A
0133 04                 INR B
0134 05                 DCR B           ;RESTORE OTHER FLAGS
0135 DB04               IN RSAD         ;RESTORE REGISTERS AND ACCUMULATOR
0137 6F                 MOV L,A
```

```
0138 DB04               IN RSAD
013A 67                 MOV H,A
013B DB04               IN RSAD
013D 5F                 MOV E,A
013E DB04               IN RSAD
0140 57                 MOV D,A
0141 DB04               IN RSAD
0143 4F                 MOV C,A
0144 DB04               IN RSAD
0146 47                 MOV B,A
0147 DB04               IN RSAD
0149 C9                 RET
0004            RSAD    EQU 4
0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

intel®

Ref. No. AA1

☐ 4004 ☒ 8008 ☐ 8080

(use additional sheets if necessary)

Program Title

RAM TEST PROGRAM

Function

Performs write and read of all zeros and ones, checkerboard test and unique address test. The RAM to be tested is successively initialized to a value and then tested. The values are zero, all ones, alternate bit word (Ø252) and its complement (Ø125). Next, increasing consecutive values are stored and then tested.

Required Hardware

Teletypewritter

Required Software

Requires teletypewritter I/O subroutines. (Other subroutines included.)

Input Parameters

After an "A" is printed, the operator types the initial and final RAM addresses separated by a comma and ended with a carriage return. All numbers are in octal.

Output Results

For each error found, the address, contents and the value which should have been read are printed with the heading. No errors are signified by printing "A" for the next test.

Registers Modified: All	**Maximum Subroutine Nesting Level:** 4 or 5
RAM Required: 20 bytes	**Assembler/Compiler Used:** MAC8
ROM Required: 429 bytes	**Programmer:** S Kung/D Wallace/L Brenden Varian Instrument Division
	Company: 2700 Mitchell Drive Walnut Creek, Calif.

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

intel®

Ref. No. AB1

☐ 4004 ☐ 4040 ☒ 8008 ☐ 8080 ☐ 3000

(use additional sheets if necessary)

Program Title

TTY Binary Load Routine

Function

This program will load memory via TTY which are formatted into blocks of 256 or less binary bytes and tests each block against a checksum frame for any read errors. The tape format requires a rubout to indicate start of block followed by the starting page address, the starting byte address, the word count, up to 256 bytes of data and a checksum in that order. A block of data may overlap pages but may not exceed 256 bytes in length. The last block of data should be followed by two consecutive rubouts to indicate end of data. The program will then branch to page 000 byte 000.

Required Hardware

REQUIRED HARDWARE: ASR-33 teletype, teletype interface to 8008

Required Software

Intel MCS-8 hardware assembler page 000 or TTY input and output routines

Input Parameters

None

Output Results

Program will be loaded into allocated memory locations. A checksum error will cause a program halt. Interrupting with a No-Op will restart the program.

Registers Modified: A,B,C,D,E,H,L	**Assembler/Compiler Used:** MAC8
RAM Required: 73 bytes	**Programmer:** David Crellen
ROM Required: MCS-8 hardware assembler, pp. 000	**Company:** Continental Design
Maximum Subroutine Nesting Level: 2	**Address:** 7723 Convoy Court San Diego, Ca. 92111

DESCRIPTION OF TEST DATA, DEFINITION OF TERMS, ADDITIONAL INFORMATION FOR THE TTY BINARY LOADER AND TTY BINARY DUMP ROUTINES.

It is recommended that the TTY Binary Dump routine be assembled first. A known section of memory, preferably ROM should be dumped using the Binary dump routine. By manually checking the first few bytes with an octal dump of the same area the user can determine whether or not he's on the right track.

The complete program checkout procedure would be as follows:

Load a page of octal data using the octal loader.
Dump the page in binary.
Load the page back in using the binary loader.
Dump the page using the octal dump program.

The tape produced by the octal loader in the final step should compare exactly with the tape loaded in initially.

SOME TERMS DEFINED; AND CONVENTIONS USED:

The Checksum produced by the dump program and checked by the loader is a modulo-256_{10} sum of all data in the block. Assuming a valid punch operation, a faulty checksum indicates a read error.

The byte count is the number of bytes to be loaded into memory. At the expiration of the byte count, the checksum is examined.

The dump produces, and the loader requires, an eight bit rubout preceding the binary address of the data. Thus, the format is:

```
ooooo.ooo     rubout
ooooo.ooo     starting address, page
ooooo.ooo     starting address, byte
ooooo.ooo     byte count
.
.
.
'byte count'  frames
.
.
.
ooooo.ooo     checksum

ooooo.ooo     rubout
ooooo.ooo     rubout   ...stop code
```

```
                    ;REF.  NO AB1
                    ;PROGRAM TITLE TTY BINARY LOAD ROUTINE
                    ;
                    ;
                    ;
                    ;
                     ;TTY BINARY LOAD ROUTINE
                     BEGIN:
0000 CD3500                 CALL    READ;           READ A FRAME
0003 FEFF                   CPI     377Q;           START OF BLOCK?
0005 C20000                 JNZ     BEGIN;          NO.
0008 CD3500                 CALL    READ
000B FEFF                   CPI     377Q;           2 RUBOUTS = STOP CODE
000D CA4700                 JZ      DONE
0010 67                     MOV     H,A;            LOAD STARTING ADDRESS PAGE
0011 CD3500                 CALL    READ
0014 6F                     MOV     L,A;            LOAD STARTING ADDRESS BYTE
0015 CD3500                 CALL    READ
0018 5F                     MOV     E,A;            LOAD BYTE COUNT
0019 1600                   MVI     D,000;          INITIALIZE CHECKSUM
                    LOAD:
001B CD3500                 CALL    READ
001E 77                     MOV     M,A;            STORE BYTE
001F 82                     ADD     D;              ADD TO CHECKSUM
0020 57                     MOV     D,A;            UPDATE CHECKSUM
0021 2C                     INR     L;              NEXT BYTE
0022 C22600                 JNZ     SKIP;           NO PAGE LAP
0025 24                     INR     H
                    SKIP:
0026 1D                     DCR     E;              DECREMENT BYTE COUNT
0027 C21B00                 JNZ     LOAD;           LOOP UNTIL BYTE COUNT = 0
002A CD3500                 CALL    READ;           READ CHECKSUM FROM TAPE
002D 92                     SUB     D;              COMPARE CALCULATED CHECKSUM TO
                    ;                               READ CHECKSUM
002E CA0000                 JZ      BEGIN;          READ NEXT BLOCK
                    STOP:
0031 76                     HLT;  000               CHECKSUM
                    ;                               ERROR STOP.  PROGRAM CAN BE RESTA
                    ;                               WITH A "NOP" INTERRUPT.
0032 C30000                 JMP     BEGIN
                    READ:
0035 45                     MOV     B,L;            SAVE REGISTERS
0036 4C                     MOV     C,H
0037 214800                 LXI     H,SAVER
003A 70                     MOV     M,B
003B 2C                     INR     L
003C 71                     MOV     M,C
003D 2C                     INR     L
003E 72                     MOV     M,D
```

```
003F AF                 XRA     A;      ACC=0; FLAGS SET
0040 CF                 RST     TYINP;  DATA INPUT FROM TTY
0041 56                 MOV     D,M;    NOT SEND TO PRINTER
0042 2D                 DCR     L;      RESTORE REGISTERS
0043 4E                 MOV     C,M
0044 2D                 DCR     L
0045 6E                 MOV     L,M
0046 61                 MOV     H,C
                DONE:
0047 C9                 RET
                SAVER:
0048                    DS 3
0001            TYINP   EQU 1
0000                    END
```

Ref.# AB2

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☒8008 ☐8080 ☐8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

TTY BINARY DUMP ROUTINE

Function

This program will punch the contents of memory via TTY which are formatted into blocks of 256 or less binary bytes with checksum. The tape format begins with a rubout to indicate start of block followed by the starting page address, the starting byte address, the word count, up to 256 bytes of data and a checksum in that order. The start and end memory locations are entered from the TTY keyboard.

Required Hardware

ASR-33 teletype, teletype interface to 8008

Required Software

Intel MCS-8 Hardware assembler page 000 or TTY input and output routines and TTY octal to binary converting routine (BILDE).

Input Parameters

On TTY keyboard enter D followed by colon then ending page (PPP), ending byte (BBB), starting page (ppp), starting byte (bbb).

D: PPP: BBB: ppp: bbb:

Output Results

Program will punch binary tape from designated memory locations and calculate and punch a checksum. Tape feed frames are punched between blocks and a stop code (two consecutive rubouts) are punched at end of dump. The program will end by branching to page 000 byte 000.

Registers Modified: A, B, C, D, E, H, L	**Programmer:** David Crellen
RAM Required:	**Company:** Continental Design
ROM Required: MCS-8 Hardware assembler pg. 000	**Address:** 7723 Convoy St.
Maximum Subroutine Nesting Level: 3	**City:** San Diego
Assembler/Compiler Used: MAC8	**State:** CA 92111

```
                ;REF. NO. AB2
                ;PROGRAM TITLE TTY BINARY DUMP
                ;
                ;
                ;
                        ;TTY BINARY DUMP ROUTINE
                BEGIN:
0000 AF                 XRA     A
0001 FF                 RST     TTYIN;          INPUT TTY-SEND TO PRINTER
0002 F680               ORI     200Q;           ADD PARITY
0004 FEC4               CPI     304Q;           D?
0006 C20000             JNZ     BEGIN;          NO.
0009 AF                 XRA     A
000A FF                 RST     TTYIN
000B F680               ORI     200Q
000D FEBA               CPI     272Q;           COLON?
000F C20000             JNZ     BEGIN;          NO.
0012 218E00             LXI     H,ENDER;        POINT TO ENDING VALUE
0015 CD9F00             CALL    BILDE;          ACCEPT 3-DIGIT OCTAL FROM TTY
                ;                               CONVERT TO BINARY REPRESENTATION
 0018 73                MOV     M,E;            STORE ENDING PAGE
0019 2C                 INR     L;              POINT TO BYTE END VALUE
001A CD9F00             CALL    BILDE;          ACCEPT OCTAL FROM TTY
001D 73                 MOV     M,E;            STORE ENDING BYTE
001E CD9F00             CALL    BILDE;          ACCEPT OCTAL FROM TTY
0021 63                 MOV     H,E;            STORE START PAGE
0022 CD9F00             CALL    BILDE;          ACCEPT OCTAL FROM TTY
0025 6B                 MOV     L,E;            STORE START BYTE
                PADR:
0026 CD6A00             CALL    FEED;           GENERATE LEADER FOR PUNCH
0029 CD7600             CALL    BYTEC;          CALCULATE FIRST-BLOCK BYTE COUNT
002C 7C                 MOV     A,H;            MOVE PAGE ADDRESS TO ACC FOR
                ;                               BINARY PUNCH.  NOTE THAT THE
                ;                               REQUIRED LEADING RUBOUT HAS BEEN
                ;                               PUNCHED BY THE FEED ROUTINE PRE-
                ;                               CEEDING EACH BLOCK
002D F7                 RST     TTYOT;          PUNCH STARTIN ADDRESS
002E 7D                 MOV     A,L;            MOVE BYTE ADDRESS TO ACC
002F F7                 RST     TTYOT;          PUNCH STARTING ADDRESS -- BYTE
0030 AF                 XRA     A;              COMPUTE BYTE COUNT
0031 93                 SUB     E
0032 F7                 RST     TTYOT;          BYTE COUNT IN ACC--PUNCH ONTO TA
0033 1E00               MVI     E,000;          INITIALIZE CHECKSUM
                PTEXT:
0035 7E                 MOV     A,M;            FETCH DATUM FROM MEMORY
 0036 83                ADD     E;              COMPUTE CHECKSUM
0037 5F                 MOV     E,A;            UPDATE CHECKSUM
0038 7E                 MOV     A,M;            CHECKSUM CALCULATION DESTROYED D
                ;                               IN ACC.  IS FETCHED FROM MEMORY A
```

```
                        PUNCH:
0039 F7                         RST     TTYOT;          PUNCH DATUM
003A 7C                         MOV     A,H;            SAVE PAGE VALUE
003B 4D                         MOV     C,L;            SAVE BYTE VALUE
003C 218E00                     LXI     H,ENDER;        POINT TO ENDING VALUE
003F BE                         CMP     M;              LAST PAGE?
0040 C24B00                     JNZ     P2;             NO.
0043 47                         MOV     B,A
0044 79                         MOV     A,C;            MOVE BYTE ADDRESS TO ACC FOR COM
0045 2C                         INR     L;              POINT TO LAST BYTE
0046 BE                         CMP     M;              LAST BYTE?
0047 78                         MOV     A,B
0048 CA5C00                     JZ      PEND;           YES--GO TO END OF JOB
                        P2:
004B 0C                         INR     C;              NEXT BYTE
004C C25700                     JNZ     P3
004F 69                         MOV     L,C;            RESTORE H&L REGISTERS
0050 67                         MOV     H,A;            RESTORE H&L REGISTERS
0051 24                         INR     H
0052 7B                         MOV     A,E
0053 F7                         RST     TTYOT;          PUNCH CHECKSUM
 354 C32600                     JMP     PADR;           GO PUNCH NEXT BLOCK
                        P3:
0057 69                         MOV     L,C;            RESTORE H&L REGISTERS
0058 67                         MOV     H,A
0059 C33500                     JMP     PTEXT
                        PEND:
005C 7B                         MOV     A,E;            PUNCH CHECKSUM
005D F7                         RST     TTYOT
005E CD6A00                     CALL    FEED;           GENERATE TRAILER AND 1 OF
                                                        2 NEEDED RUBOUTS FOR STOP CODE
                        ;
0061 3EFF                       MVI     A,377Q
0063 F7                         RST     TTYOT;          PUNCH RUBOUT
0064 CD6A00                     CALL    FEED;           MORE TRAILER
0067 C30000                     JMP     DONE
                        ;THE VALUE OF DONE IS EQUATED TO
                        ;000 000 IN THIS ASSEMBLY.  THIS ADDRESS
                        ;IS THE DESIRED BRANCH AFTER THE DUMP.
                        ;IS COMPLETED.  A VALUE OF 000 000 RETURNS
                        ;CONTROL TO THE TTY OPERATING SYSTEM SO
                        ;FURTHER COMMANDS MAY BE MADE.  IF THE
                        ;DUMP PROGRAM IS TO BE MADE A CALLED
                        ;SUBROUTINE, A "RET" INSTRUCTION WOULD
                        ;BE APPROPRIATE.  THE PROGRAM COULD ALSO
                        ;BE MADE TO HALT HERE, IF DESIRED.
                        FEED:
 36A 1E28                       MVI     E,040;          INTER-RECORD GAP OF 32 FRAMES
                        F:
006C AF                         XRA     A;              INSURE TAPEFEED IS PUNCHED
006D F7                         RST     TTYOT;          PUNCH TAPEFEED
```

```
006E 1D                 DCR     E;              REDUCE "FRAMES REMAINING" COUNT
006F C26C00             JNZ     F;              LOOP UNTIL DONE
0072 3EFF               MVI     A,377Q;         LOAD LEADING RUBOUT FOR BLOCK ST
0074 F7                 RST     TTYOT;          PUNCH START OF BLOCK
0075 C9                 RET     ;               RE-ENTER PROGRAM
                BYTEC:
0076 44                 MOV     B,H;            SAVE H
0077 4D                 MOV     C,L;            SAVE L
0078 7C                 MOV     A,H;            GET INITIAL H TO ACC
0079 218E00             LXI     H,ENDER;        POINT TO FINAL H
007C BE                 CMP     M;              INITIAL PAGE = FINAL PAGE?
007D C28600             JNZ     NEQ;            NO.
0080 2C                 INR     L;              POINT TO FINAL L
0081 7E                 MOV     A,M;            GET FINAL L
0082 91                 SUB     C;              ACC=FINAL - INITIAL
0083 C38800             JMP     SKIP
                NEQ:
0086 2C                 INR     L;              POINT TO FINAL L
0087 7E                 MOV     A,M;            GET FINAL L
                SKIP:
0088 EEFF               XRI     377Q;           1'S COMPLEMENT
 38A 5F                 MOV     E,A;            STORE BYTE COUNT IN E
008B 60                 MOV     H,B;            RESTORE H
008C 69                 MOV     L,C;            RESTORE L
008D C9                 RET
                ENDER:
008E                    DS 2                    ;THE FINAL ADDRESS IS STORED HER
0000            DONE    EQU     0;              THIS IS THE BRANCH ADDRESS ON PR
                ;                               COMPLETION
0006            TTYOT   EQU     6;              TTY OUTPUT ROUTINE
0007            TTYIN   EQU     7;              TTY INPUT ROUTINE -- ECHOES BACK
                ;                               TO PRINTER.  EQUATE TO 000010 IF
                ;                               DATA ENTRY IS PREFERRED
009F            BILDE   EQU     0002370;                ACCEPTS 3-DIGIT OCTAL TE
                ;                               COLON AND CONVERTS TO BINARY WIT
                ;                               RESULT IN "E"
0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB3

☐ 4004 ☐ 4040 ☒ 8008 ☐ 8080 ☐ 3000 (use additional sheets if necessary)

Program Title

MEMORY DUMP

Function

Lists memory in octal: start & stop point user definable

Required Hardware

TTY -- standard Intel setup

Required Software

TTY routines
Carriage return & linefeed routines

Input Parameters

None

Output Results

Lists memory on teletype

Registers Modified:	Assembler/Compiler Used:
All	MAC8
RAM Required: Last 7 words of 013 1 RAM or ROM for pgm storage	**Programmer:** LeRoy J. Kniskern
ROM Required:	**Company:** Poly-Scientific Div. Litton Systems, Inc.
Maximum Subroutine Nesting Level:	**Address:** 1213 N. Main St. Blacksburg, Va. 24060

```
                        ;REF.  NO.  AB3
                       ;PROGRAM TITLE MEMORY DUMP
                       ;
                       ;
                       ;
                       ;
                       ;
                       ;
                       ;DUMPS SELECTED MEMORY IN OCTAL
                        ;
                        ;USER INSTRUCTIONS
                        ;   CALL THIS SUBROUTINE
                        ;   TTY WILL TYPE F
                        ;   TYPE FIRST ADDRESS TO BE DUMPED
                        ;     IN FORMAT HHH:LLL:
                        ;     EXAMPLE 010:356:
                        ;     ADDRESS IN OCTAL, 3 DIGITS IMMEDIATELY
                        ;     PRECEEDING : ARE INTERPERTED AS ADDRESS
                        ;   TTY WILL TYPE L
                        ;   TYPE LAST ADDRESS TO BE DUMPED IN SAME FORMAT
                        ;   ADDRESSES NEED NOT BE ON SAME PAGE
                        ;
                        ;   EXTERNAL SUBROUTINES REQUIRED:
                        ;     TTYIN-INTEL TELETYPE INPUT ROUTINE
                        ;     TOUT - INTEL TELTYPE OUTPUT ROUTINE
                        ;     CRLF -CARRIAGE RET & LINE FEED ROUTINE
                        ;
                        ;   REGISTERS USED - ALL
                        ;
                        ;   RAM USED - LOCATIONS 371 THRU 377 IN 013
                        ;
                        ;
                        ;   OUTPUT FORMAT -ADDRESS LISTED EVERY 8 LOCATIONS
                        ;     CONTENTS OF M IN OCTAL
                        ;
                       BEGIN:
0000 260B                       MVI     H,0130
0002 CD2800                     CALL    CRLF;   CARRIAGE RETURN & LINE FEED
0005 0646                       MVI     B,'F'
0007 CD8500                     CALL    TOUT
000A CD2800                     CALL    CRLF
000D CDA600                     CALL    TAKE3;  GET FIRST 'H'
0010 2EFC                       MVI     L,374Q
0012 73                         MOV     M,E
0013 CDA600                     CALL    TAKE3;  LET FIRST 'L'
0016 2EFB                       MVI     L,373Q
 318 73                         MOV     M,E
0019 CD2800                     CALL    CRLF
001C 064C                       MVI     B,'L'
001E CD8500                     CALL    TOUT
```

```
0021 CD2800             CALL    CRLF
0024 CDA600             CALL    TAKE3;  GET LAST 'H'
0027 2EFA               MVI     L,372Q
0029 73                 MOV     M,E
002A CDA600             CALL    TAKE3;  GET LAST 'L'
002D 2EF9               MVI     L,371Q
002F 73                 MOV     M,E
0030 2C                 INR     L
0031 2C                 INR     L
0032 46                 MOV     B,M
0033 2C                 INR     L
0034 4E                 MOV     C,M
0035 68                 MOV     L,B
0036 61                 MOV     H,C
                PRT:
0037 CD5800             CALL    PRADR
                PRT1:
003A 5E                 MOV     E,M
003B CD8200             CALL    PROCT
003E CD6E00             CALL    ADCK
0041 CA5400             JZ      DONE
 344 2C                 INR     L
0045 C24900             JNZ     PRT2
0048 24                 INR     H
                PRT2:
0049 7D                 MOV     A,L
004A E607               ANI     007
004C FE00               CPI     000
004E CC5800             CZ      PRADR
0051 C33A00             JMP     PRT1
                DONE:
0054 CD2800             CALL    CRLF
0057 C9                 RET     ;               END OF DUMP (COULD BE JUMP INSTR
                PRADR:
0058 CD2800             CALL    CRLF
005B 5C                 MOV     E,H
005C CD8200             CALL    PROCT
005F 5D                 MOV     E,L
0060 CD8200             CALL    PROCT
0063 062D               MVI     B,'-'
0065 CD8500             CALL    TOUT
0068 063B               MVI     B,';';          SPACE
006A CD8500             CALL    TOUT
006D C9                 RET
                ADCK:
006E 45                 MOV     B,L
 36F 4C                 MOV     C,H
0070 21F916             LXI     H,0133710
0073 7E                 MOV     A,M
0074 B8                 CMP     B
```

```
0075 C27F00             JNZ     NOTIT
0078 2C                 INR     L
0079 7E                 MOV     A,M
007A B9                 CMP     C
007B C27F00             JNZ     NOTIT
007E C9                 RET
                NOTIT:
007F 68                 MOV     L,B
0080 61                 MOV     H,C
0081 C9                 RET
                PROCT:
0082 7B                 MOV     A,E
0083 E6C0               ANI     300Q
0085 07                 RLC
0086 07                 RLC
0087 CD9F00             CALL    OCT1
008A 7B                 MOV     A,E
008B E638               ANI     070Q
008D 0F                 RRC
008E 0F                 RRC
008F 0F                 RRC
 390 CD9F00             CALL    OCT1
0093 7B                 MOV     A,E
0094 E607               ANI     007
0096 CD9F00             CALL    OCT1
0099 063B               MVI     B,';';  SPACE
009B CD8500             CALL    TOUT
009E C9                 RET
                OCT1:
009F C630               ADI     '0';    060 OCTAL
00A1 47                 MOV     B,A
00A2 CD8500             CALL    TOUT
00A5 C9                 RET
                TAKE3:
00A6 CD3900             CALL    TTYIN
00A9 78                 MOV     A,B
00AA FE3A               CPI     ':'
00AC CABB00             JZ      IN3
00AF 2EFF               MVI     L,377Q
00B1 56                 MOV     D,M
00B2 70                 MOV     M,B
00B3 2D                 DCR     L
00B4 5E                 MOV     E,M
00B5 72                 MOV     M,D
00B6 2D                 DCR     L
00B7 73                 MOV     M,E
 3B8 C3A600             JMP     TAKE3
                IN3:
00BB CDCA00             CALL    CONOCT
00BE D0                 RNC     ;               NO ERROR, BINARY NOW IN -E-
```

```
00BF 063F               MVI     B,'?'
00C1 CD8500             CALL    TOUT
00C4 CD2800             CALL    CRLF
00C7 C3A600             JMP     TAKE3
                CONOCT:
00CA 2EFF               MVI     L,377Q
00CC CDF200             CALL    CON1
00CF D8                 RC
00D0 5F                 MOV     E,A
00D1 2D                 DCR     L
00D2 CDF200             CALL    CON1
00D5 D8                 RC
00D6 07                 RLC
00D7 07                 RLC
00D8 07                 RLC
00D9 83                 ADD     E
00DA 5F                 MOV     E,A
00DB 2D                 DCR     L
00DC CDF200             CALL    CON1
00DF D8                 RC
00E0 47                 MOV     B,A
 3E1 D604               SUI     004
00E3 D2EE00             JNC     NOTOK
00E6 78                 MOV     A,B
00E7 0F                 RRC
00E8 0F                 RRC
00E9 83                 ADD     E
00EA 5F                 MOV     E,A
00EB AF                 XRA     A
00EC 1F                 RAR
00ED C9                 RET
                NOTOK:
00EE 3E01               MVI     A,001
00F0 1F                 RAR
00F1 C9                 RET
                CON1:
00F2 7E                 MOV     A,M
00F3 C6C8               ADI     310Q
00F5 D8                 RC
00F6 D6F8               SUI     370Q
00F8 C9                 RET
0028            CRLF    EQU     50Q;    CHANGE AS REQUIRED
0085            TOUT    EQU     205Q;   CHANGE AS REQUIRED
0039            TTYIN   EQU     71Q;    CHANGE AS REQUIRED
0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB4

☐ 4004 ☒ 8008 ☐ 8080

(use additional sheets if necessary)

Program Title

PROM PROGRAMMER FOR INTELLEC 8

Function

Changes programmer from fixed timing to PROM dependent timing. Programs 50% more than minimum required, ensuring permanency.

Required Hardware

Intellec 8 with PROM programming board and system monitor PROMs.

Required Software

Works with Intellec 8 system monitor version 1.0 and 1.1.

Input Parameters

Interfaces with user exactly as original version.

Output Results

Noticeably faster programming with no requirement for two (2) passes to ensure permanency. Locations that cannot be programmed result in return to system with a printout of the location.

Registers Modified: Register D used in addition to others used in original version.	**Maximum Subroutine Nesting Level:** N/A
RAM Required: None additional	**Assembler/Compiler Used:** Intellec 8/Mod 8 Macro Assembler, Ver. 1.0
ROM Required: None additional	**Programmer:** William Haskett Northeast Electronics
	Company: Box 649 Concord, N.H. 03301

```
                        ;REF. NO. AB4
                        ;PROGRAM PROM PROGRAMMER FOR INTELLEC 8
                        ;
                        ;
                        ;
                        ;
                        ;
3CCB                    INCAD   SET     3CCBH           ;
3BDE                    START   SET     3BDEH           ;
3EF2                    HEXAD   SET     3EF2H           ;MONITOR SUBROUTINES
3C75                    LONG    SET     3C75H           ;
3C72                    SHORT   SET     3C72H           ;
3B9A                            ORG     3B9AH
3B9A 1608               NEW:    MVI     D,08H           ;SET MINIMUM OVERCHARGE
3B9C 79                         MOV     A,C             ;FET ADDRESS TO PROGRAM
3B9D EEFF                       XRI     0FFH            ;COMPLIMENT IT
3B9F D30A                       OUT     10              ;SEND TO PROM
3BA1 DB02               OLD:    IN      2               ;READ DATA IN PROM
3BA3 EEFF                       XRI     0FFH            ;
3BA5 BE                         CMP     M               ;COMP WITH DESIRED DATA
3BA6 CAB23B                     JZ      DOWN            ;IF OK, GOTO OVERCHARGE
3BA9 14                         INR     D               ;INCREMENT PASS COUNT
3BAA 3E70                       MVI     A,70H           ;GET MAXIMUM PASSES
3BAC BA                         CMP     D               ;COMPARE WITH COUNT
3BAD CADA3B                     JZ      ERROR           ;CAN'T PROGRAM, GIVE UP
3BB0 14                         INR     D
3BB1 14                         INR     D               ;ADD TWO TO PASS COUNT
3BB2 15                 DOWN:   DCR     D               ;SUB ONE FROM COUNT
3BB3 CACC3B                     JZ      DONE            ;FINISHED IF ZERO
3BB6 3EFF                       MVI     A,0FFH
3BB8 AE                         XRA     M               ;GET DATA TO PROGRAM
3BB9 D30B                       OUT     11              ;SEND IT TO PROM
3BBB 3EB7                       MVI     A,0B7H
3BBD D309                       OUT     9               ;ENABLE PROG PULSES
3BBF CD753C                     CALL    LONG            ;WAIT THREE PULSES
3BC2 3E37                       MVI     A,37H
3BC4 D309                       OUT     9               ;STOP PULSES
3BC6 CD723C                     CALL    SHORT           ;COOL OFF
3BC9 C3A13B                     JMP     OLD             ;DO IT AGAIN
3BCC 0C                 DONE:   INR     C               ;STEP NEXT PROM ADD
3BCD CADE3B                     JZ      START           ;QUIT IF OVERFLOW
3BD0 CDCB3C                     CALL    INCAD           ;STEP MEMORY ADDRESS
3BD3 1D                         DCR     E               ;DEC COUNT OF POS TO PROG
3BD4 C29A3B                     JNZ     NEW             ;START PROG NEW LOCATION
3BD7 C3DE3B                     JMP     START           ;RETURN TO SYSTEM
3BDA 79                 ERROR:  MOV     A,C             ;GET ADD WHERE FAILED
3BDB CDF23E                     CALL    HEXAD           ;PRINT IT
3BDE 00                         NOP
0000                            END                     ;ENDS AT START OF MONITOR
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AC1

☐ 4004 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

8080 I/O SYSTEM STATUS DISPLAY

Function

Display current I/O assignment information when invoked by the INTELLEC8/ MOD 80 MONITOR (Version 1.0)

Required Hardware

INTELLEC 8/MOD 80, 1 1702A PROM

Required Software

INTELLEC 8/MOD 80 MONITOR, Version 1.0

Input Parameters

Calls IOCHK and uses the returned value for driving the display

Output Results

SYSTEM STATUS

CONSOLE	TTY
READER	PTR
PUNCH	PTP
LIST	TTY

Registers Modified: N/A	**Maximum Subroutine Nesting Level:** 3 bytes
RAM Required: NONE	**Assembler/Compiler Used:** 8080 Resident Macro Assembler
ROM Required: 1 1702A	**Programmer:** K. Burgett
	Company:

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/4040 ☐ 8008 ☒ 8080 ☐ 8048 ☐ 8085 ☐ Other _____ (use additional sheets if necessary)

Program Title

READ AND INTERRUPT MODIFICATIONS TO THE 8080 SYSTEM MONITOR

Function

These modifications allow printing of headings or operator instructions at the beginning of a Read operation (from the front of the Hex Load tape). They also allow console control of the interrupt enable F-F. This prevents interrupts until the service subroutines are loaded.

Required Hardware

Intellec 8-80 (or equal)
PROM Programmer

Required Software

8080 Monitor, Version 2.0

Input Parameters

Monitor Commands:
"R"
"IE"
"ID"

Output Results

The Monitor Commands cause the following action:

"R" Read Hex Tape and print heading (all characters before the first ":") on the console.

"IE" Enable the 8080 interrupt F-F.

"ID" Disable the 8080 interrupt F-F.

Registers Modified: Not Important	Programmer: C. Vincent Phillips
RAM Required: None	Company: Alkon Corporation
ROM Required: 48H (Or 1 1702A)	Address: 5329 N. High St.
Maximum Subroutine Nesting Level: 0 (Not important)	City: Columbus
Assembler/Compiler Used: 8080 Macro Assembler	State: Ohio 43214

```
                        ;REF. NO. AD2
                        ;PROGRAM TITLE READ AND INTERRUPT MODIFICATIONS
                        ;
                        ;
                        ;
                        ;LINKAGES
3C20                    LER     EQU     3C20H           ;ERROR ROUTINE
3D59                    EXPR    EQU     3D59H           ;GET EXPRESSION
3CAD                    CRLF    EQU     3CADH           ;CARRIAGE RETURN,LINE FEED
3EEE                    RIX     EQU     3EEEH           ;READ,CHECK AND MASK PARI
3F6D                    TI      EQU     3F6DH           ;INPUT FROM CONSOLE AND E
386A                    START   EQU     386AH           ;MAIN CONTROL LOOP
3C32                    CO      EQU     3C32H           ;CONSOLE OUTPUT
                        ;THIS CHIP CONTAINS MONITOR COMMAND OVERFLOW
                        ;THE UNUSED BRANCHES IN THE COMMAND BRANCH
                        ;TABLE HAVE BEEN MOVED TO THIS CHIP TO EASE
                        ;FUTURE MODIFICAIONS.
                        ;THE FOLLOWING MODIFICAIONS MUST BE MADE TO
                        ;CHIP 3800 (COMMAND BRANCH TABLE)
                        ;       LOCATION        OLD     NEW
                        ;       38A3    2 203C  0036
                        ;       38A5            203C    0336
                        ;       38A7            203C    0636
                        ;       38AF            203C    0936
                        ;       38B3            203C    0C36
                        ;       38B5            DF3A    0F36
                        ;       38BB            203C    1236
                        ;       38BD            203C    1536
                        ;
3600                            ORG             3600H   ;CHIP 3600H
                        ;NEW COMMAND BRANCH TABLE
                        ;
3600 C33636                     JMP             INTER   ;I ENTRY POINT
3603 C3203C                     JMP             LER     ;J ENTRY POINT
3606 C3203C                     JMP             LER     ;K ENTRY POINT
3609 C3203C                     JMP             LER     ;O ENTRY POINT
360C C3203C                     JMP             LER     ;Q ENTRY POINT
360F C31836                     JMP             READ    ;R ENTRY POINT
3612 C3203C                     JMP             LER     ;U ENTRY POINT
3615 C3203C                     JMP             LER     ;V ENTRY POINT
                        ;
                        ;READ ROUTINE
                          ;THIS SUBROUTINE CAUSES A HEADING TO BE
                          ;PRINTED ON THE CONSOLE FROM THE INPUT TAPE
                          ;BEFORE THE BALANCE OF THE TAPE IS READ IN
                          ;THE NORMAL MANNER.  ALL CHARACTERS BEFORE
                          ;THE FIRST : EXCEPT NULLS ARE PRINTED.  THIS
                          ;IS USEFUL WHEN LOADING FROM MAG TAPE OR A
                          ;DISK.
```

```
                        ;
                READ:
3618 0D                         DCR     C               ;GET ONE ADDRESS
3619 CD593D                     CALL    EXPR
361C CDAD3C                     CALL    CRLF            ;SPACE UP
361F CDAD3C                     CALL    CRLF
                RA:
3622 E1                         POP     H               ;GET BIAS ADDRESS
3623 E5                         PUSH    H
3624 CDEE3E                     CALL    RIX             ;READ CHARACTER
3627 CA2236                     JZ      RA              ;SKIP IF NULL
362A 4F                         MOV     C,A             ;
362B CD323C                     CALL    CO              ;TYPE ON CONSOLE
362E D63A                       SUI     ':'             ;LOOKFOR RECORD MARK
3630 C22236                     JNZ     RA
3633 C3EE3A                     JMP     3AEEH           ;JUMP TO READ ROUTINE
                ;
                ;INTERRUPT CONTROL ROUTINE
                  ;IE ENABLE INTERRUPT
                  ;ID DISABLE INTERRUPT
                  ;THIS SUBROUTINE ALLOWS THE CONTROL OF
                  ;THE INTERRUPT ENABLE FROM THE CONSOLE.
                  ;THIS IS USEFUL TO PREVENT INTERRUPTS
                  ;UNTIL THE INTERRUPT SERVICE SUBROUTINES
                  ;ARE LOADED.
                  ;
                  ;THE FOLLOWING MODIFICAIONS MUST BE MADE
                  ;TO THE MONITOR:
                  ;
                  ;       LOCATION        OLD     NEW
                  ;       386A            FB      00
                  ;       3F9F            FB      00
                  ;
                INTER:
3636 CD6D3F                     CALL    TI              ;INPUT CHARACTER
3639 FE45                       CPI     'E'             ;COMPARE WITH "E"
363B C24236                     JNZ     I1              ;JUMP IF NOT "E"
363E FB                         EI                      ;ENABLE INTERRUPTS
363F C36A38                     JMP     START           ;RETURN
                I1:
3642 FE44                       CPI     'D'             ;COMPARE WITH "D"
3644 C2203C                     JNZ     LER             ;ERROR IF NOT "D"
3647 F3                         DI                      ;DISABLE INTERRUPTS
3648 C36A38                     JMP     START           ;RETURN
                ;
0000                            END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AC2

☐ 4004 ☐ 8008 ☒ 8080 ☐ 4040

(use additional sheets if necessary)

Program Title

LIST 1

Function

HIGH SPEED LIST DEVICE FOR INTELLEC 8: REQUIRES NO ADDITIONAL OUTPUT BOARD BECAUSE THE PROM PROGRAMMER SOCKET IS USED FOR THE PRINTER CONNECTION.

Required Hardware

INTELLEC 8, ONE PROM CHIP, A HIGH SPEED PRINTER SUCH AS CENTRONICS 306, TELETYPEWRITER

Required Software

INTELLEC MONITOR VER. 1.0

Input Parameters

COMMAND "AL=1" IN MONITOR WILL CAUSE EXTERNAL PRINTER (CONNECTED VIA PROM PROGRAMMER) TO BE USED FOR "LIST" INFORMATION; (E.G. ASSEMBLER PRINTOUT IN PASS 2 OR 4).

Output Results

Registers Modified: ACCUMULATOR	**Maximum Subroutine Nesting Level:** 0
RAM Required: NONE OTHER THAN MONITOR'S STACK	**Assembler/Compiler Used:** 8080 ver 2.0
ROM Required: 35 BYTES	**Programmer:** Doug Raymond Zehntel
INSTALL AS CHIP #7, PROM BOARD 3	**Company:** 2440 Stanwell Drive Concord, Ca. 94520

```
;REF. NO. AC2
;PROGRAM TITLE HIGH SPEED LIST FOR INTELLEC 8
;
;
;
;
        ;*****AN EASILY CONNECTED LIST DEVICE*****
        ;
        ;INTELLEC 8 WITH 8080 CPU
        ;
        ;DOUG RAYMOND, ZEHNTEL, INC.
        ;
        ;
        ;THE FOLLOWING IS A SUBROUTINE WHICH
        ;PERMITS THE USE OF AN EXTERNAL LIST
        ;DEVICE (I.E. HIGH SPEED PRINTER) WITH
        ;THE INTELLEC 8 MONITOR VERSION 1.0
        ;
        ;PRINTER CONNECTION IS MADE VIA THE
        ;PROM PROGRAMMING SOCKET ON THE
        ;  INTELLEC'S FRONT PANEL.
        ;
        ;USE OF A HIGH SPEED PRINTER SPEEDS
        ;ASSEMBLY TREMENDOUSLY, BECAUSE
        ;ONLY TWO PASSES(P=1,P=4) ARE REQUIRED
        ;2)HIGHSPEED READER IS NOT SLOWED
        ;DOWN BY THE SLOW PRINTING ON THE CONSOLE.
        ;
        ;PRIOR TO RUNNING THE ASSEMBLER, TYPE
        ;"AL=1" INTO THE MONITOR, AND THIS
        ;SUBROUTINE WILL BE USED TO SEND
        ;"LIST" INFORMATION TO THE EXTERNAL
        ;HIGHSPEED PRINTER.
        ;
        ;THE ROUTINE IS SPECIFICALLY FOR A
        ;CENTRONICS 306, WHOSE CHARACTERISTICS
        ;ARE:  DATA-------PARALLEL 7 LINE ASCII
        ;      STROBE----NORMALLY HIGH
        ;      BUSY------HIGH WHEN BUSY
        ;THE CENTRONICS 306 HAS SEVERAL
        ;JUMPER OPTIONS.  TWO OF THESE ARE
        ;IMPORTANT FOR PROPER APPEARANCE
        ;OF THE LISTED MATERIAL.
        ;AUTOMATIC LINEFEED SHOULD BE DISABLED.
        ;OPTION DSC SHOULD BE ENABLED.
        ;
        ;DATA 0 OUT ON PIN 3
        ;     1            2
        ;     2            1
```

```
                        ;      3            21
                        ;      4            20
                        ;      5            19

                        ;      6          18
                        ;STROBE OUT ON PIN 17
                        ;BUSY IN ON PIN 11
                        ;GROUND BY A CLIP LEAD TO SLOT IN CHASSIS
                        ;BOTTOM.
                        ;A 24 PIN WIREWRAP SOCKET CAN BE USED
                        ;FOR A CONNECTOR.

                                ORG 3712H;ENTRY POINT FOR AL=1
3712
3712 C31A37                     JMP PRINT
                                ORG 371AH;OR OTHER AVAILABLE PROM
371A                                    ;LOCATION

371A 3E00               PRINT:  MVI A,0
371C D301                       OUT 1    ;ENABLES SENSING OF BUSY
371E DB02               PBZ:    IN 2
3720 07                         RLC
3721 D21E37                     JNC PBZ  ;CARRY IS TRUE IF BUSY WAS TRUE
 724 3E80                       MVI A, 80H
3726 D301                       OUT 1    ;RESTORES CONTROLS TO NORMAL
3728 79                         MOV A,C  ;CHARACTER IS IN C REG
3729 F680                       ORI 80H  ;BIT 7 IS USED AS STROBE
372B FEFF                       CPI 0FFH
372D C8                         RZ       ;DON'T SEND RUBOUT TO CENTRONICS 306
372E 2F                         CMA      ;I/O BOARD IS INVERTING
372F D302                       OUT 2
3731 F680                       ORI 80H
3733 D302                       OUT 2    ;STROBE IS NOW ACTIVE
3735 E67F                       ANI 7FH
3737 D302                       OUT 2    ;STROBE IS RETIRED
3739 C9                         RET
0000                            END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AC3

☐ 4004 ☐ 8008 ☒ 8080 ☐ 4040

(use additional sheets if necessary)

Program Title

I/O SIMULATION MACROS

Function

WHEN ASSEMBLED INTO A PROGRAM, THESE TWO MACROS, 'INPUT' and 'OUTPUT', ALLOW SIMULATION OF ALL 'IN' AND 'OUT' 8Ø8Ø INSTRUCTIONS. ALL 'IN'S' AND 'OUT'S' ARE LOGGED ON THE INTELLEC 8/MOD80 CONSOLE

Required Hardware

INTELLEC 8/MOD8Ø AND TTY

Required Software

MOD80 MONITOR

Input Parameters

Output Results

FOR OUT XX: OUTPUT XX = YY

FOR IN XX: INPUT XX = (EXPECT INPUT VALUE)

Registers Modified: A, FLAGS	**Maximum Subroutine Nesting Level:** ∿ 8 BYTES
RAM Required: 185 BYTES	**Assembler/Compiler Used:** 8Ø8Ø MACRO ASSEMBLER
ROM Required:	**Programmer:** K. BURGETT
	Company:

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

intel®

Ref. No. AB5

☐ 4004 ☐ 8008 ☒ 8080 ☐ 4040

(use additional sheets if necessary)

Program Title

Tape Duplicator

Function

Duplicates a tape read in from the H.S. tape reader by punching a copy on the TTY terminal with a leader added at both ends.

Required Hardware

TTY
H.S. Reader
Intellec 8

Required Software

Program Object Tape

Input Parameters

Load the program with the system monitor. Use the system monitor command(s) to change the memory at address Ø2 to ØlH. Place the tape to be copied in the H.S. reader. Turn on the tape punch and press reset on the console.

Output Results

A copy of the tape being read into the H.S. reader will be generated by the TTY punch with a leader at both the beginning and end of the tape. The program automatically returns to the system monitor.

Registers Modified:	Maximum Subroutine Nesting Level:
RAM Required:	Assembler/Compiler Used: Intellec 8 Macro Assembler
ROM Required:	Programmer: Jon Zoller
	Company: Dorsett Electronics

```
                        ;REF. NO. AB5
                        ;PROGRAM TITLE TAPE DUPLICATOR
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;INPUT FROM H.S. READER-OUTPUT TO TTY
                        LED     MACRO   BLNK
                                MVI     E,BLNK
                        CLEAR:  MVI     B,00
                                MOV     C,B
                                CALL    PO
                                DCR     E
                                JNZ     CLEAR
                                ENDM
3806                    RI      EQU     3806H
380C                    PO      EQU     380CH
0100                            ORG     100H
0100 310020                     LXI     SP,2000H
0103 CD0638             START:  CALL    RI          ;GET A CHARACTER
0106 57                         MOV     D,A
0107 A7                         ANA     A           ;IS CHARACTER BLANK?
0108 CA0301                     JZ      START       ;YES, IGNORE
                        +       LED     150
010B 1E96               +       MVI     E,00096H
010D 0600               +CLEAR: MVI     B,00
010F 48                 +       MOV     C,B
0110 CD0C38             +       CALL    PO
0113 1D                 +       DCR     E
0114 C20D01             +       JNZ     CLEAR


0117 4A                         MOV     C,D
0118 1E0F               AGAIN:  MVI     E,15
011A CD0C38             GO:     CALL    PO          ;PUNCH CHARACTER
011D CD0638                     CALL    RI          ;GET ANOTHER CHARACTER
0120 4F                         MOV     C,A
0121 A7                         ANA     A
0122 1D                         DCR     E
0123 C23501                     JNZ     JOHN
                        +       LED     100
0126 1E64               +       MVI     E,00064H
0128 0600               +CLEAR: MVI     B,00
012A 48                 +       MOV     C,B
012B CD0C38             +       CALL    PO
012E 1D                 +       DCR     E
012F C22801             +       JNZ     CLEAR
```

```
0132 C30038            JMP     3800H
0135 A7        JOHN:   ANA     A
0136 CA1A01            JZ      GO
0139 C31801            JMP     AGAIN
0000                   END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB6

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

CRC GEN

Function

Generate a 16 bit cyclic redundancy check (CRC) for a data string of up to 2^{16} bytes. The generator polynomial and initial conditions are defined by the user.

Required Hardware

None

Required Software

None

Input Parameters

BC contains size of data string in bytes
HL points to beginning of data string
DE contains initial value of CRC, usually all zeros or all ones
RAM required - 2 bytes plus data string
RAM or ROM - 60 bytes (subroutine only)

Output Results

All registers are modified
DE contains value of CRC
The data string is unmodified

Registers Modified:	Assembler/Compiler Used:
All	MAC80
RAM Required:	Programmer:
2 bytes plus character string	Geoffrey Karlin
ROM Required:	Company:
60 bytes	Applied Digital Data Systems, Inc.
Maximum Subroutine Nesting Level:	Address:
0	100 Marcus Blvd., Hauppauge, N.Y. 11787

```
                        ;REF. NO. : AB6
                        ;PROGRAM TITLE: CRC GEN-GENERATE A 16 BIT CYCLIC REDUNDANCY CHEC
                        ;
                        ;
                        ;CALLING PARAMTERS
                        ;
                        ;       (BC) - NO. OF BYTES
                        ;        HL - BEGINNING OF FILE
                        ;        DE - INITIAL CONDITION OF CRC REGISTER
                        ;
                        ;RETURNING PARAMETERS
                        ;
                        ;       (DE) - CRC
0100                            ORG     100H
                        ;

0100 0000               POINT:  DW      0000            ;MEMORY LOCATION NEEDED BY SUBRO
                        DATA:   DB      0FEH            ;TEST PATTERN, 5 BYTES
0103 00                         DB      00
0104 00                         DB      00              ;IF CRC REGISTER IS INIT.
 105 05                         DB      05H             ;TO ALL ONES, CRC IS 1E07
0106 00                         DB      00
                        ;
0021                    GX1     EQU     21H             ;FOR THIS EXAMPLE
0010                    GX2     EQU     10H             ;  G(X)=X(16)+X(12)+X(5)+1
                        ;
                        ;  CALLING PROGRAM FOR SUBROUTINE VERIFICATION
                        ;
0107 310001             CALLPR:LXI      SP,0100H        ;SET UP PARAMETERS TO ALL
010A 010500                     LXI     B,0005H         ;  CRCGEN
010D 210201                     LXI     H,DATA
0110 11FFFF                     LXI     D,0FFFFH
0113 CD1D01                     CALL    CRCGEN
0116 EB                         XCHG
0117 220001                     SHLD    POINT           ;TEST OVER, STORE CRC IN POINT
011A C30038                     JMP     3800H           ;RETURN TO 8080 MONITOR
                        ;
                        ;
                        ;  ACTUAL SUBROUTINE STARTS HERE
                        ;
                        ;
011D 220001             CRCGEN:SHLD     POINT           ;POINT IS LOC. OF NEXT P(X)
0120 C5                         PUSH    B
0121 E1                         POP     H               ;HL - BYTE CTR
0122 0E08                       MVI     C,8             ;C - BIT CTR
                        ;
0124 E5                 CRC2:   PUSH    H               ;LOAD (B) @ POINT
0125 2A0001                     LHLD    POINT           ;  (LOAD B NEXT P(X))
0128 46                         MOV     B,M
```

```
0129 23                 INX     H
012A 220001             SHLD    POINT
012D E1                 POP     H
                ;
012E 7B         CRC4:   MOV     A,E             ;SHIFT CRC LEFT (DE)
012F 83                 ADD     E
0130 5F                 MOV     E,A
0131 7A                 MOV     A,D
0132 8A                 ADC     D
0133 57                 MOV     D,A
0134 DA5401             JC      CRC3            ;JUMP IF MSB = 1
                ;
0137 78                 MOV     A,B             ;SHIFT P (X) LEFT (B)
0138 80                 ADD     B
0139 47                 MOV     B,A
013A D24501             JNC     CRCOUT          ;JUMP IF MSB = 0
                ;
013D 7B         CRCXOR:MOV      A,E             ;XOR CRC & G(X)
013E EE21               XRI     GX1
0140 5F                 MOV     E,A
0141 7A                 MOV     A,D
 142 EE10               XRI     GX2
0144 57                 MOV     D,A
                ;
0145 0D         CRCOUT:DCR      C
                                                ;CK BIT CNT
0146 C22E01             JNZ     CRC4
0149 0E08               MVI     C,8
014B 2D                 DCR     L               ;CK WORD CNT
014C C22401             JNZ     CRC2
014F 25                 DCR     H
0150 F22401             JP      CRC2
0153 C9                 RET
                ;
0154 78         CRC3:   MOV     A,B             ;SHIFT LEFT P(X)
0155 80                 ADD     B
0156 47                 MOV     B,A
0157 DA4501             JC      CRCOUT          ;JUMP IF MSB = 1
015A C33D01             JMP     CRCXOR          ;JUMP TO XOR, MSB = 0
                ;
0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB7

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

16 bit CRC for polynomial X16+X12+X5+1 (polynomial for SDLC)

Function

Produces a 16 bit CRC with 8 bit input bytes. Care should be taken with Most/Least bit feeding of the data byte and CRC residue. Does not require a table or contain any loops. Requires 24H memory locations and executes in 72.5 usec.

Required Hardware

None

Required Software

Main program to include routine as in-line code or modified routine to be called as subroutine.

Input Parameters

Register "A" is data byte
Register "B" is low byte of CRC residue
Register "C" is high byte of CRC residue

Output Results

CRC residue in registers B & C

Registers Modified: A, B, C, D, E, L	**Assembler/Compiler Used:** 8080 Macro Assembler, Ver 1.0
RAM Required: 24H bytes, OR	**Programmer:** Bill McGonigal
ROM Required: 24H bytes	**Company:** Sycor
Maximum Subroutine Nesting Level:	**Address:** 100 Phoenix Drive Ann Arbor, Michigan 48104

```
                ;REF NO.  AB7
                ;PROGRAM 16 BIT CRC FOR POLYNOMIAL X16+X12+X5+1 (POLYNOMIAL FOR
                ;
                ;
                ;
                ;
                ;CRC SUBROUTINE
                ;POLYNOMIAL IS X16 +X12 +X5 +1
                ;CRC USES MOST SIGNIFICANT BIT (BACKWARDS)
                ;REG B IS LOW BYTE OF CRC RESIDUE
                ;REG C IS HIGH BYTE OF CRC RESIDUE
                ;DATA BYTE IS EXPECTED IN THE
                ;ACCUMULATOR
                ;
                ;

0100                    ORG     100H
0100 A9         ENTRY:  XRA     C               ;X-OR DATA BYTE
                                                ;WITH CRC HIGH BYTE
0101 57                 MOV     D,A             ;TEMP STORE IN REG D
 102 0F                 RRC                     ;ROTATE RIGHT FOUR BITS
0103 0F                 RRC
0104 0F                 RRC
0105 0F                 RRC
0106 5F                 MOV     E,A
0107 AA                 XRA     D
0108 E6F0               ANI     0F0H
010A A8                 XRA     B
010B 6F                 MOV     L,A             ;RESTORE (TEMP SAVE)
010C 7B                 MOV     A,E
010D 07                 RLC
010E E61F               ANI     1FH             ;  MASK
0110 AD                 XRA     L
0111 6F                 MOV     L,A             ;RESTORE
0112 7A                 MOV     A,D
0113 07                 RLC
0114 E601               ANI     1               ;MASK
0116 A8                 XRA     B
0117 AD                 XRA     L
0118 47                 MOV     B,A             ;LOW BYTE IS COMPLETE
0119 7B                 MOV     A,E
011A E60F               ANI     0FH             ;MASK
011C AA                 XRA     D
011D 4F                 MOV     C,A             ;STORE
011E 7B                 MOV     A,E
 11F AA                 XRA     D
 120 07                 RLC
0121 E6E0               ANI     0E0H            ;MASK
0123 A9                 XRA     C
```

```
0124 4F                 MOV     C,A
0125 76                 HLT                     ;(NOT PART OF
                                                ;SUBROUTINE)

0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB8

☐ 4004 ☐ 4040 ☐ 8008 ☐ 8080 (use additional sheets if necessary)

Program Title

CRC16

Function

This macro calculates a CRC16 check word using the generation polynomial X**16+X**15+X**2+1. It can be used to generate a check word for a record that doesn't include one or to check that a record including a check word is correct.

Required Hardware

8080

Required Software

MAC80 for assembly

Input Parameters

"BUFF" is the address to a word that contains the address to the first byte in the record to be included in the CRC16 accumulation.

"LEN" is the address to a word that contains the number of bytes to be included in the CRC16 accumulation not including the two check bytes. The record for which the CRC16 is to be calculated must occupy consecutive bytes in memory. If a CRC16 is to be generated for a record that doesn't contain one, the two bytes following the record where the CRC16 usually lies must be zero.

Output Results

The CRC16 check word is left in the BC register pair, B being the most significant byte. The HL register pair contains the address to the least significant check byte in the record. The CPU zero flag is set if and only if the BC register pair =0. This occurs if the record contains a correct CRC16 check word. The A, D, and E registers are left undefined. The record address vector and record length are not changed.

Registers Modified: All	**Assembler/Compiler Used:** MAC80
RAM Required: 6 bytes (in stack)	**Programmer:** Gerald Berger
ROM Required: First call: 64 bytes Other calls: 10 bytes	**Company:** Cybernetics Int. AB Box 48
Maximum Subroutine Nesting Level:	**Address:** Danderyd, Sweden

```
;REF. NO. AB8.
;PROGRAM NAME CRC16

;
;
;
;
;
;       TITLE           CRC16
;
;       FUNCTION        THIS MACRO CALCULATES A CRC16 CHECK WORD
;            THE GENERATION POLYNOMIAL X**16+X**15+X**2+1.  IT
;            CAN BE USED TO GENERATE A CHECK WORD FOR A RECORD T
;            DOESN'T INCLUDE ONE OR TO CHECK THAT A RECORD INCLU
;            A CHECK WORD IS CORRECT
;
;       REQUIRED HARDWARE               8080
;
;       REQUIRED SOFTWARE           --  MAC80 FOR ASSEMBLY
;
;       INPUT PARAMETERS
;            THE MACRO CALL INCLUDES 2 PAAMETERS
;
;            BUFF       "BUFF" IS THE ADDRESS TO A WORD THAT CON
;            THE ADDRESS TO THE FIRST BYTE IN THE RECORD TO BE
;            INCLUDED IN THE CRC16 ACCUMULATION.
;
;            LEN        "LEN" IS THE ADDRESS TO A WORD THAT CONT
;            THE NUMBER OF BYTES TO BE INCLUDED IN THE CRC16
;            ACCUMULATION NOT INCLUDING THE TWO CHECK BYTES.
;            THE RECORD FOR WHICH CRC16 IS TO BE CALCULATED MUST
;            CONSECUTIVE BYTES IN MEMORY.  IF A CRC16 IS TO BE
;            GENERATED FOR A RECORD THAT DOEN'T CONTAIN ONE, THE
;            BYTES FOLLOWING THE RECORD WHERE THE CRC16 USUALLY
;            MUST BE ZERO.
;
;       OUTPUT RESULTS
;
;            THE CRC16 CHECK WORD IS LEFT IN THE BC REGISTER PAI
;            B BEING THE MOST SIGNIFICANT BYTE.  THE
;            HL RREGISTER PAIR CONTAINS THE ADDRESS TO THE LEAST
;            SIGNIFICANT CHECK BYTE IN THE RECORD.  THE
;            CPU ZERO FLAG IS SET IF AND ONLY IF THE BC
;            REGISTER PAIR = 0.  THIS OCCURS IF THE RECORD
;            CONTAINS A CORRECT CRC16 CHECK WORD.  THE A, D, AND E
;            REGISTERS ARE UNDEFINED.  THE RECORD ADDRESS VECTOR
;            AND RECORD LENGTH ARE NOT CHANGED.
;
;       DESCRIPTION
;
```

```
                ;           THE MACRO USES A SUBROUTINE XCRCS TO GENERATE THE
                ;           CRC16 CHECK WORD.  THE CODE FOR THIS SUBROUTINE IS
                ;           GENERATED BY THE MACRO AT THE FIRST CALL TO THE MAC
                ;           ONLY.  XCRCS USES 4 BYTES OF RAM MEMORY IN THE STACK
                ;           FOR TEMPORARY DATA STORAGE AND IS THUS REENTRANT.
                ;
                ;       TIMING      71600 STATES PER 100 BYTES
                ;
                ;       RAM REQUIRED    6 BYTES IN STACK
                ;
                ;       ROM REQUIRED    FIRST CALL      64 BYTES
                ;                       OTHER CALLS     10 BYTES
                ;
1000                    ORG     1000H           ;USED FOR TEST PURPOSES ONLY
                ;
                ;               CRC16
                ;               -----
                ;
FFFF            XCRCF   SET NOT 0
                CRC16   MACRO   BUFF,LEN
                        LHLD    LEN
                        XCHG                    ;   LENGTH OF RECORD IN DE
                        LHLD    BUFF            ;ADDRESS TO BUFFER IN HL
                        IF      XCRCF           ;GENERATE CODE IF FIRST CALL TO
                XCRCF   SET     0               ;
                        JMP     BYPAS           ;HOP PAST SUBROUTINE
                XCRCS::
                        MOV     B,M             ;FIRST BYTE OF RECORD TO B
                        INX     H
                        MOV     C,M             ;SECOND BYTE TO C
                NBYF:                           ;PROCESS NEXT BYTE
                        INX     H               ;ADDRESS TO NEXT BYTE IN HL
                        MOV     A,M             ;NEW BYTE TO A
                        PUSH    H               ;STORE CURRENT ADDRESS IN STACK
                        PUSH    D
                        LXI     H,0A001H        ;GENERATION POLYNOMIAL  TO HL
                        MVI     E,8             ;INITIALIZE BIT COUNT IN E
                NBIT:                           ;PROCESS NEXT BIT
                                                ;SHIFT NEXT BIT OF NEW BYTE INTO
                        RAR
                        MOV     D,A             ;SAVE NEW BYTE IN D
                        MOV     A,C
                        RAR
                        MOV     C,A             ;AND INTO C
                        MOV     A,B             ;SHIGTFT MOST SIGNIFICANT BIT IN
                        RAR
                        MOV     B,A
                        JNC     NODIV
                                                ;HERE IF SDIVISABLE
                        XRA     L
```

```
                        MOV     B,A
                        MOV     A,C
                        XRA     H
                        MOV     C,A
                NODIV:
                        MOV     A,D             ;NEW BYTE BACK TO A
                        DCR     E
                        JNZ     NBIT

                                                ;HERE IF BYTE DONE
                        POP     D               ;NUMBER OF BYRTES LEFT TO DE
                        POP     H               ;ADDRESS TO LAST BYTE PROCESSED
                        DCR     E               ;DECREASE LENGTH COUNT BY 1
                        JNZ     NBYTE           ;TAKE NEXT BYTE IF NOT DONE
                        DCR     D
                        JP      NBYTE

                                                ;HERE IF RECORD DONE
                                                ;SET ZERO FLAG IF BC=0
                        MOV     A,B
                        ANA     A
                        RNZ
                        ADD     C
                        RET

                BYPAS:
                        ENDIF
                        CALL    XCRCS           ;CALCULATE CRC16
                        ENDM
0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB9

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 (use additional sheets if necessary)

Program Title

CRECH

Function

Computes CRC characters for IBM compatible floppy disk. Also works for Synchronous Data Link Control (SDLC).

Required Hardware

None

Required Software

None

Input Parameters

HL = start address of data
DE = end address of data

Output Results

BC = CRC characters

Registers Modified: A,B,C,H,L	**Assembler/Compiler Used:** 8080 Assembler
RAM Required: 50_{10}	**Programmer:** Dick Springer
ROM Required: None	**Company:** General Microwave Corporation
Maximum Subroutine Nesting Level: None	**Address:** 155 Marine Street Farmingdale, N.Y. 11735

98-034B

```
                        ;REF NO. AB9
                        ;PROGRAM CRECH
                        ;
                        ;
                        ;
                        ;
                        ;SUBROUTINE CRECH COMPUTES THE CRC CHECK WORDS REQUIRED FOR
                        ;READING FROM OR WRITING ON IBM-FORMAT FLOPPY DISKS. THE TWO-
                        ;BYTE CHECK WORD FOR A BLOCK OF DATA IS THE WORD PRODUCED BY
                        ;THE GENERATING POLYNOMIAL X**16+X**12+X**5+1 WHEN THE CHECK
                        ;WORD IS INITIALIZED TO THE VALUE 0FFFFH.
                        ;
                        ;THE SUBROUTINE INITIALIZES THE CHECK WORD TO 0FFFFH AND
                        ;PROCESSES A BLOCK OF DATA WITH INITIAL MEMORY LOCATION IN
                        ;REGISTER PAIR (H,L) AND FINAL MEMORY LOCATION IN REGISTER
                        ;PAIR (D,E). THE CHECK WORD APPEARS IN REGISTER PAIR (B,C)
                        ;WITH THE FIRST BYTE ON THE DISK IN REGISTER C AND THE SECOND
                        ;BYTE IN REGISTER B.
                        ;
                        ;IF A DIFFERENT CHECK WORD INITIALIZATION IS DESIRED, AS MAY
                        ;BE THE CASE IF PROCESSING IS TO BE CONTINUED AFTER PROCESSING
                        ;A BLOCK ELSEWHERE IN MEMORY, THE INITIAL VALUE SHOULD BE
                        ;INSERTED IN (B,C) AND THE ROUTINE ENTERED AT LOCATION CRECH+3.
                        ;
                        ;AFTER PROCESSING, REGISTER PAIR (B,C) CONTAINS THE CHECK WORD,
                        ;REGISTER PAIR (D,E) CONTAIN THE INPUT FINAL ADDRESS, AND
                        ;REGISTER PAIR (H,L) CONTAIN THE ADDRESS FOLLOWING THAT OF
                        ;THE LAST BYTE PROCESSED.
                        ;
                        ;IF THE FINAL ADDRESS PRECEDES THE INITIAL ADDRESS ONE BYTE
                        ;IS PROCESSED.
0000 01FFFF     CRECH:  LXI     B,0FFFFH
0003 D5                 PUSH    D
0004 7E                 MOV     A,M
0005 A9                 XRA     C
0006 57                 MOV     D,A
0007 0F                 RRC
0008 0F                 RRC
0009 0F                 RRC
000A 0F                 RRC
000B E60F               ANI     0FH
000D AA                 XRA     D
000E 5F                 MOV     E,A
000F 0F                 RRC
0010 0F                 RRC
0011 0F                 RRC
0012 57                 MOV     D,A
0013 E61F               ANI     1FH
0015 A8                 XRA     B
```

```
0016 4F          MOV     C,A
0017 7A          MOV     A,D
0018 E6E0        ANI     0E0H
001A AB          XRA     E
001B 47          MOV     B,A
001C 7A          MOV     A,D
001D 0F          RRC
001E E6F0        ANI     0F0H
0020 A9          XRA     C
0021 4F          MOV     C,A
0022 23          INX     H
0023 D1          POP     D
0024 7A          MOV     A,D
0025 BC          CMP     H
0026 D8          RC
0027 C20300      JNZ     CRECH+3
002A 7B          MOV     A,E
002B BD          CMP     L
002C D8          RC
002D C30300      JMP     CRECH+3
0000             END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB10

☐ 4004 ☐ 4040 ☒ 8008 ☐ 8080

(use additional sheets if necessary)

Program Title

Legible paper tape

Function

The program punches legible characters on paper tape, useful for tape labelling. The characters are formed in the conventional 5x7 matrix and the 64 character sub-set of ASCII is represented. The characters to be punched are taken from the console device and decoded as an index into a table of bit patterns. Console input is convenient where a separate punch is available but is not a necessary part of the logic: a list of characters could as easily be referenced. Carriage return, line feed and the control characters are echoed but not punched.

Required Hardware

Hardware: console and paper tape punch

Required Software

Console I/O and punch output routines of Intellec 8 monitor.

Input Parameters

ASCII characters in A

Output Results

Legible paper tape: bit patterns presented to punch output routine in B

Registers Modified: A,B,C,D,E,H,L	**Assembler/Compiler Used:** ISIS 8080 Macro Assembler, V1.0
RAM Required: 387 decimal	**Programmer:** Intellec 8 Macro assembler
ROM Required:	**Company:** P.R. Cave - DR
Maximum Subroutine Nesting Level:	**Address:** Dept. Mechanical Engrg. The University, Dundee DD1 4HN England

98-034B

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB11

☐ 4004 ☐ 4040 ☒ 8008 ☒ 8080 (use additional sheets if necessary)

Program Title

Banner Print & Punch

Function

Create, on a listing device or tape perforator, a graphic representation of certain ASCII characters.

Required Hardware

CBAN Console Keyboard, List Device
LBAN List Device
CPBN Console Keyboard, Tape Perforator
PBAN Tape Perforator

Required Software

None

Input Parameters

CBAN Console Keyboard Input
CPBN

LBAN The B and C registers contain the high and low bytes respectively
PBAN of the address of a buffer. The buffer contains the ASCII codes for the characters to be printed or punched, and is terminated by the ASCII code for carriage return (0DH). Care should be taken that the graphic representations of the specified characters will fit on one line of the list device. Legal characters include the alphabet, the digits and special characters: Space, comma, *, +,-, ., /, :, ;,<,=,>,?, and @.

Output Results

CBAN Graphic representations of the input characters drawn on the out-
LBAN put from the listing device. Each character is drawn in a 1, 3, or 5 by 7 matrix with 2 columns between characters.

CPBN Graphic representations of the input characters drawn on perfor-
PBAN ated tape. Each character is drawn in a 1, 3, or 5 by 7 matrix with 2 blanks between characters.

Registers Modified: All	**Assembler/Compiler Used:** 8008 Macro Assembler Ver. 20
RAM Required: 46 bytes	**Programmer:** C. E. Ohme
ROM Required: 657 bytes	**Company:** Ohme
Maximum Subroutine Nesting Level: 3	**Address:** 44750 Winding Lane Fremont, CA 94538

```
                ;REF. NO. AB11
                ;PROGRAM TITLE BANNER PRINT & PUNCH
                ;
                ;
                ;
                ;
1000                    ORG     1000H


                ;       BANNER ROUTINES

                ;  MACROS TO SIMULATE 8080 INSTRUCTIONS
                ;  (VALID ONLY IN THESE ROUTINE)

000D            CR      EQU     0DH             ;CARRIAGE RETURN
000A            LF      EQU     0AH             ;LINE FEED

                ;       THE CBAN ROUTINE ACCEPTS CHARACTERS
                ;       FROM THE CONSOLE AND DRAWS THEM ON
                ;       THE LIST DEVICE.  A CARRUAGE RETURN
                ;       TERMINATES EACH LINE.  PLEASE STAY
                ;       ON THE PAPER.

1000 CD5E11     CBAN:   CALL    CRLF            ;PRINT CR LF
1003 1640               MVI     D,40H           ;MASK FOR LINE 1
1005 21CB12             LXI     H,BUFF          ;BUFFER ADDR
1008 22C512             SHLD    BADR            ;INIT BADR
100B 7C                 MOV     A,H
100C 45                 MOV     B,L
100D 21C512             LXI     H,BADR
1010 70                 MOV     M,B
1011 23                 INX     H
1012 CDC112             CALL    INCHL

1015 77                 MOV     M,A
1016 67                 MOV     H,A
1017 68                 MOV     L,B

1018 22C712     CB1:    SHLD    CADR            ;SAVE CAHR ADDR
101B 7C                 MOV     A,H
101C 45                 MOV     B,L
101D 21C712             LXI     H,CADR
1020 70                 MOV     M,B
1021 23                 INX     H
1022 CDC112             CALL    INCHL

 025 77                 MOV     M,A
1026 67                 MOV     H,A
1027 68                 MOV     L,B
```

```
1028 CDB512            CALL    CI              ;INPUT A CHAR
102B E67F              ANI     7FH             ;REMOVE PARITY
102D 5F                MOV     E,A             ;SAVE CHAR
102E 2AC712            LHLD    CADR            ;RESTORE CHAR ADDR

1031 21C712            LXI     H,CADR
1034 7E                MOV     A,M
1035 23                INX     H
1036 CDC112            CALL    INCHL

1039 66                MOV     H,M
103A 6F                MOV     L,A

103B 7B                MOV     A,E             ;CHAR TO DRAW
103C 77                MOV     M,A             ;STORE CHAR

103D FE0D              CPI     CR              ;CAP RET
103F CA6210            JZ      EOP             ;IF END OF INPUT
1042 CDEC10            CALL    CONV            ;CONVERT CHAR
1045 DA1810            JC      CB1             ;IF ILLEGAL
 348 CD1E11            CALL    BCH             ;PRINT LINE 1
104B 2AC712            LHLD    CADR            ;RESTORE CHAR ADDR
104E 21C712            LXI     H,CADR
1051 7E                MOV     A,M
1052 23                INX     H
1053 CDC112            CALL    INCHL

1056 66                MOV     H,M
1057 CDC112            CALL    INCHL
105A 6F                MOV     L,A

105B 23                INX     H               ;FOR NEXT CHAR
105C CDC112            CALL    INCHL

105F C31810            JMP     CB1             ;LOOP
1062 CDB210    EOP:    CALL    EOL             ;COMPLETE PICTURE
1065 CD5E11            CALL    CRLF            ;PRINT CR LF
1068 CD5E11            CALL    CRLF            ;PRINT CR LF
106B C30010            JMP     CBAN            ;FOR NEXT PICTURE

               ;       THE LBAN SUBROUTINE ACCEPTS THE ADDRESS OF A
               ;       BUFFER IN THE B AND C REGISTERS.  ASCI
               ;       CHARACTERS IN THE BUFFER ARE DRAWN ON THE
               ;       LIST DEVICE UNTIL AN ASCII CARRIAGE RETURN
               ;       IA ENCOUNTERED.

 J6E 60        LBAN:   MOV     H,B             ;BUFFER ADDR H
106F 69                MOV     L,C             ;BUFFER ADDR L
1070 22C512            SHLD    BADR            ;SAVE BUFFER ADDR
```

```
1073 7C              MOV     A,H
1074 45              MOV     B,L
1075 21C512          LXI     H,BADR
1078 70              MOV     M,B
1079 23              INX     H
107A CDC112          CALL    INCHL

107D 77              MOV     M,A
107E 67              MOV     H,A
107F 68              MOV     L,B

1080 CD5E11          CALL    CRLF            ;PRINT CR LF
1083 1640            MVI     D,40H           ;MASK FOR LINE 1
1085 2AC512    LB1:  LHLD    BADR            ;BUFFER ADDR
1088 21C512          LXI     H,BADR
108B 7E              MOV     A,M
108C 23              INX     H
108D CDC112          CALL    INCHL

1090 66              MOV     H,M
1091 6F              MOV     L,A

1092 7E        LB2:  MOV     A,M             ;CHAR TO DRAW
1093 FE0D            CPI     CR              ;CAR RET
1095 CAB210          JZ      EOL             ;IF END OF LINE
1098 CDEC10          CALL    CONV            ;CONVERT CHAR
109B D41E11          CNC     BCH             ;IF LEGAL CHAR
109E 2AC712          LHLD    CADR            ;RESTORE CHAR ADDR
10A1 21C712          LXI     H,CADR
10A4 7E              MOV     A,M
10A5 23              INX     H
10A6 CDC112          CALL    INCHL

10A9 66              MOV     H,M
10AA 6F              MOV     L,A

10AB 23              INX     H               ;FOR NEXT CHAR
10AC CDC112          CALL    INCHL

10AF C39210          JMP     LB2             ;LOOP
10B2 CD5E11    EOL:  CALL    CRLF            ;PRINT CRLF
10B5 7A              MOV     A,D             ;MASK
10B6 1F              RAR                     ;FOR NEXT LINE
10B7 E67F            ANI     7FH             ;REMOVE TOP BIT
10B9 C8              RZ                      ;IF PICTURE DONE
10BA 57              MOV     D,A             ;MASK
'0BB C38510          JMP     LB1             ;LOOP

               ;     THE CPBN ROUTINE ACCEPTS CHARACTERS
               ;     FROM THE CONSOLE AND DRAWS THEM ON
```

```
                ;       THE PUNCH DEVICE.
10BE CDB512     CPBN:   CALL    CI              ;INPUT A CHAR
10C1 E67F               ANI     7FH             ;REMOVE PARITY
10C3 CDEC10             CALL    CONV            ;CONVERT CHAR
10C6 D46811             CNC     PCH             ;IF LEGAL CHAR
10C9 C3BE10             JMP     CPBN            ;LOOP

                ;       THE PBAN SUBROUTINE ACCEPTS THE ADDRESS OF A
                ;       BUFFER IN THE B AND C REGISTERS.  ASCII
                ;       CAHARACTERS IN THE BUFFER ARE DRAW ON THE
                ;       PUNCH DEVICE UNTIL AN ASCII CARRIAGE RETURN
                ;       IS ENCOUNTERED.

10CC 60         PBAN:   MOV     H,B             ;BUFFER ADDR H
10CD 69                 MOV     L,C             ;BUFFER ADDR L
10CE 7E         PB1:    MOV     A,M             ;CHAR TO DRAW
10CF FE0D               CPI     CR              ;CAR RET

10D1 C8                 RZ                      ;IF END OF BUFFER
10D2 CDEC10             CALL    CONV            ;CONVERT CHAR
10D5 D46811             CNC     PCH             ;IF LEGAL CHAR
 )D8 2AC712             LHLD    CADR            ;RESTORE CHAR ADDR
10DB 21C712             LXI     H,CADR
10DE 7E                 MOV     A,M
10DF 23                 INX     H
10E0 CDC112             CALL    INCHL

10E3 66                 MOV     H,M
10E4 6F                 MOV     L,A

10E5 23                 INX     H               ;FOR NEXT CHAR
10E6 CDC112             CALL    INCHL

10E9 C3CE10             JMP     PB1             ;LOOP

                ;       CONV CONVERTS THE CHARACTER IN A TO A CRAF
                ;       TABLE ORDINAL IN H AND L.  THE CHARACTER IS
                ;       SAVED IN E.  THE ORIGINAL H AND L ARE STORED
                ;       AT CADR.  D IS PRESERVED.

10EC 5F         CONV:   MOV     E,A             ;SAVE CHAR
10ED 22C712             SHLD    CADR            ;SAVE CHAR ADDR
10F0 7C                 MOV     A,H
10F1 45                 MOV     B,L
10F2 21C712             LXI     H,CADR
10F5 70                 MOV     M,B
 9F6 23                 INX     H
  F7 CDC112             CALL    INCHL

10FA 77                 MOV     M,A
```

```
10FB 67                 MOV     H,A
10FC 68                 MOV     L,B

10FD 7B                 MOV     A,E             ;CHAR TO CONVERT
10FE D620               SUI     ' '             ;SPACE
1100 D8                 RC                      ;IF < SPACE
1101 CA1211             JZ      FND             ;IF SPACE
1104 D60A               SUI     '*'-' '                 ;* = 0
1106 D8                 RC                      ;IF< *
1107 FE31               CPI     'Z'+1-'*'       ;CHECK FOR > Z
1109 C21B11             JNZ     CNVER           ;IF > Z
110C C601               ADI     1               ;* = 1
110E 47                 MOV     B,A             ;CHAR INDEX
110F 87                 ADD     A               ;2 * CHAR INDEX
1110 87                 ADD     A               ;4 * CHAR INDEX
1111 80                 ADD     B               ;5 * CHAR INDEX
1112 C6A2       FND:    ADI     GRAF AND 0FFH ; GRAF BASE L
1114 6F                 MOV     L,A             ;GRAF L
1115 3E11               MVI     A,GRAF SHR 8    ;GRAF BASE H
1117 CE00               ACI     0               ;GRAF H
1119 67                 MOV     H,A             ;GRAF H
 .1A C9                 RET
.11B C6FF       CNVER:  ADI     0FFH            ;SET CARRY
111D C9                 RET

                ;       BCH PRINTS ONE LINE OF THE CHARACTER
                ;       WHOSE GRAF ORDINAL IS IN H AND L UPON
                ;       ENTRY.  THE LINE PRINTED IS DETERMINED
                ;       BY THE MASK IN D.

111E 0E05       BCH:    MVI     C,5             ;COL COUNT
1120 22C912     BCH1:   SHLD    GADR            ;SAVE GRAF ADDR
1123 7C                 MOV     A,H
1124 45                 MOV     B,L
1125 21C912             LXI     H,GADR
1128 70                 MOV     M,B
1129 23                 INX     H
112A CDC112             CALL    INCHL

112D 77                 MOV     M,A
112E 67                 MOV     H,A
112F 68                 MOV     L,B

1130 7E                 MOV     A,M             ;GRAF CODE
1131 A7                 ANA     A               ;SET FLAGS
1132 FA5411             JM      BCH3            ;IF END OF CHAR
'1.35 43                MOV     B,E             ;CHAR TO PRINT
 .36 A2                 ANA     D               ;GRAF BIT
1137 C23C11             JNZ     BCH2            ;TO PRINT CHAR
113A 0620               MVI     B,' '           ;SPACE
```

```
113C CD9C12   BCH2:   CALL    LO              ;PRINT CHAR OR SP
113F 2AC912           LHLD    GADR            ;RESTORE GRAF ADDR
1142 21C912           LXI     H,GADR          ;
1145 7E               MOV     A,M
1146 23               INX     H
1147 CDC112           CALL    INCHL

114A 66               MOV     H,M
114B 6F               MOV     L,A

114C 23               INX     H               ;FOR NEXT GRAF
114D CDC112           CALL    INCHL

1150 0D               DCR     C               ;DECR COL CNT
1151 C22011           JNZ     BCH1            ;IF MORE
1154 0620     BCH3:   MVI     B,' '           ;SPACE
1156 CD9C12           CALL    LO              ;BETWEEN CHARS
1159 0620             MVI     B,' '           ;SPACE
115B C39C12           JMP     LO              ;SPACE AND RET

              ;       CRLF PRINTS CARRIAGE RETURN, LINE FEED
 115E 060D    CRLF:   MVI     B,CR            ;CAR RET
1160 CD9C12           CALL    LO              ;LIST
1163 060A             MVI     B,LF            ;LINE FEED
1165 C39C12           JMP     LO              ;LIST AND RET

              ;       PCH PUNCHES THE CHARACTER WHOSE GRAF
              ;       ORDINAL IS IN H AND L UPON ENTRY

1168 0E05     PCH:    MVI     C,5             ;CHAR COUNT
116A 22C912   PCH1:   SHLD    GADR            ;SAVE GRAF ADDR
116D 7C               MOV     A,H
116E 45               MOV     B,L
116F 21C912           LXI     H,GADR
1172 70               MOV     M,B
1173 23               INX     H
1174 CDC112           CALL    INCHL
1177 77               MOV     M,A
1178 67               MOV     H,A
1179 68               MOV     L,B

117A 7E               MOV     A,M             ;GRAF CODE
117B A7               ANA     A               ;SET FLAGS
117C FA9811           JM      PCH2            ;IF END OF CHAR
117F 47               MOV     B,A             ;GRAF CODE

 1180 CDB012          CALL    PO              ;PUNCH CODE
  1183 2AC912         LHLD    GADR            ;RESTORE GRAF ADDR
1186 21C912           LXI     H,GADR
1189 7E               MOV     A,M
```

```
118A 23                 INX     H
118B CDC112             CALL    INCHL

118E 66                 MOV     H,M
118F 6F                 MOV     L,A

1190 23                 INX     H               ;FOR NEXT GRAF
1191 CDC112             CALL    INCHL

1194 0D                 DCR     C               ;DECR CHAR CNT
1195 C26A11             JNZ     PCH1            ;IF MORE
1198 0600       PCH2:   MVI     B,0             ;BLANK

119A CDB012             CALL    PO              ;BETWEEN CHARS
119D 0600               MVI     B,0             ;BLANK
119F C3B012             JMP     PO              ;BLANK AND RET

                ;       THE GRAF TABLE CONTAINS THE GRAPHIC
                ;       DESCRIPTION OF THE LEGAL CHARACTERS.

11A2 00000000 GRAF:     DB      00H,00H,00H,00H,00H     ;SP
 1A6 00
11A7 14081480           DB      14H,08H,14H,80H,80H     ;*
11AB 80
11AC 081C0880           DB      08H,1CH,08H,80H,80H     ;+
11B0 80
11B1 03808080           DB      03H,80H,80H,80H,80H     ;,
11B5 80
11B6 08080880           DB      08H,08H,08H,80H,80H     ;-
11BA 80
11BB 01808080           DB      01H,80H,80H,80H,80H     ;.
11BF 80
11C0 03040810           DB      03H,04H,08H,10H,60H     ;/
11C4 60
11C5 3E414141           DB      3EH,41H,41H,41H,3EH     ;0
11C9 3E
11CA 217F0180           DB      21H,7FH,01H,80H,80H     ;1
11CE 80
11CF 21434549           DB      21H,43H,45H,49H,36H     ;2
11D3 36
11D4 22414949           DB      22H,41H,49H,49H,36H     ;3
11D8 36
11D9 0C14247F           DB      0CH,14H,24H,7FH,04H     ;4
11DD 04
11DE 7A494949           DB      7AH,49H,49H,49H,46H     ;5
11E2 46
 1E3 3E494949           DB      3EH,49H,49H,49H,26H     ;6
 1E7 26
11E8 43444850           DB      43H,44H,48H,50H,60H     ;7
11EC 60
```

```
11ED 36494949          DB      36H, 49H, 49H, 49H, 36H     ;8
11F1 36
11F2 32494949          DB      32H, 49H, 49H, 49H, 3EH     ;9
11F6 3E
11F7 12808080          DB      12H, 80H, 80H, 80H, 80H     ;:
11FB 80
11FC 13808080          DB      13H, 80H, 80H, 80H, 80H     ;;
1200 80
1201 08142241          DB      08H, 14H, 22H, 41H, 41H     ;<
1205 41
1206 14141480          DB      14H, 14H, 14H, 80H, 80H     ;=
120A 80
120B 41412214          DB      41H, 41H, 22H, 14H, 08H     ;>
120F 08
1210 20404548          DB      20H, 40H, 45H, 48H, 30H     ;?
1214 30
1215 7F404F41          DB      7FH, 40H, 4FH, 41H, 7FH     ;@
1219 7F
121A 7F484848          DB      7FH, 48H, 48H, 48H, 7FH     ;A
121E 7F
121F 7F494949          DB      7FH, 49H, 49H, 49H, 36H     ;B
 223 36
1224 7F414141          DB      7FH, 41H, 41H, 41H, 41H     ;C
1228 41
1229 7F414141          DB      7FH, 41H, 41H, 41H, 3EH     ;D
122D 3E
122E 7F494949          DB      7FH, 49H, 49H, 49H, 41H     ;E
1232 41
1233 7F484848          DB      7FH, 48H, 48H, 48H, 40H     ;F
1237 40
1238 7F414149          DB      7FH, 41H, 41H, 49H, 4FH     ;G
123C 4F
123D 7F080808          DB      7FH, 08H, 08H, 08H, 7FH     ;H
1241 7F
1242 417F4180          DB      41H, 7FH, 41H, 80H, 80H     ;I
1246 80
1247 03010101          DB      03H, 01H, 01H, 01H, 7FH     ;J
124B 7F
124C 7F081422          DB      7FH, 08H, 14H, 22H, 41H     ;K
1250 41
1251 7F010101          DB      7FH, 01H, 01H, 01H, 01H     ;L
1255 01
1256 7F201020          DB      7FH, 20H, 10H, 20H, 7FH     ;M
125A 7F
125B 7F300806          DB      7FH, 30H, 08H, 06H, 7FH     ;N
125F 7F
 260 7F414141          DB      7FH, 41H, 41H, 41H, 7FH     ;O
 264 7F
1265 7F484848          DB      7FH, 48H, 48H, 48H, 78H     ;P
1269 78
```

```
126A 7F414543          DB      7FH,41H,45H,43H,7FH     ;Q
126E 7F
126F 7F484C4A          DB      7FH,48H,4CH,4AH,79H     ;R
1273 79
1274 32494949          DB      32H,49H,49H,49H,26H     ;S
1278 26
1279 40407F40          DB      40H,40H,7FH,40H,40H     ;T
127D 40
127E 7F010101          DB      7FH,01H,01H,01H,7FH     ;U
1282 7F
1283 700C030C          DB      70H,0CH,03H,0CH,70H     ;V
1287 70
1288 7F020402          DB      7FH,02H,04H,02H,7FH     ;W
128C 7F
128D 63140814          DB      63H,14H,08H,14H,63H     ;X
1291 63
1292 60100F10          DB      60H,10H,0FH,10H,60H     ;Y
1296 60
1297 43454951          DB      43H,45H,49H,51H,61H     ;Z
129B 61

                ; THE FOLLOWING SUBROUTINES ARE SUBSTITUTES
                ; FOR MONITOR ROUTINES.  NORMALLY THE
                ; CORRESPONDING ROUTINES IN MONITOR 2.1
                ; WOULD BE USED.

                ; OUTPUT A CHARACTER TO THE LIST DEVICE

129C 3E02       LO:     MVI     A,2
129E D30A       LO1:    OUT     0AH
12A0 DB01       LO2:    IN      1
12A2 E604               ANI     4
12A4 C2A012             JNZ     LO2
12A7 78                 MOV     A,B
12A8 EEFF               XRI     0FFH
12AA D308               OUT     8
12AC AF                 XRA     A
12AD D30A               OUT     0AH
12AF C9                 RET
                ; OUTPUT A CHARACTER TO THE PUNCH

12B0 3E01       PO:     MVI     A,1
12B2 C39E12             JMP     LO1

                ; INPUT A CHARACTER FROM THE CONSOLE

 2B5 DB01       CI:     IN      1
12B7 E601               ANI     1
12B9 C2B512             JNZ     CI
```

```
12BC DB00               IN      0
12BE EEFF               XRI     0FFH
12C0 C9                 RET

                ; INCREMENT H AND L REGISTERS

12C1 2C         INCHL:  INR     L
12C2 C0                 RNZ
12C3 24                 INR     H
12C4 C9                 RET

                ; TEMPORARY STORAGE

12C5            BADR:   DS      2       ;BUFFER ADDR
12C7            CADR:   DS      2       ;CHAR ADDR
12C9            GADR:   DS      2       ;GRAF ADDR
12CB            BUFF:   DS      40      ;BUFFER
0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB12

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000

(use additional sheets if necessary)

Program Title

Tape Labeler for MDS

Function

Responds to "O" command in MDS monitor and allows user to insert a label on a paper tape he or she may be generating.

Required Hardware

MDS-800, punch, and a PROM resident monitor

Required Software

See listing

Input Parameters

.O<TEXT FOR LABEL>,

Output Results

Text is punch in large character on paper tape

Registers Modified: All	**Assembler/Compiler Used:** MAC80
RAM Required: Amount to store text	**Programmer:** Matt Townsend
ROM Required: 512 bytes	**Company:**
Maximum Subroutine Nesting Level:	**Address:**

98-034C

```
                        ;REF. NO. AB12
                        ;PROGRAM TITLE TAPE LABELER FOR MDS
                        ;
                        ;
                        ;
                        ;MDS MONITOR CALL ENTERED VIA THE "O" COMMAND THAT WILL
                        ;PROVIDE A LABEL FOR PAPER TAPES. REQUIRES A CHANGE IN
                        ;THE MONITOR'S "CTBL" TO SET THE "O" COMMAND JUMP ADDRRE
                        ;FROM AN ERROR CONDITION TO THE DATA 00F6.
                        ;THIS PROGRAM IS ENTERED FROM THE START ROUTINE IN THE
                        ;MONITOR, WILL ACCEPT CHARACTERS, THEN, UPON RECEIVING
                        ;A <CR-LF> WILL PUNCH THE TITLE TO PTP. THIS PROGRAM
                        ;UTILIZES MDS SUBROUTINES TO DO THE DIRTY WORK SO I'LL
                        ;START BY DEFINING THEM
FEC0                    LEAD EQU 0FEC0H         ;LEADER PUNCH
F80F                    PUN  EQU 0F80FH         ;LISTER OUTPUT
FCA2                    CI   EQU 0FCA2H         ;CONSOLE INPUT
FD3A                    CO   EQU 0FD3AH         ;CONSOLE OUTPUT
3000                    ORG 3000H
                        ;
                        ;NITTY GRITTY STARTS HERE
                        ;
3000 21FF3D             LABEL:  LXI H,3DFFH
3003 1600                       MVI D,00H       ;SET UP CHARACTER COUNTER
3005 CDA2FC             PCMD:   CALL CI         ;GO GET A CHARACTER
3008 4F                         MOV C,A
3009 CD3AFD                     CALL CO         ;ECHO IT
300C FE0D                       CPI 0DH         ;IS IT A CARR. RET?
300E CA1730                     JZ BANN         ;Z=YES ,GO PUNCH THE CHARS.
3011 14                         INR D           ;INCR THE CHAR CNTR
3012 71                         MOV M,C
3013 2B                         DCX H           ;SAVE THE CHAR IN C REG.
3014 C30530                     JMP PCMD        ;GO FOR MORE
3017 0E0A               BANN:   MVI C,0AH       ;LF TO CONSOLE
3019 CD3AFD                     CALL CO
301C CDC0FE                     CALL LEAD
301F 21FF3D                     LXI H,3DFFH
3022 1E0A                       MVI E,0AH
3024 4E                 BAN1:   MOV C,M         ;FETCH 1ST CHAR
3025 CD3F30                     CALL BANP       ;PUNCH IT
3028 15                         DCR D           ;DECR CHAR CNTR
3029 CA3B30                     JZ EXT          ;NZ=GOT MORE CHARS
302C 2B                         DCX H           ;POINT TO MEM LOC
302D 1D                         DCR E
302E C22430                     JNZ BAN1
3031 0E0D                       MVI C,0DH
3033 CD0FF8                     CALL PUN
3036 1E0A                       MVI E,0AH
3038 C32430                     JMP BAN1
```

```
303B CDC0FE             EXT:    CALL LEAD       ;PUNCH TRAILER
303E C9                         RET             ;BACK TO MONITOR
                        ;PUNCHES AN ASCII CHARACTER AS A BIG ONE ON PAPER
                        ;TAPE ALLOWING MDS TAPES TO BE NAMED. ENTERS WITH THE
                        ;CHARACTER IN C REG. USES MDS PUNCH SUBROUTINE TO THE
                        ;TTY.
                        ;
303F E5                 BANP:   PUSH H
3040 D5                         PUSH D
3041 C5                         PUSH B
3042 79                         MOV A,C
3043 E63F                       ANI 3FH
3045 4F                         MOV C,A
3046 07                         RLC
3047 07                         RLC             ;MULT BY FOUR
3048 81                         ADD C           ;MULT BY 5
3049 216930                     LXI H,PTBL-5    ;LOAD SRT LOC OF CHAR TBL
304C D25030                     JNC BAN
304F 24                         INR H           ;INCR HIGH M LOC
3050 5F                 BAN:    MOV E,A
3051 1600                       MVI D,00H
3053 19                         DAD D           ;SET H AND L TO TBL LOC
3054 0605                       MVI B,05H       ;SET UP TABLE CNTR
3056 4E                 PBAN:   MOV C,M         ;FETCH TABLE CHAR
3057 CD0FF8                     CALL PUN
305A 23                         INX H
305B 05                         DCR B
305C C25630                     JNZ PBAN        ;Z=END OF PUNCH
305F 0E00                       MVI C,00H
3061 CD0FF8                     CALL PUN        ;PUNCH 3 SPACES
3064 CD0FF8                     CALL PUN
3067 CD0FF8                     CALL PUN
306A C1                         POP B
306B D1                         POP D
306C E1                         POP H
306D C9                         RET
                        ;BANNER PUNCH CHARACTER TABLE
306E FC121112           PTBL:   DB 374Q,22Q,21Q,22Q,374Q        ;A
3072 FC
3073 FF898989                   DB 377Q,211Q,211Q,211Q,166Q     ;B
3077 76
3078 7E8181C9                   DB 176Q,201Q,201Q,201,102Q      ;C
307C 42
307D FF818181                   DB 377Q,201Q,201Q,201Q,176Q     ;D
3081 7E
3082 FF898989                   DB 377Q,211Q,211Q,211Q,201Q     ;E
3086 81
3087 FF090909                   DB 377Q,11Q,11Q,11Q,1Q          ;F
308B 01
308C 7E81B191                   DB 176Q,201Q,261Q,221Q,162Q     ;G
```

```
3090 72
3091 FF080808                  DB 377Q,10Q,10Q,10Q,377Q        ;H
3095 FF
3096 00C9FFC9                  DB 0Q,201,377Q,201,0Q           ;I
309A 00
309B 4080C97F                  DB 100Q,200Q,201,177Q,1Q        ;J
309F 01
30A0 FF081C22                  DB 377Q,10Q,34Q,42Q,301Q        ;K
30A4 C1
30A5 FF808080                  DB 377Q,200Q,200Q,200Q,200Q     ;L
30A9 80
30AA FF020402                  DB 377Q,2Q,4Q,2Q,377Q           ;M
30AE FF
30AF FF040820                  DB 377Q,4Q,10Q,40Q,377Q         ;N
30B3 FF
30B4 7E818181                  DB 176Q,201Q,201Q,201Q,176Q     ;O
30B8 7E
30B9 FF090909                  DB 377Q,11Q,11Q,11Q,6Q          ;P
30BD 06
30BE 7E8181A1                  DB 176Q,201Q,201Q,241Q,176Q     ;Q
30C2 7E
30C3 FF091929                  DB 377Q,11Q,31Q,51Q,306Q        ;R
30C7 C6
30C8 86898989                  DB 206Q,211Q,211Q,211Q,161Q     ;S
30CC 71
30CD 0101FF01                  DB 1Q,1Q,377Q,1Q,1Q             ;T
30D1 01
30D2 7F808080                  DB 177Q,200Q,200Q,200Q,177Q     ;U
30D6 7F
30D7 1F608060                  DB 37Q,140Q,200Q,140Q,37Q       ;V
30DB 1F
30DC FF402040                  DB 377Q,100Q,40Q,100Q,377Q      ;W
30E0 FF
30E1 E31C081C                  DB 343Q,34Q,10Q,34Q,343Q        ;X
30E5 E3
30E6 0304F804                  DB 3Q,4Q,370Q,4Q,3Q             ;Y
30EA 03
30EB C1A19189                  DB 301Q,241Q,221Q,211Q,207Q     ;Z
30EF 87
30F0 00FF8181                  DB 0Q,377Q,201Q,201Q,0Q         ;LFT BRACKET
30F4 00
30F5 03041860                  DB 3Q,4Q,30Q,140Q,200Q   ;/
30F9 80
30FA 008181FF                  DB 0Q,201Q,201Q,377Q,0Q         ;RIGHT BRACKET
30FE 00
30FF 0402FF02                  DB 4Q,2Q,377Q,2Q,4Q             ;UP ARROW
3103 04
3104 2070A820                  DB 40Q,160Q,250Q,40Q,40Q        ;LEFT ARROW
3108 20
3109 00000000                  DB 0,0,0,0,0                    ;BUMMER
```

```
310D 00
310E 00DFDF00          DB 0Q,337Q,337Q,0Q,0Q          ;QUOTE
3112 00
3113 00070007          DB 0Q,7Q,0Q,7Q,0Q              ;"
3117 00
3118 24FF24FF          DB 44Q,377Q,44Q,377Q,44Q       ;#
311C 24
311D 8689FF89          DB 206Q,211Q,377Q,211Q,161Q    ;$
3121 71
3122 836318C4          DB 203Q,143Q,30Q,304Q,303Q     ;%
3126 C3
3127 609699A6          DB 140Q,226Q,231Q,246Q,340Q    ;&
312B E0
312C 00000700          DB 0,0,7,0,0                   ;'
3130 00
3131 003C4281          DB 0Q,74Q,102Q,201Q,0Q   ;(
3135 00
3136 0081423C          DB 0Q,201Q,102Q,74Q,0Q          ;)
313A 00
313B 894AFF4A          DB 211Q,112Q,377Q,112Q,211Q    ;*
313F 89
3140 08087E08          DB 10Q,10Q,176Q,10Q,10Q         ;+
3144 08
3145 B0700000          DB 260Q,160Q,0Q,0Q,0Q           ;,
3149 00
314A 08080808          DB 10Q,10Q,10Q,10Q,10Q   ;-
314E 08
314F 60600000          DB 140Q,140Q,0Q,0Q,0Q           ;.
3153 00
3154 80601804          DB 200Q,140Q,30Q,4Q,3Q          ;LEFT SLASH
3158 03
3159 7E818181          DB 176Q,201Q,201Q,201Q,176Q     ;0
315D 7E
315E 0082FF80          DB 0Q,202Q,377Q,200Q,0Q         ;1
3162 00
3163 C2A19189          DB 302Q,241Q,221Q,211Q,206Q     ;2
3167 86
3168 42898989          DB 102Q,211Q,211Q,211Q,166Q     ;3
316C 76
316D 1F10FC10          DB 37Q,20Q,374Q,20Q,20Q         ;4
3171 10
3172 4F898989          DB 117Q,211Q,211Q,211Q,161Q     ;5
3176 71
3177 7E898989          DB 176Q,211Q,211Q,211Q,160Q     ;6
317B 70
317C C1211109          DB 301Q,41Q,21Q,11Q,7Q          ;7
3180 07
3181 76898989          DB 166Q,211Q,211Q,211Q,166Q     ;8
3185 76
3186 46898989          DB 106Q,211Q,211Q,211Q,176Q     ;9
```

```
318A 7E
318B A6A60000                   DB 2460,2460,00,00,00           ;:
318F 00
3190 B6760000                   DB 2660,1660,00,00,00           ;;
3194 00
3195 08102442                   DB 100,200,440,1020,1010        ;<
3199 41
319A 24242424                   DB 440,440,440,440,440          ;=
319E 24
319F 81422410                   DB 2010,1020,440,200,100        ;>
31A3 08
31A4 02B90909                   DB 20,2710,110,110,60           ;?
31A8 06
0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

intel

Ref. No. AB13

☐ 4004 ☐ 4040 ☐ 8008 ☑ 8080 (use additional sheets if necessary)

Program Title

Page listing program

Function

Provide facility for listing information in a pagenated, numbered format. This is accomplished thru the system software with the console printer.

Required Hardware

8080 Development System, Console device.

Required Software

Monitor (Ver 2.0), tape to be listed.

Input Parameters

The desired heading is typed on the console. (First page foot, then mark, then actual heading. Rubout echoes and deletes previous character, control 'I' is horizontal tab, type page #1 followed by a control 'P'. Include spaces for necessary decimal carry digits. A control 'D' terminates entry. Then the reader device supplies the body text. Finish by typing 'LC' until bottom of last page is marked by an extra heading.

Output Results

The desired formated listing is obtained.

Registers Modified:	**Assembler/Compiler Used:** 8080 Resident Assembler
RAM Required: CDH Bytes	**Programmer:** Michael Christian
ROM Required: Monitor I/O Routines	**Company:** ARION CORPORATION
Maximum Subroutine Nesting Level: Three levels	**Address:** 825 Boone Avenue North Minneapolis, MN 55427

```
                        ;REF NO.  AB13
                        ;PROGRAM NAME LISTING PROGRAM
                        ;
                        ;
                        ;
                        ;
                        ; LINKAGE ADDRESSES:
3803                    CI      EQU     3803H
3806                    RI      EQU     3806H
3809                    CO      EQU     3809H
                        ; ASCII CONSTANTS:
000D                    CR      EQU     0DH
000A                    LF      EQU     0AH
007F                    RUBOUT  EQU     7FH
                        ;
                        ; MAIN PROGRAM:
                        ;
0040                            ORG     40H             ; RE-ENTRY POINT, OR TO
0040 C32702                     JMP     THEAD           ; RE-ENTRY POINT, OR TO
                                                        ;   USE DEFALT HEADING.
 200                            ORG     200H
                        ;
                        ; NORMAL ENTRY POINT OF PROGRAM:
                        ;
0200 219E02             BEGIN:  LXI     H,MHEAD
0203 CD4002                     CALL    PRINT           ; PRINT HEADING MESSAGE
0206 21C102                     LXI     H,HEAD
0209 1600                       MVI     D,0
020B CD0338             B0:     CALL    CI
020E E67F                       ANI     RUBOUT          ; TO REMOVE PARITY BIT
0210 FE04                       CPI     'D'-'@'         ; CONTROL D = FINISHED
0212 CA2302                     JZ      B1              ; IF HEADING COMPLETE
0215 FE7F                       CPI     RUBOUT
0217 CA9002                     JZ      RUB             ; DELETE LAST CHARACTOR TYPED
021A 23                         INX     H
021B 77                         MOV     M,A             ; SAVE CHARACTOR
021C CD6502                     CALL    PRT             ; ECHO CHARACTOR
021F 14                         INR     D
0220 C30B02                     JMP     B0
0223 21C102             B1:     LXI     H,HEAD
0226 72                         MOV     M,D             ; SAVE HEADING LENGTH
0227 0642               THEAD:  MVI     B,66            ; SET # OF LINES/PAGE
0229 21C102                     LXI     H,HEAD
022C CD4002                     CALL    PRINT
022F CD0638             READ:   CALL    RI
 232 DA2F02                     JC      READ            ; IF NOT RECEIVED, TRY AGAIN
 235 E67F                       ANI     RUBOUT          ; REMOVE PARITY BIT
0237 CD6502             SEND:   CALL    PRT             ; PRINT BODY CHARACTOR
023A DA2702                     JC      THEAD           ; IF PAGE FULL, PRINT HEADING
```

```
023D C32F02             JMP     READ
                ;
                ; SUBROUTINE:
                ;
                ; PRINT MESSAGE, STARTING AT <HL>.  FIRST BYTE IS LENGTH:
0240 56         PRINT:  MOV     D,M
0241 23          PRO:   INX     H
0242 7E                 MOV     A,M
0243 CD6502             CALL    PRT             ; PRINT CHARACTOR
0246 E5                 PUSH    H
0247 FE10               CPI     'P'-'@'         ; CONTROL 'P'= INCREMENT PAGE #
0249 CC5202             CZ      INCPN
024C E1                 POP     H
024D 15                 DCR     D
024E C24102             JNZ     PRO             ; IF NOT FINISHED
0251 C9                 RET
                ;
                ; INCREMENT ASCII NUMBER STORE IN STRING
                ;   IMMEDIATELY PRECEDING <HL> ADDRESS.
0252 2B         INCPN:  DCX     H
0253 3E56               MVI     A,' '
 255 BE                 CMP     M
0256 C25B02             JNZ     $+5
0259 3630               MVI     M,'0'           ; SUBSITIUTE FOR LEADING BLANK
025B 34                 INR     M
025C 3E39               MVI     A,'9'
025E BE                 CMP     M               ; TEST FOR CARRY
025F F0                 RP                      ; IF ALL DONE
0260 3630               MVI     M,'0'
0262 C35202             JMP     INCPN           ; TO PROCESS CARRY
                ;
                ; PRINT CHARACTOR RECEIVED IN <A>:
                ;   IF <CONTROL I>, EXECUTE HORIZONTAL TAB.
                ;   IF <CR>, ALSO RESET TAB COLUMN COUNTER (E).
                ;   IF <LF>, ALSO DECREMENT LINS COUNTER (B),
                ;     RETURNING CARRY SET IF 'BOTTOM OF PAGE'.
0265 FE09       PRT:    CPI     'I'-'@'         ; CONTROL 'I'= HORIZONTAL TAB
0267 4F                 MOV     C,A
0268 CD0938             CALL    CO
026B 79                 MOV     A,C
026C FE0D               CPI     CR
026E C27302             JNZ     $+5
0271 1E00               MVI     E,0             ; RESET COLUMN COUNTER
0273 FE0A               CPI     LF
0275 C27B02             JNZ     $+6
0278 05                 DCR     B               ; ONE LINE PRINTED
 279 37                 STC
 27A C8                 RZ                      ; IF BOTTOM OF PAGE
027B FE20               CPI     ' '
027D FA8102             JM      $+4             ; IF NON-PRINTING CHARACTOR
```

```
0280 1C                 INR     E               ; TO COUNT COLUMN PRINTED
0281 A7                 ANA     A               ; CLEAR CARRY
0282 C9                 RET
0283 0E20       TAB:    MVI     C,' '
0285 CD0938             CALL    CO
0288 1C                 INR     E
0289 3E07               MVI     A,7
028B A3                 ANA     E
028C C28302             JNZ     TAB             ; IF NOT DONE
028F C9                 RET
0290 7A         RUB:    MOV     A,D
0291 A7                 ANA     A
0292 CA0B02             JZ      B0              ; IF NUL STRING
0295 7E                 MOV     A,M
0296 CD6502             CALL    PRT             ; ECHO CHARACTOR REMOVED
0299 15                 DCR     D
029A 2B                 DCX     H
029B C30B02             JMP     B0              ; GET NEXT CHARACTOR
                ;
                ; MESSAGES:
                ;
 ?9E 220D0A54   MHEAD:  DB      L1,CR,LF,'TYPE PAGE HEADING:'
02A2 59504520
02A6 50414745
02AA 20484541
02AE 44494E47
02B2 3A

02B3 0D0A3C42           DB      CR,LF,'<BODY END>',CR,LF
02B7 4F445920
02BB 454E443E
02BF 0D0A

0022            L1      EQU     $-MHEAD-1

02C1 0A0A0A0A   HEAD:   DB      L2,LF,LF,LF,LF, ; (DEFALT HEADING)
02C5 0A

02C6 2E0D0A0A           DB      '. ',CR,LF,LF,LF,LF
^?CA 0A0A

000A            L2      EQU     $-HEAD-1
0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB14

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

SOURCE PAPER TAPE TO MAGNETIC CASSETTE

Function

Will copy a source paper tape onto a magnetic cassette. End statement must be followed by a carriage return. Program will ignore leading blanks.

Required Hardware

TI Silent 700 Terminal connected in place of TTY.

Required Software

Supplied with Silent 700 terminal. Also see User's Library program Ref. No. AC8

Input Parameters

None

Output Results

Paper tape copied on magnetic cassette A X off character follows the end statement.

Registers Modified: None	**Assembler/Compiler Used:** 8080 Macro Ver 3.0
RAM Required: 157 decimal	**Programmer:** John H. Brandt
ROM Required: None	**Company:** E-Systems, Inc.
Maximum Subroutine Nesting Level: 8 bytes	**Address:** Box 1056, Dept. 54221 Greenville, Texas 75401

```
                ;REF NO AB14.
                ;PROGRAM NAME SOURCE PAPER TAPE TO MAGNETIC CASSETTE
                ;
                ;
                ;
                ;
                ;       SOURCE PAPER TAPE TO MAGNETIC CASSETTE
                ;       END STATEMENT IS FOLLOWING BY XOFF
                ;       PROGRAM IGNORES LEADING BLANKS
                ;
                ;
3806            RI      EQU     3806H           ;READER ENTRY POINT
3763            TO      EQU     3763H           ;TAPE OUTPUT ENTRY POINT
0013            XOFF    EQU       13H
000D            CR      EQU       0DH
0044            DD      EQU       44H
0045            EE      EQU       45H
004E            N       EQU       4EH
0020            SPA     EQU       20H
3D25            DELAY   EQU     3D25H           ;DELAY ENTRY POINT
0100                    ORG     100H
0100 2E00               MVI     L,00H
0102 1600               MVI     D,00H
0104 62                 MOV     H,D
0105 CD0638     START:  CALL    RI
0108 DA0501             JC      START
010B 4F                 MOV     C,A
010C 7A                 MOV     A,D
010D FE01               CPI     01H
010F C21A01             JNZ     COMCK
0112 79                 MOV     A,C
0113 FE0D               CPI     CR
0115 C26D01             JNZ     TAPE
0118 1600               MVI     D,00H
011A 79         COMCK:  MOV     A,C
011B FE3B               CPI     3BH             ;COMMENT DELIMITER
011D CA5B01             JZ      FLSET
0120 7C                 MOV     A,H
0121 FE01               CPI     01H
0123 C26001             JNZ     SPCK
0126 79                 MOV     A,C
0127 FE0D               CPI     CR
0129 CA6B01             JZ      SPEL
012C FE45               CPI     EE
012E C26D01             JNZ     TAPE
0131 CD6337             CALL    TO
 134 CD0638             CALL    RI
0137 4F                 MOV     C,A
0138 FE4E               CPI     N
```

```
013A C26D01             JNZ     TAPE
013D CD6337             CALL    TO
0140 CD0638             CALL    RI
0143 4F                 MOV     C,A
0144 FE44               CPI     DD
0146 C26D01             JNZ     TAPE
0149 CD6337             CALL    TO
014C CD0638             CALL    RI
014F 4F                 MOV     C,A
0150 FE0D               CPI     CR
0152 C26D01             JNZ     TAPE
0155 0E13               MVI     C,XOFF
0157 CD6337             CALL    TO
015A 76                 HLT
015B 1601       FLSET:  MVI     D,01H
015D C36D01             JMP     TAPE
0160 79         SPCK:   MOV     A,C
0161 FE20               CPI     SPA
0163 C26D01             JNZ     TAPE
0166 2601               MVI     H,01H
0168 C36D01             JMP     TAPE
 16B 2600       SPEL:   MVI     H,00H
016D 79         TAPE:   MOV     A,C
016E FE0D               CPI     CR
0170 CA7A01             JZ      RESET
0173 2C                 INR     L
0174 CD6337             CALL    TO
0177 C30501             JMP     START
017A 7D         RESET:  MOV     A,L             ;CHECK FOR SHORT LINE
017B D619               SUI     25D
017D FA8801             JM      DELY            ;ADD DELAY
0180 2E00               MVI     L,00H
0182 CD6337             CALL    TO
0185 C30501             JMP     START
0188 2F         DELY:   CMA
0189 C601               ADI     01H
018B 2E0D               MVI     L,0DH
018D CD253D             CALL    DELAY
0190 2D                 DCR     L
0191 C28D01             JNZ     $-4
0194 3D                 DCR     A
0195 C28B01             JNZ     DELY+3
0198 2E00               MVI     L,00H
019A CD6337             CALL    TO
019D C30501             JMP     START
0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB15

☐ 4004 ☐ 4040 ☒ 8008 ☐ 8080

(use additional sheets if necessary)

Program Title

I Command (INSERT DATA in hexadecimal form from the TTY into RAM)

Function

This program loads hexadecimal code into sequential RAM locations beginning at the address specified. An upper address limit may also be specified, if desired. It is useful for loading hex machine code directly into RAM for corrections, debugging, execution, or PROM programming.

Required Hardware

TTY (Intellec 8/Mod 8 configuration uses port 0, 1 and 8)

Required Software

INTELLEC 8/Mod 8 MONITOR, VERSION 2.1
Modified by changing two bytes in the command brand table as follows:
address 3885_H was 43_H change to $B0_H$
address 3886_H was $3C_H$ change to $3F_H$

Input Parameters

SEE ATTACHMENT

Output Results

Registers Modified: All:A,B,C,D,E,H,L	**Assembler/Compiler Used:** INTELLEC 8/MOD8
RAM Required: (03,04,05,08,09,0A,0B) plus bytes to be loaded by program	**Programmer:** G. Bechthold
ROM Required: (eg 3FB0 - 3FFF)	**Company:** National Research Council Radio and Electrical Eng. Div.
Maximum Subroutine Nesting Level: 3 levels	**Address:** Montreal Road Ottawa, Ontario K1A 0R8

98-034B

ATTACHMENT Input Parameters

FORMAT I low address, high address CR (or ,)
or I low address, CR (or ,)

xxxx (sp) HH (sp) HH (sp) HH (sp) ....

Note:
1. A space (sp) may be substituted for the , when entering commands
2. Parameters with underscore are printed by the I routine
3. XXXX - is the address of the byte beginning each line (similar to the D command DISPLAY DATA)
4. HH - two ASCII characters, entered via the TTY, converted to hexadecimal values and placed in a byte of memory
5. The high address in the command may be used when loading RAM to prevent overwriting data above that address. If it is omitted data may be written from the low address to the top of available RAM. Entry in either mode may be terminated by typing CR or (sp)
or ,

Error Conditions

1. If the low address or high address is greater than 16 bits, only the last 4 hex digits of the argument will be used as the address
2. If the low address is greater than the high address, only one byte at the low address will be inserted.
3. If the low address, high address or data contains an invalid character, or if the the high address is omitted and two delimiters (, or (sp) followed by , or (sp) or CR) do not follow the low address the monitor will immediately type * (CR)(1_F) and await the next command.

ATTACHMENT Output Results

The hexadecimal address of the first memory location to be loaded is printed followed by a space. Two hexadecimal characters may then be typed and will be loaded into a single byte of memory, followed by an automatic space. This procedure continues, loading successive memory addresses, until a maximum of 16 bytes have been loaded. An automatic CR, LF, and the address of the first byte of this new line will then be printed. All lines are blocked into integral multiples of 16, as in the Display Data command, so the first and last lines may be less than 16 bytes in order to synchronize the display. Data entry continues until a CR is typed, the high address (if specified following the I command) is loaded, or the top address of available RAM is reached. The Insert operation is then terminated and control returned to the monitor.

```
                ;REF. NO. AB15.
                ;PROGRAM (INSERT DATA IN HEXADECIMAL FORM FROM THE TTY INTO RAM)
                ;
                ;
                ;
                ;
                ;INSERT DATA IN HEXADECIMAL FORM
                ;FROM THE TTY INTO RAM.
                ;
3FB0                    ORG     3FB0H
3FB0 CD803D     INSRT:  CALL    3D80H           ;EXP-ENTER ADDRESS LIMITS
3FB3 CDC73C             CALL    3CC7H           ;CR/LF
3FB6 CDCF3D             CALL    3DCFH           ;GETAD-RETRIVE ADDRESS
3FB9 AF                 XRA     A               ;CHECK IF UPPER LIMIT
3FBA 92                 SUB     D               ;OF ADDRESS =0 (ADDRESS
3FBB 9B                 SBB     E               ;NOT SPECIFIED)
3FBC CAF33F             JZ      UPLIM           ;YES
3FBF CDFB3D             CALL    3DFBH           ;LADR-PRINT ADDRESS
3FC2 CD4B3C     NXTBY:  CALL    3C4BH           ;BLANK
3FC5 CD443F             CALL    3F44H           ;RI-READ IN 1ST. CHAR.
3FC8 CD533E             CALL    3E53H           ;NIBBLE-CONV. TO HEX, CHECK ILLE
3FCB DA433C             JC      3C43H           ;LER-PRINT * RET. TO MONITOR
3FCE 07                 RLC
3FCF 07                 RLC
3FD0 07                 RLC
3FD1 07                 RLC
3FD2 4F                 MOV     C,A             ;TEMP. STORE 1ST. CHAR.
3FD3 CD443F             CALL    3F44H           ;RI-READ IN 2ND. CHAR.
3FD6 CD533E             CALL    3E53H           ;NIBBLE-CONV. TO HEX, CHECK ILLE
3FD9 DA433C             JC      3C43H           ;LER-PRINT * RET. TO MONITOR
3FDC B1                 ORA     C
3FDD CDCF3D             CALL    3DCFH           ;GETAD-RETRIVE ADDRESS
3FE0 77                 MOV     M,A             ;STORE BYTE
3FE1 CDDD3D             CALL    3DDDH           ;HIL0-CHECK UPPER LIMIT
3FE4 DA4438             JC      3844H           ;YES. RET. TO MONITOR
3FE7 CD2F3F             CALL    3F2FH           ;SAVIT-STORE ADDRESS
3FEA 79                 MOV     A,C             ;CHECK FOR COMPLETION OF
3FEB E60F               ANI     0FH             ;LINE (16 BYTES)
3FED C2C23F             JNZ     NXTBY           ;GET NEXT TWO CHAR.
3FF0 C3B33F             JMP     INSRT+3         ;NEW LINE. PRINT ADDRESS
3FF3 CD3E3E     UPLIM:  CALL    3E3EH           ;MEMCK-FIND UPPER LIM. RAM
3FF6 210A00             LXI     H,000AH         ;LOAD UPPER LIMIT
3FF9 71                 MOV     M,C             ;LOCATIONS OF THE
3FFA 2C                 INR     L               ;I PROGRAM
3FFB 77                 MOV     M,A
3FFC C3B63F             JMP     INSRT+6         ;PRINT ADDRESS & BEGIN
 000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AA2

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 (use additional sheets if necessary)

Program Title

8080 RAM MEMORY TEST

Function

MEMORY TEST for Intellec/80 system

Required Hardware

Intellec/80 I/O board (TTY data output port ØØH)
(TTY status input port Ø1H)
(DISPLAY output port FFH)

Required Software

None

Input Parameters

TTY may be disabled (for long test periods) by typing 2 successive keys (causing over-run error) and turning it off. Assembler Values "TOP"and "BOTTOM" define the limits of the RAM addresses under test (normally 0000H to 2000H).

Output Results

Four types of error messages can occur (if the TTY is enabled):

1) Instead of 67H, 65H was written into address 1234H;
-67-65-1234 (02H shown on front panel display)

2) Instead of 67H, 65H was read from address 1234H;
-67-67-1234 (02H shown on front panel display)

3) Address 1234 not in RAM under test;
X-55-FF-1234 (55H shown on front panel display)

4) 31H was written into address 1234 while addressing another address in memory (i.e. "Cross-Write error")
X-55-31-1234 (55H shown on front panel display)

NOTE---The error bit pattern will always be displayed.

Registers Modified:	Assembler/Compiler Used:
ALL	MAC80 or Resident Assembler
RAM Required:	**Programmer:**
NONE (except RAM under test)	B. Searle
ROM Required:	**Company:**
256 Bytes (1 chip)	Ministry of Transport (CANADA)
Maximum Subroutine Nesting Level:	**Address:** T.A.C.D., Floor 10-C Tower "C", Place De Ville OTTAWA, CANADA
No stack used	

98-034B

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AA3

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 (use additional sheets if necessary)

Program Title

Memory Diagnostic Program

Function

Writes test bytes in any range of memory and compares the written bit combination with what is read. Upon detection of a defective memory location, an error message is printed specifying the address, reference and actual values.

Required Hardware

TTY on Port 0 + 1

Required Software

Intellec 80 System Monitor

Input Parameters

Address range input from TTY-console:

<low address>,<high address> <CR>

Delimiter between values = ','

- Closing input parameters with CR
- Parameters in HEX

Output Results

Error message if program finds a defective memory location:

```
ADDR:   REF:   ACT:      (TABLE TITLES)
XXXX    XX     XX        (ERROR MESSAGE)
 .       .      .
 .       .      .
```

Registers Modified:	Assembler/Compiler Used:
All	Intellec 80 Macro Assembler
RAM Required:	**Programmer:**
12H	P. Schneider/H.P. Teufer
ROM Required:	**Company:**
181H	Landis & Gyr AG
Maximum Subroutine Nesting Level:	**Address:** CH-6301, Zug Switzerland

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB16

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 (use additional sheets if necessary)

Program Title

Compare Object Code Tape with Memory

Function

This program extends the Intellec 8 System Monitor's 'Compare" command to check the data from a hex format tape against the current information stored in memory.

Required Hardware

Intellect 8/80 Development System or equivalent (with tape reader)

Required Software

8080 Monitor, Version 2.0 (for other versions, linkage addresses must be corrected)

Input Parameters

To compare Prom, type "C(address)(CR)" (no change)

To compare a tape, ready the tape and type "C(CR)" (omitting the memory address, as the address is specified by the hex format tape.)

Output Results

If all data checks, control is passed back to the Monitor's command loop. If a mismatch is found, it is reported in the following format: (Mem address) (Mem contents) (Compare data)

Also strings of mismatches are separated by a blank in the 'Compare tape' routine only.

Registers Modified:	Assembler/Compiler Used:
All	8080 Macro Assembler
RAM Required:	**Programmer:**
None	Michael Christian
ROM Required:	**Company:**
90H	Arion Corporation
Maximum Subroutine Nesting Level:	**Address:** 825 Boone Avenue North Minneapolis, MN 55427
(not significant)	

```
                ;REF NO. AB16.
                ;PROGRAM NAME COMPAR OBJECT CODE TAPE WITH MEMORY
                ;
                ;
                ;
                ;
                ;
                ; COMPARE OBJECT CODE TAPE WITH MEMORY
                ;                8/20/75
                ;
                ; THIS PROGRAM EXTEND THE MONITOR 'COMPARE'
                ; COMMAND TO CHECK THE DATA FROM A HEX FORMAT
                ; OBJECT CODE TAPE AGAINST THE CURRENT INFORMATION
                ; STORED IN MEMORY, AND REPORTS ANY MISMATCHES
                ; FOUND.
                ;
                ; > TO COMPARE A PROM, TYPE "C<ADDRESS><CR>".
                ; > TO COMPARE A TAPE, LOAD THE TAPE TO BE COMPARED,
                ;     AND TYPE "C<CR>" (OMITTING THE ADDRESS).
                ;
                ;   THE FORMAT OF ANY MISMATCHES FOUND IS:
                ;    <MEM ADDRESS> <MEM CONTENTS> <COMPARE DATA>
                ;
                ; MONITOR (VER 2.0) ADDRESSES REFERENCED -
                ;
395A            COMP    EQU     395AH
3F6E            TI      EQU     3F6EH
3E11            NIBBLE  EQU     3E11H
3D71            EX1     EQU     3D71H
000D            CR      EQU       0DH
3C3E            LER     EQU     3C3EH
3EA0            RI      EQU     3EA0H
3C7D            BYTE    EQU     3C7DH
3CC8            CRLF    EQU     3CC8H
3DBA            LADR    EQU     3DBAH
3DC2            LBYTE   EQU     3DC2H
3C4E            BLK     EQU     3C4EH
3C50            CO      EQU     3C50H
386B            START   EQU     386BH
                ;
                ;*****************************************************
                ;
                ; ENTRY POINT - FIND OUT IF COMPARE IS WITH
                ;   PROM OR WITH TAPE.
                ;
 400                    ORG     3400H   ; OR ANY AVAILABLE ROM ADDRESS
                COMPARE:
3400 0D                 DCR     C       ; GET ONE PARAMETER
```

```
3401 215E39             LXI     H,COMP+4
3404 E5                 PUSH    H           ; SET UP RETURN ADDRESS FROM EXPR
3405 210000             LXI     H,0
3408 CD6E3F             CALL    TI          ; GET FIRST CHARACTOR
340B 47                 MOV     B,A         ;  SAVE TO TEST AS DELIMITER
340C CD113E             CALL    NIBBLE
340F D2713D             JNC     EX1,7       IF HEX DIGIT, REJOIN EXPR CODE

3412 D1                 POP     D           ; BUMP DUMMY RETURN ADDRESS
3413 78                 MOV     A,B
3414 FE0D               CPI     CR
3416 C23E3C             JNZ     LER         ;NOT VALID DELIMITER, ERROR
                ;
                ;*****************************************************
                ;
                ; READ HEX TAPE - GET NEXT BYTE AND TEST MEMORY ADDRESS
                ;    SPECIFIED IN RECORD.  REPORT MISMATCH, IF FOUND.
                ;
3419 E5                 PUSH    H           ; RESET MISMATCH FLAG & ZERO COUNTER
341A CDA03E     RED0:   CALL    RI
341D DA3E3C             JC      LER         ; IF END OF TAPE, ERROR
3420 063A               MVI     B,':'
3422 90                 SUB     B
3423 C21A34             JNZ     RED0        ; LOOP UNTIL START OF RECORD
3426 57                 MOV     D,A         ; ZERO CHECKSUM
3427 CD7D3C             CALL    BYTE
342A CA7B34             JZ      FINISH      ; ZERO RECORD LENGTH, ALL DONE
342D 5F                 MOV     E,A
342E CD7D3C             CALL    BYTE        ; GET MSB OF ADDRESS
3431 F5                 PUSH    PSW
3432 CD7D3C             CALL    BYTE        ; GET LSB OF ADDRESS
3435 E1                 POP     H
3436 6F                 MOV     L,A         ; SET UP STARTING ADDRESS IN HL
3437 CD7D3C             CALL    BYTE        ; RECORD TYPE

343A CD7D3C     RED1:   CALL    BYTE        ; GET DATA
343D BE                 CMP     M
343E CA6034             JZ      MATCH       ; IF GOOD DATA, SKIP REPORT
3441 F5                 PUSH    PSW
3442 CDC83C             CALL    CRLF
3445 CDBA3D             CALL    LADR        ; PRINT ADDRESS OF MISMATCH
3448 CD4E3C             CALL    BLK
344B 7E                 MOV     A,M
344C CDC23D             CALL    LBYTE       ; PRINT STORED DATA
344F CD4E3C             CALL    BLK
3452 F1                 POP     PSW
3453 CDC23D             CALL    LBYTE       ; PRINT READ DATA
 456 E3                 XTHL                ; GET COUNTER AND FLAG
3457 23                 INX     H           ;  COUNT ONE MISMATCH
3458 3E80               MVI     A,80H
```

```
345A B4                 ORA     H
345B 67                 MOV     H,A         SET MISMATCH FLAG
345C E3                 XTHL
345D C36B34             JMP     NEXT
                ;
                ;*****************************************************
                ;
                ;  UPDATE POINTER AND TEST IF END OF ONE RECORD.
                ;
3460 E3         MATCH:  XTHL                ; GET COUNTER AND FLAG
3461 7C                 MOV     A,H
3462 B7                 ORA     A
3463 FCC83C             CM      CRLF        ; MARK END OF MISMATCH STRING
3466 3E7F               MVI     A,7FH
3468 A4                 ANA     H
3469 67                 MOV     H,A         ; CLEAR FLAG
346A E3                 XTHL
346B 23         NEXT:   INX     H           ; POINT TO NEXT MEMORY LOCATION
346C 1D                 DCR     E           ; ONE RECORD READ
346D C23A34             JNZ     RED1        ; IF NOT DONE, GET NEXT BYTE
3470 CD7D3C             CALL    BYTE        ; GET CHECKSUM
3473 CA1A34             JZ      RED0        ; IF OK, GET NEXT RECORD

3476 0E2A               MVI     C,'*'
3478 CD503C             CALL    CO          ; FLAG CHECKSUM ERROR

347B CD4E3C     FINISH: CALL    BLK
347E 2B                 DCX     H
347F CDBA3D             CALL    LADR        ; PRINT LAST ADDRESS CHECKED
3482 CDC83C             CALL    CRLF
3485 E1                 POP     H
3486 3E7F               MVI     A,7FH
3488 A4                 ANA     H
3489 67                 MOV     H,A
348A CDBA3D             CALL    LADR        ; PRINT # OF MISMATCHES
348D C36B38             JMP     START       ; RETURN TO MONITOR
                ;
                ;*************************************************************
                ;
3896                    ORG     3896H
3896 0034               DW      COMPARE ; MODIFY THE MONITOR'S COMMAND BRANCH
                                        ;   TABLE TO JUMP TO NEW ROUTINE.
                ;
                ;*************************************************************
                ;
0000                    END
```

Ref.# AB17

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/4040 ☒ 8008 ☐ 8080 ☐ 8048 ☐ 8085 ☐ Other_____ (use additional sheets if necessary)

Program Title

K, PROGRAM TRAP

Function

This program provides two traps (search/wait) for debugging other programs which use RAM memory. Displays the contents of five registers and the trap address when the trap occurs.

Required Hardware

Intellec 8 computer and TTY on Port 0.

Required Software

Monitor VER. 1.1 as the trap program is written. For Monitor VER. 2.1 change the trap program equates (See the attached sheet).

Input Parameters

See the attached sheet.

Output Results

See the attached sheet.

Registers Modified: A, B, C, D, E, H, L	Programmer: R. T. Kirk
RAM Required: 10H to 1FH	Company: Kenway Incorporated (801)292-2401
ROM Required: 3100 to 323E	Address: 525 W. 350
Maximum Subroutine Nesting Level: 3	City: N. Bountiful
Assembler/Compiler Used: Intellec Assembler Ver. 1.0	State: UTAH 84010

```
;TITLE: K, PROGRAM TRAP
 ; A PROGRAM BREAK POINT OR TRAP IS SET BY THIS PROGRAM.
 ; FOLLOWING POWER ON START, TYPE G3230 CR TO INITIALIZE
 ;     THE TRAP PROGRAM.
 ; THIS PROGRAM CAN ONLY BE USED BY RAM TYPE MEMORY.
 ; TO SET A TRAP TYPE K FOLLOWED BY TRAP ADDRESS AND CR.
 ; TO CLEAR ONE OR TWO TRAPS TYPE K FOLLOWED BY CR.
 ; A MAXIMUM OF TWO TRAPS CAN BE SET. IF THE MAXIMUM IS
 ;     EXCEEDED  TERR WILL BE PRINTED ON THE TTY.
 ; WHEN A TRAP IS ENCOUNTERED THE CONTENTS OF THE A,B,C,
 ;     D,E REGISTERS ARE PRINTED FOLLOWED BY THE TRAP
 ;     ADDRESS.  THE FORMAT IS:  AB CD EØ ADDRESS
 ; THE TRAP IS CLEARED WHEN ENCOUNTERED.
 ; A TRAP REQUIRES 3 MEMORY LOCATIONS. THEREFORE NEVER
 ;     ALLOW THE TRAP TO AFFECT AN ENTRY POINT.
 ;     EXAMPLE: A TRAP CANBE SET AT TR1; MOV A,M, DEC C OR
 ;              JNZ TR1Ø BUT NOT AT TR2: MVI D,0FFH UNLESS
 ;              IT CANBE DETERMINED THAT PROGRM EXECUTION
 ;              WILL NEVER ARRIVE AT TR3 BEFORE THE TRAP.
 ;              IS ENCOUNTERED.
 ;                    TR1:  MOV .  A,M
 ;                          DEC    C
 ;                          JNZ    TR1Ø
 ;                    TR2:  MVI    D,ØFFH
 ;                    TR3:  MVI    E,ØFFH
 ; FOR MONITOR VER 1.1 CHANGE THE CONTENTS OF MEMORY
 ;     LOCATIONS 3C21 AND 3C22 TO 3100 HEX.  FOR MONITOR
 ;     VER 2.1 CHANGE THE CONTENTS OF MEMORY LOCATIONS
 ;     3889 AND 388A TO 3100 HEX.
 ; FOR MONITOR VERSION 2.1 CHANGE THE FOLLOWING EQUATES TO:
 ;     EXPR  EQU  3D8ØH
 ;     INCHL EQU  3DEBH
 ;     START EQU  3844H
 ;     TTY   EQU  3C57H
 ;     VERO  EQU  3835H
 ;
    ;
```

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/4040 ☐ 8008 ☒ 8080 ☐ 8048 ☐ 8085 ☐ Other _____ (use additional sheets if necessary)

Program Title

DEBUG - A two prom debugging package to be used in minimum 8080 systems.

Function

To inspect, dump, move and find data in memory.

Required Hardware

Required Software

Input Parameters

Output Results

SEE WRITEUP

Registers Modified: ALL	Programmer: Tom Lafleur
RAM Required: Stack and one location above stack for flag	Company: Naval Electronics Lab Ctr.,
ROM Required: 2-1702 (512 location)	Address: La Posta Observatory Route 1, Box 591
Maximum Subroutine Nesting Level:	City: Campo
Assembler/Compiler Used: 8080 Macro Assembler	State: CA 92006 (714) 225-7705

WRITEUP

COMMAND SYNTAX

Inspect Memory

I ADDR = XXXX
- Address to inspect

(XXXX) DD XX
- XX: Space bar to go to next location
- XX: New data (two hex charter)
- XX: CTL 'A' (01H) will ask for new address
- DD: Current Data
- (XXXX): Address

Dump Memory in Hex

D ADDR = XXXX, XXXX
- XXXX: Last location
- XXXX: First location

Dump range

Will display 16 memory locations per line. (XXXX) DD DD DD DD DD.

Move Block of Memory

M ADDR = XXXX, XXXX TO XXXX
- XXXX: New address of block
- XXXX: Last address of block
- XXXX: First address of block

Find Byte

F ADDR = XXXX, XXXX, D=DD, N=DD
- N=DD: Carriage return if no replacement
- N=DD: or replacement data
- D=DD: Search data
- XXXX: Last Address of search range
- XXXX: First address of search range

This command will print out the address of all locations that contain the search data. An option is also available that will replace the search data with a new data byte.

GOTO

G ADDR = XXXX
- Jump address

Will exit with interrupt on.

Notes:

1. Under lined data is user entered.
2. Dump and find routine allows for a return to the command loop by pressing any key on the TTY.
3. All routine allows for exit by pressing ALT-MODE (7DN) or break (00H) key in response to any input, will respond with '?'.

Options

Commands are given for jumps to user-written high speed I/O devices. The commands are:

R	(Jump 3500H)	Read hex
W	(Jump 3503H)	Write hex routine
E	(Jump 3506H)	End file

Output Results

First time through debug will give an ID message; then will type out a '?' when it's ready for input.

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AA4

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 (use additional sheets if necessary)

Program Title

PUNCH TEST or TTY READER/PUNCH TEST

Function

1) Tests paper tape punch (using High Speed or TTY Reader)

2) Complete TTY READER/PUNCH TEST

Required Hardware

INTELLEC/80: I/O board for TTY output

Required Software

INTELLEC/80 monitor (or equivalent I/O drivers).

Input Parameters

None.

Output Results

1) Binary count tape (& TTY listing if TTY punch is used).

2) On errors, TTY prints actual & expected data before exiting to monitor

TEST METHOD

Stop CPU and advance reader tape one character to simulate a character error.

Registers Modified:	Assembler/Compiler Used:
ALL	Mac-80 resident or cross Assembler
RAM Required:	**Programmer:**
ø (except stack)	B. Searle
ROM Required:	**Company:**
3F Hex	Ministry of Transport (CANADA)
Maximum Subroutine Nesting Level:	**Address:** TACD, Tower C, Place de Ville
about 4	Ottawa, Ontario, CANADA, K1A 0N8

```
                ;REF.  NO.  AA4.
                ;PROGRAM NAME PUNCH TEST OR TTY READER/PUNCH TEST
                ;
                ;
                ;
                ;
                        TITLE ' ----- RPTEST ----- '
                ;
                ;
                ;*****************************************************
                ;*****************************************************
                ;
                ;       TELETYPE READER/PUNCH TEST
                ;
                ;
                ;       START TEST AT LOCATION 1000 (HEX), THEN
                ;       PUT PUNCHED LEADER IN READER & START
                ;       READER WHILE BINARY COUNT IS PUNCHING
                ;
                ;       IF ERROR OCCURS, THE ACTUAL CHARACTER
                ;       READ IN AND THE EXPECTED CHARACTER ARE
                ;       TYPED OUT.  THE TEST PROGRAM THEN EXITS
                ;       TO THE INTELLEC 80 MONITOR.
                ;
                ;
                ;       AUTHOR:  B.  SEARLE
                ;                MINISTRY OF TRANSPORT
                ;                OTTAWA,  CANADA
                ;                (613) 996-0221
                ;
                ;       DATE:    JANUARY 30, 1975
                ;
                ;*****************************************************************
                ;*****************************************************************
                ;
                ;
                ;
                ;
                ;******* INTELLEC 8/80 MONITOR ENTRY POINTS, VERSION 1.1
                ;                                        ;VERSION 2.0 ENTRY POINT
                ;

3C30                    BLK     EQU     3C30H           ;3CE4
3CAD                    CRLF    EQU     3CADH           ;3CC8
3DB0                    LBYTE   EQU     3DB0H           ;3DC2
 )C0                    LEAD    EQU     3DC0H           ;3DD2
_.66C                   PO      EQU     3E6CH           ;3E78
3E94                    RI      EQU     3E94H           ;3EA0
3800                    MONITR  EQU     3800H           ;3800
```

```
                        ;
                        ;
                        ;
                        ;
                        ;****** "PUNCH" CONDITIONAL ASSEMBLY CONTROL
                        ;
0000                    FALSE   EQU     0000H
00FF                    TRUE    EQU     00FFH
00FF                    PUNCH   EQU     TRUE
                        ;
3640                    ORG     3640H
                        ;
                        ;
                        ;
                        ;****** START UP
                        ;
3640 FB         START:  EI
3641 CDAD3C             CALL CRLF                       ; PRINT CR & LF
                        ;
                        ;
                        ;
                        ;
                        IF PUNCH ;;;;;;
                        ;****** FIRST PUNCH LEADER, SYNC CHARACTER, AND
                        ;****** ONE FULL BINARY COUNT PATTERN
                        ;
3644 CDC03D     INITP:  CALL LEAD                       ; PUNCH LEADER
3647 0EFF               MVI C,00FFH                     ; PUNCH SYNC CHARACTER
3649 CD6C3E             CALL PO
364C 0E00               MVI C,00H                       ; INITIALIZE PUNCH CHARA
364E CD6C3E     INIT1:  CALL PO                         ; PUNCH & INCREMENT CHAR
3651 0C                 INR C
3652 C24E36             JNZ INIT1                       ; LOOP UNTIL FULL COUNT
                        ENDIF ;;;;;;;
                        ;
                        ;
                        ;
                        ;
                        ;****** NOW INITIALIZE READER TAPE POSITION
                        ;
3655 CD943E     INITR:  CALL RI                         ; READ UNTIL SYNC CHARAC
3658 3C                 INR A
3659 C25536             JNZ INITR
365C 5F                 MOV E,A                         ; INITIALIZE COUNT PATTE
                        ;
                        ;
                        ;
                        ;
                        ;****** TEST: READ TAPE & COMPARE TO COUNT PATTERN
                        ;
```

```
365D CD943E    TEST1:   CALL RI                          ; READ NEXT CHARACTER
3660 93                 SUB E                            ; TEST AND BRANCH IF ERR
3661 C46C36             CNZ ERROR
                        IF PUNCH ;;;;;;
3664 4B                 MOV C,E                          ; PUNCH CHARACTER
3665 CD6C3E             CALL PO
                        ENDIF ;;;;;;;
3668 1C                 INR E                            ; LOOP TO NEXT CHARACTER
3669 C35D36             JMP TEST1
                    ;
                    ;
                    ;
                    ;
                    ;****** ERROR PRINT OF ACTUAL & EXPECTED CHARACTER
                    ;
366C 83         ERROR:  ADD E                            ; RESTORE ACTUAL CHARACT
366D CDB03D             CALL LBYTE                       ; PRINT ACTUAL CHARACTER
3670 CD303C             CALL BLK                         ; PRINT BLANK
3673 7B                 MOV A,E                          ; PRINT EXPECTED CHARACT
3674 CDB03D             CALL LBYTE
3677 CDAD3C             CALL CRLF                        ; PRINT CR, LF
                        IF PUNCH ;;;;;;
367A C30038             JMP MONITR                       ; EXITS TC MONITOR
                        ENDIF ;;;;;;;
                                                         ; OR IF PRINT & PUNCH DE
                                                         ; ARE DIFFERENT, COULD .
367D C34036             JMP START                        ; RE-SYNC & CONTINUE TES
                    ;
                    ;
                    ;
                    ;
0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AA5

☐ 4004 ☐ 4040 ☐ 8008 ☑ 8080 (use additional sheets if necessary)

Program Title

READER TEST

Function

Tests High Speed paper tape reader or teletype reader.

Required Hardware

INTELLEC 8/80: I/O board for reader control,
I/O board for TTY output.

Required Software

INTELLEC 8/80 monitor (or equivalent I/O drivers).

Input Parameters

Binary count tape (use a patched-up continuous loop for a continuous test).

Output Results

Actual reader data & expected data are typed out on TTY for any read error, and the input tape is re-synchronized (in case of dropped or extra characters).

TEST METHOD

Stop CPU and advance reader tape one character to simulate a character error.

Registers Modified: ALL	**Assembler/Compiler Used:** Mac-80 resident or cross Assembler
RAM Required: Ø (except stack)	**Programmer:** B. Searle
ROM Required: 25 Hex	**Company:** Ministry of Transport (CANADA)
Maximum Subroutine Nesting Level: about 4	**Address:** TACD, Tower C, Place de Ville OTTAWA, ONTARIO, CANADA K1A ON8

```
                ;REF.  NO.  AA5
                ;PROGRAM TITLE READER TEST
                ;
                ;
                ;
                ;
                        TITLE ' ----- RPTEST ----- '
                ;
                ;
                ;*****************************************************************
                ;*****************************************************************
                ;
                ;       TELETYPE READER PUNCH TEST
                ;
                ;
                ;       START TEST AT LOCATION 1000 (HEX), THEN
                ;       PUT PUNCHED LEADER IN READER & START
                ;       READER WHILE BINARY COUNT IS PUNCHING
                ;
                ;       IF ERROR OCCURS, THE ACTUAL CHARACTER
                ;       READ IN AND THE EXPECTED CHARACTER ARE
                ;       TYPED OUT.  THE TEST PROGRAM THEN EXITS
                ;       TO THE INTELLEC 80 MONITOR.
                ;
                ;
                ;       AUTHOR: S.  SEARLE
                ;               MINISTRY OF TRANSPORT
                ;               OTTAWA, CANADA
                ;               (613) 996-0221
                ;
                ;       DATE: JANUARY 30, 1975
                ;
                ;*****************************************************************
                ;*****************************************************************
                ;
                ;
                ;
                ;
                ;****** INTELLEC 8/80 MONITOR ENTRY POINTS, VERSION 1.1
                ;                                               ;VERSION 2.0 ENTRY POINT
                ;
3C30                    BLK     EQU     3C30H           ;3CE4
3C40                    CRLF    EQU     3C40H           ;3CC8
3DB0                    LBYTE   EQU     3DB0H           ;3DC2
3DC0                    LEAD    EQU     3DC0H           ;3DD2
 E94                    PO      EQU     3E94H           ;3E78
3E94                    RI      EQU     3E94H           ;3EA0
3800                    MONITR  EQU     3800H           ;3800
                ;
```

```
                        ;
                        ;
                        ;
                        ;****** "PUNCH"  CONITIONAL ASSEMBLY CONTROL
                        ;
0000                    FALSE   EQU     0000H
00FF                    TRUE    EQU     00FFH
0000                    PUNCH   EQU     FALSE
                        ;

3600                    ORG     3600H
                        ;
                        ;
                        ;
                        ;****** START UP
                        ;
3600 FB                 START:  EI
3601 CD403C                     CALL    CRLF                    ;PRINT CR & LF
                        ;
                        ;
                        ;
                        ;
                                IF PUNCH ;;;;;;
                        ;****** FIRST PUNCH LEADER, SYNC CHARACTER, AND
                        ;****** ONE FULL BINARY COUNT PATTERN
                        ;
                        INITP:  CALL    LEAD                    ;PUNCH LEADER
                                MVI     C,00FFH                 ;PUNCH SYNC CHARACTER
                                CALL    PO
                                MVI     C,00H                   ;INITIALIZE PUNCH CHARAC
                        INIT1:  CALL    PO                      ;PUNCH & INCREMENT CHARA
                                INR     C
                                JNZ     INIT1                   ;LOOP UNTIL FULL COUNT P
                                ENDIF   ;;;;;;;
                        ;
                        ;
                        ;
                        ;
                        ;****** NOW INITIALIZE READER TAPE POSTION
                        ;
3604 CD943E             INITR:  CALL    RI                      ;READ UNTIL SYNC CHARACT
3607 3C                         INR     A
3608 C20436                     JNZ     INITR
360B 5F                         MOV     E,A                     ;INITIALIZE COUNT PATTER
                        ;
                        ;
                        ;
                        ;
                        ;****** TEST: READ TAPE & COMPARE TO COUNT PATTERN
                        ;
```

```
360C CD943E    TEST1:   CALL    RI                      ;READ NEXT CHARACTER
360F 93                 SUB     E                       ;TEST AND BRANCH IF ERRO
3610 C41736             CNZ     ERROR
                        IF      PUNCH;;;;;;
                        MOV     C,E                     ;PUNCH CHARACTER
                        CALL    PO
                        ENDIF   ;;;;;;;
3613 1C                 INR     E                       ;LOOP TO NEXT CHARACTER
3614 C30C36             JMP     TEST1
                    ;
                    ;
                    ;
                    ;
                    ;****** ERROR PRINT OF ACTUAL & EXPECTED CHARACTER
                    ;
3617 83             ERROR:  ADD     E                   ;RESTORE ACTUAL CHARACTE
3618 CDB03D             CALL    LBYTE                   ;PRINT ACTUAL CHARACTER
361B CD303C             CALL    BLK                     ;PRINT BLANK
361E 7B                 MOV     A,E                     ;PRINT EXPECTED CHARACTE
361F CDB03D             CALL    LBYTE
3622 CD403C             CALL    CRLF                    ;PRINT CR, LF
                        IF      PUNCH;;;;;;
                        JMP     MONITR                  ;EXITS TO MONITOR
                        ENDIF   ;;;;;;;
                                                        ;OR IF PRINT & PUNCH DEV
                                                        ;ARE DIFFERENT, COULD ..
3625 C30036             JMP     START                   ;RE-SYNC & CONTINUE TEST
                    ;
                    ;
                    ;
                    ;
0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AC4

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000

(use additional sheets if necessary)

Program Title

TALLY R2050 HSPTR DRIVER

Function

Extension to the Intel 8080 monitor to handle a Tally model R2050 photoelectric tape reader at 200 cps.

Required Hardware

Intellec 8 Mod 80
Tally model R2050 photoelectric tape reader
Power supply for the reader
Connecting cables

Required Software

Intel 8080 monitor

Input Parameters

None

Output Results

Data read is passed back to the monitor

Registers Modified: Data in A all others, saved & restored	**Assembler/Compiler Used:** 8080 Macro Assembler, V2.0
RAM Required: None	**Programmer:** J.J. deZubeldia
ROM Required: 44 bytes	**Company:** IKASLAN S.A.
Maximum Subroutine Nesting Level: 2	**Address:** Barrio de Zabalondo MUNGUIA (Vizcaya) SPAIN

```
                ;REF NO.  AC4.
                ;PROGRAM NAME TALLY R2050 HSPTR DRIVER
                ;
                ;
                ;
                ;
                ;
                ;
                ;INTEL 8080 MONITOR VER.  1.0
                ;EXTENSION TO HANDLE A TALLY MODEL R 2050
                ;PHOTOELECTRIC TAPE READER AT 200 CPS.
                ;USING THE INTELLEC INPUT PORT #3 AS TAPE
                ;READER DATA INPUT AND BIT #2 (00000100)
                ;ON OUTPUT PORT #1 AS TAPE ADVANCE.
                ;READER OUTPUTS INVERTED DATA,
                ;HANDLER RETURNS NON-INVERTED DATA IN A.
                ;
                ;JJZ 16.4.75
                ;VER.  2.1
                ;
                ;
0003            DIPR    EQU     3       ;DATA INPUT PORT
0001            COPR    EQU     1       ;COMMAND OUTPUT PORT
0004            CBIT    EQU     4       ;COMMAND BIT ON COPR
                ;
                ;
3706                    ORG     3706H   ;TABLE ENTRY FOR USER
                                        ;SUPPLIED READER
3706 C31B37             JMP     371BH   ;JUMP TO DRIVER
371B                    ORG     371BH
371B DB03               IN      DIPR    ;INPUT DATA
371D 2F                 CMA             ;COMPLEMENT DATA
371E B7                 ORA     A       ;CLEAR CARRY
371F F5                 PUSH    PSW
3720 AF                 XRA     A       ;CLEAR ACCUMULATOR
3721 D301               OUT     COPR    ;SEND A 100 MICROSEC.
3723 CD2F37             CALL    DCMI    ;ADVANCE PULSE TO
3726 3E04               MVI     A,CBIT  ;THE READER THROUGH
3728 D301               OUT     COPR    ;PORT COPR
372A CD3837             CALL    DLYM    ;5 MSEC.  DELAY
372D F1                 POP     PSW
372E C9                 RET
                ;
                ;
                ;100 MICROSE.  DELAY SUBROUTINE
                ;
 72F F5         DCMI:   PUSH    PSW
3730 3E0C               MVI     A,12
3732 3D         DCM0:   DCR     A
```

```
3733 C23237             JNZ     DCM0
3736 F1                 POP     PSW
3737 C9                 RET
                ;
                ;
                ;5 MILLISEC. DELAY SUBROUTINE
                ;
3738 F5         DLYM:   PUSH    PSW
3739 3E2A               MVI     A,42
373B CD2F37     DLY0:   CALL    DCMI
373E 3D                 DCR     A
373F C23B37             JNZ     DLY0
3742 F1                 POP     PSW
3743 C9                 RET
0000            END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AC5

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 (use additional sheets if necessary)

Program Title

Tally - allows Tally 2200 line printer to be used in the assembly stage of programming with Intellec 80.

Function

Program first tests to see if the printer is ready. Then it checks the data in the output line to see if it is output or control data. If output it is sent to the Tally buffer until the buffer is full. With the control for line feed, null or buffer full, the line is printed. Page control is also given.

Required Hardware

One interface board connected to a Tally 2200 line printer. I/O ports connected to J4 J5 on second I/O board. (Full details in Figure 1 of attachment)

Required Software

Assembler and monitor

Input Parameters

Gate 7 Bit 0 -- TEST = 1 when the printer is ready
1 BUFB = 1 if hardware is still in use
2 HDWB = 1 if hardware is still in use
3 CRST = 1 when control signal has been received
4 BUFF = 1 when buffer is full
5 VFU1 = 1 when end of page is reached
6 VFU2 = 1 for start of new page

Output Results

Gate 6 8-bit ASCII data

Gate 7 Bit 0 resets receive buffer
1 SLEW = 1 to start paper feed
2 CSCD = 1 to strobe data into Tally
3 PCMD = 1 to start printing

Registers Modified:	Assembler/Compiler Used:
A	8080 MDS Macro Assembler Ver 1.0
RAM Required:	**Programmer:**
0	Miguel Angel Ainsa/S. Pick
ROM Required:	**Company:**
96 bytes (1 chip)	ITTLS - Standard Electrica
Maximum Subroutine Nesting Level:	**Address:** Avenida America KM7.2 Madrid 27 SPAIN
1	

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

intel®

Ref. No. AC6

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

Model 101 Centronics Printer Handler

Function

Accepts character output for Model 101 Centronics printer from assembler or other source. Buffers print characters in RAM performing TTY compatible operations with control characters. Causes line to be printed upon receipt of line feed. Counts lines and keeps track of pagination. Inserts title at top of each page.

Required Hardware

Model 101 Centronics printer with appropriate connectors and cable. Out Port 3 goes to data and strobe of Model 101, bits 0-6 are data and bit 7 is strobe. Input Port 1, bit 7 is ready status.

Required Software

No additional software required.

Input Parameters

C register to contain ASCII character to be printed.

Output Results

A,C register modified. All others preserved.

Registers Modified: A,C	**Assembler/Compiler Used:** 8080 Macro Assembler
RAM Required: 110_{16}	**Programmer:** T. Riddell
ROM Required: None	**Company:** Data Test Corporation
Maximum Subroutine Nesting Level: 3 + 8 bytes stack storage	**Address:** 2450 Whitman Road Concord, Ca.

```
                        ;REF. NO. AC6
                        ;PROGRAM TITLE MODEL 101 CENTRONICS PRINTER HANDLER
                        ;
                        ;
                        ;
                        ;ASSEMBLER ORIENTED LINE PRINTER HANDLER
                        ;
3700                            ORG     3700H           ;ABS JUMP VECTORS
3700 C39C3C                     JMP     3C9CH           ;COMMAND IN VECTOR TTY
3703 C33C37                     JMP     ASYPRT          ;COMMAND OUT VECTOR
3712                            ORG     3712H
3712 C33C37                     JMP     ASYPRT          ;LIST OUT VECTOR
                        ;
3720                            ORG     3720H
3720                    LISTOUT EQU     $
3720 79                         MOV     A,C             ;COMPLEMENT CHARACTER
3721 2F                         CMA
3722 4F                         MOV     C,A
3723 DB01               STATLOOP:       IN      STATUS  ;GET STATUS FROM PRTR
3725 E680                       ANI     80H
 727 CA2337                     JZ      STATLOOP        ;LOOP TIL READY
372A 3E7F                       MVI     A,7FH           ;MASK OF F  CHARACTER
372C A1                         ANA     C
372D D303                       OUT     LPRT            ;MARK CHAR OUT
372F C680                       ADI     80H             ;STROVE CHAR OT
3731 D303                       OUT     LPRT
3733 3E7F                       MVI     A,7FH           ;MASK OFF CHAR
3735 A1                         ANA     C
3736 D303                       OUT     LPRT            ;MARK CHAR OUT
3738 79                         MOV     A,C             ;RESTORE CHARACTER
3739 2F                         CMA
373A 4F                         MOV     C,A
373B C9                         RET                     ;***EXIT
0001                    STATUS  EQU     1               ;STATUS IN PORT
0003                    LPRT    EQU     3               ;LINE PRT DATA OUT
                        ;
373C                    ASYPRT  EQU     $
373C 79                         MOV     A,C             ;SAVE ENVIFONMENT
373D 320536                     STA     CHARSV
3740 C5                         PUSH    B               ;SAVE REGS
3741 D5                         PUSH    D
3742 E5                         PUSH    H
3743 F5                         PUSH    PSW
                        ;
3744 3A0436             CKLC:   LDA     LINCNT
 747 E6FF                       ANI     0FFH
_749 C26237                     JNZ     CKCR            ;BR IF LINE COUNT NOT ZERO
374C 0E0C                       MVI     C,0CH
374E CD2037                     CALL    LISTOUT         ;FORM FEED
```

```
3751 3ECA              MVI     A,-54
3753 320436            STA     LINCNT          ;SET COUNT=-54
3756 210636            LXI     H,HEAD
3759 CD2636            CALL    PRTLINE         ;PRINT HEAD-2LST
375C CD9A37            CALL    CLRBUF
375F C36A37            JMP     CSET1
                ;
3762 3A0536   CKCR:    LDA     CHARSV          ;GET CHARACTER
3765 FE0D              CPI     0DH             ;CARRIAGE RETURN
3767 C27337            JNZ     CKLF            ;BR IF NOT
376A 213536   CSET1:   LXI     H,BUF           ;RESET CHAR PTR
376D 220036            SHLD    CHARPOS
3770 C38A37            JMP     ASYEND
                ;
3773 FE0A     CKLF:    CPI     0AH             ;LINE FEED
3775 C28F37            JNZ     DOPRT           ;BR IF NOT
3778 218436            LXI     H,BUF+79        ;CCR TO BUF
377B 360D              MVI     M,0DH
377D 213536            LXI     H,BUF
3780 CD2636            CALL    PRTLINE
3783 CD9A37            CALL    CLRBUF
 786 210436            LXI     H,LINCNT        ;LINE COUNT + 1
3789 34                INR     M
378A F1       ASYEND:  POP     PSW             ;RESTORE REGISTERS
378B E1                POP     H
378C D1                POP     D
378D C1                POP     B
378E C9                RET                     ;*********EXIT
                ;
378F 2A0036   DOPRT:   LHLD    CHARPOS
3792 77                MOV     M,A             ;PUT CHAR IN BUF
3793 23                INX     H
3794 220036            SHLD    CHARPOS
3797 C38A37            JMP     ASYEND
                ;
379A          CLRBUF   EQU     $               ;ROUTINE ENTRY
379A 0650              MVI     B,80            ;CHAR CNT
379C 213536            LXI     H,BUF
379F 3E0D              MVI     A,0DH           ;SPACE
37A1 77       CLRLP:   MOV     M,A             ;TO MEMORY
37A2 23                INX     H
37A3 05                DCR     B
37A4 C2A137            JNZ     CLRLP           ;DO WHOLE BUFFER
37A7 C9                RET                     ;****EXIT
3600                   ORG     3600H
3600 0000     CHARPOS:         DW              0
 S02 00       PRTOFF:  DB      0
_603 00       LINOFF:  DB      0
3604 00       LINCNT:  DB      0
3605 00       CHARSV:  DB      0
```

```
3606             HEAD    EQU     $
3606 09093830            DB      '
360A 3830204D                                8080 MACRO ASSEMBLER VER 2.0'
360E 4143524F
3612 20415353
3616 454D424C
361A 45522056
361E 45522032
3622 2E30
```

```
3624 0D                  DB      0DH
3625 00          LFFLG:  DB      0
3626             PRTLINE EQU     $
 526 4E                  MOV     C,M             ;GET CHARACTER (X)
3627 CD2037              CALL    LISTOUT
362A 79                  MOV     A,C
362B FE0D                CPI     0DH             ;COMPARE WITH CARRIAGE RETURN
362D CA3436              JZ      LINEND
3630 23                  INX     H
3631 C32636              JMP     PRTLINE
3634 C9          LINEND: RET                     ;****EXIT
3635             BUF:    DS      80              ;LINE PRINTER BUFFERD,    ;COMMEN
0000                     END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AC7

☐ 4004 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

High Speed Paper Tape Reader with Stepper Motor Control

Function

This circuit and program allow paper tape to be read at approximately 150 characters per second. The reader is assigned by monitor command "AR=1." The program uses electronic damping, under software control, of a stepper motor to increase stepping speed and precision.

Required Hardware

Intellect 8-80
(See attached schematic).
Output port 3, input ports 1 and 3 shared with TTY interface.

Required Software

8080 Monitor

Input Parameters

Monitor Commands:

"AR=1" assigns reader device

"R" read hex tape or any other reader commands

Output Results

The high speed reader will perform the same functions as TTY tape reader under command of monitor, text editor, macro assembler, or other user defined programs.

Registers Modified: A, Flags	**Maximum Subroutine Nesting Level:** 0
RAM Required: 2 Bytes in Stack	**Assembler/Compiler Used:** Intellec Macro Assembler
ROM Required: 80 Bytes (50H)	**Programmer:** J. Garner
	Company: Bell Helicopter Box 482, Ft. Worth, Texas 76101

```
                        ;REF. NO. AC7
                        ;PROGRAM HIGH SPEED PAPER TAPE READER WITH STEPPER MOTOR CONTROL
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ; HIGH SPEED READER
                        ; FOR USE WITH INTELLEC 8/80
                        ; I/O PORTS ARE SHARED WITH
                        ; TTY INTERFACE.
                        ; THE STEPPER MOTOR CONTROL
                        ; MAY BE MODIFIED FOR OTHER
                        ; MOTORS BY CHANGING THE
                        ; ACCELERATION AND
                        ; DECELERATION TIMES.

3706                            ORG     3706H   ; OR 3709H FOR READER #2
3706 C32137                     JMP     3721H
 721                            ORG     3721H
3721 DB01                       IN      1       ; READ MOTOR STATUS
3723 2F                         CMA
3724 17                         RAL             ; SET CARRY IF NO TAPE
3725 D8                         RC              ; RETURN ON NO TAPE
3726 E5                         PUSH    H       ; STORE REGISTER CONTENTS
3727 E6C0                       ANI     0C0H    ; MASK FOR STATUS
3729 FA6F37                     JM      OUT1    ; LOAD PATTERN "001"
372C CA7437                     JZ      OUT2    ; LOAD PATTERN "010"
372F 3E04                       MVI     A,4
3731 D303               OUTM:   OUT     3       ; STEP MOTOR
                        DELAY   MACRO   TDEL    ; DELAY MACRO
                                MVI     H,TDEL

                        CNT1:   MVI     L,0CH   ; LOAD INNER LOOP COUNT
                        CNT2:   DCR     L
                                JNZ     CNT2
                                DCR     H
                                JNZ     CNT1
                                ENDM            ; END DELAY MACRO
                      +         DELAY   17H     ; STEP FORWARD (ACCELERATE)
3733 2617             +         MVI     H,00017H
                      +
3735 2E0C             +CNT1:    MVI     L,0CH   ; LOAD INNER LOOP COUNT
3737 2D               +CNT2:    DCR     L
 738 C23737           +         JNZ     CNT2
373B 25               +         DCR     H
373C C23537           +         JNZ     CNT1
```

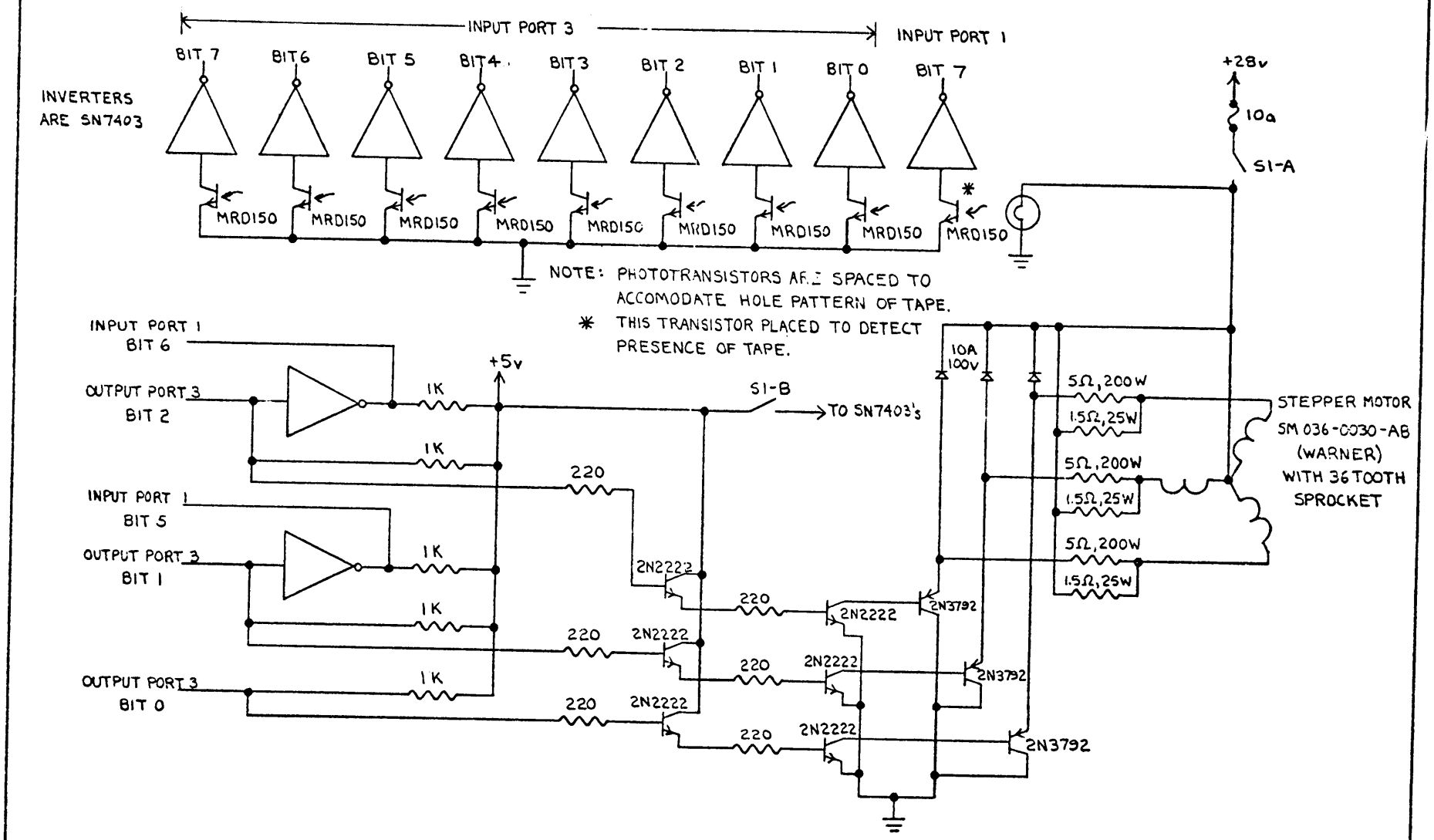
```
373F 2617               MVI     H,00017H

3741 2E0C     CNT1:     MVI     L,0CH     ; LOAD INNER LOOP COUNT
3743 2D       CNT2:     DCR     L
3744 C24337             JNZ     CNT2
3747 25                 DCR     H
3748 C24137             JNZ     CNT1

374B 0F                 RRC
374C D303               OUT     3         ; STEP BACK (DECELERATE)
             +          DELAY   6
374E 2606    +          MVI     H,00006H
             +
3750 2E0C    +CNT1:     MVI     L,0CH     ; LOAD INNER LOOP COUNT
3752 2D      +CNT2:     DCR     L
3753 C25237  +          JNZ     CNT2
3756 25      +          DCR     H
3757 C25037  +          JNZ     CNT1


375A 07                 RLC
 75B D303               OUT     3         ; STEP FORWARD (HOLD)
             +          DELAY   0CH       ; ALLOW TO STOP
375D 260C    +          MVI     H,0000CH
             +
375F 2E0C    +CNT1:     MVI     L,0CH     ; LOAD INNER LOOP COUNT
3761 2D      +CNT2:     DCR     L
3762 C26137  +          JNZ     CNT2
3765 25      +          DCR     H
3766 C25F37  +          JNZ     CNT1


3769 DB03               IN      3         ; READ DATA
376B 2F                 CMA
376C E1                 POP     H         ; RESTORE REGISTERS
376D B7                 ORA     A         ; CLEAP THE CARRY BIT
376E C9                 RET
376F 3E09     OUT1:     MVI     A,09H     ; LOAD PATTERN "100"DRIVE
3771 C33137             JMP     OUTM
3774 3E02     OUT2:     MVI     A,2
3776 C33137             JMP     OUTM
0000                    END
```



HIGH SPEED TAPE READER

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB19

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

Terminal Editor

Function

Procedure for controlling an ASR733 Texas Instruments terminal equipped with RDC (remote control device) option. Search a line in a file contained in cassette 1 with or without copying on cassette 2. The procedure is linked to the Intellec monitor.

Required Hardware

Intellec 8/Mod 80

Required Software

Intellec 80 monitor

Input Parameters

The procedure is accessed via the "U" and "V" monitor commands

<Search without copying command>::=U<String>

<Search copying command>::=V<String>

String can be of any length from 1 to 86 characters, the convention for EOF is a "$" in column 1. The line typed in need not be of the same length of the line to be found. A carriage return as first character of the string means to search the previous string in the following file.

Output Results

When the line is not found before an EOF the terminal prints a blank line. When the string is found, the terminal prints an equal sign ("=").

Registers Modified:	Assembler/Compiler Used:
NA	PLM80
RAM Required:	**Programmer:**
130	Enrico Massetti
ROM Required:	**Company:**
187H	Laboratorio Massetti
Maximum Subroutine Nesting Level:	**Address:**
Note: monitor modified	20133 Milano, Via Ronchi 17 ITALY

```
00001  1
00002  1   /*REF NO. AB19 */
00003  1   /*PROGRAM TITLE TERMINAL EDITOR*/
00004  1
00005  1   /* PROCEDURE FOR CONTROLLING AN ASR733 TEXAS TERMINAL EQUIPED
0  06  1   WITH RDC (REMOTE DEVICE CONTROL) OPTION.*/
00007  1   DECLARE (CAR,BLOCK) (86) BYTE,
00008  1          CR LITERALLY '13';
00009  1   /* LINKAGE TO 8080 MONITOR.*/
00010  1   DECLARE CO LITERALLY '3C32H';
00011  1   TTYOUT:PROCEDURE (CHAR);
00012  2   DECLARE CHAR ADDRESS;
00013  2          CHAR=SHL (CHAR,8);
00014  2          GO TO CO;
00015  2   END TTYOUT;
00016  1   DECLARE CI LITERALLY '3C76H';
00017  1   TTYIN: PROCEDURE BYTE;
00018  2        GO TO CI;
00019  2   END TTYIN;
00020  1   DLEOUT: PROCEDURE ( CARATTERE);
00021  2   DECLARE CARATTERE BYTE;
00022  2   CALL TTYOUT(16);/* DLE */
00023  2   CALL TTYOUT(CARATTERE);/* CARATTER */
00024  2   RETURN;
00025  2   END DLEOUT;
00026  1   READ$BLOCK: PROCEDURE (INBLOCK);
00027  2   DECLARE INBLOCK ADDRESS,
00028  2           BLOCK BASED INBLOCK BYTE,
00029  2           N BYTE,
00030  2           TEMP BYTE;
00031  2   IF (TEMP:= TTYIN)=CR THEN RETURN;
00032  2                         ELSE BLOCK(0)= TEMP;
00033  2   DO N = 1 TO 85;
00034  2      BLOCK (N)=TTYIN;
00035  3      IF BLOCK(N)=CR THEN RETURN;
00036  3      END;
00037  2   RETURN;
00038  2   END READ$BLOCK;
00039  1   /*
00040  1   */
00041  1   ONE$BLOCK: PROCEDURE BYTE;
00042  2   CALL DLEOUT(55); /* DLE 7 = BLOCK FWD */
00043  2   CALL READ$BLOCK (.BLOCK);
00044  2   IF BLOCK = 36 THEN DO;/* 36= $ END OF FILE */
00045  2        CALL TTYOUT (20);/* DC4 = RECORD OFF */
00046  3        CALL DLEOUT (57);/* DLE 9 = PRINTER ON */
00047  3        RETURN 0;
00048  3        END;
00049  2   ELSE RETURN 0FFH;
00050  2   END ONE$BLOCK;
00051  1   PREDISP: PROCEDURE BYTE;
00052  2   DECLARE STATUS BYTE;
00053  2   /* CASSETTE 1 IN PLAYBACK 2 IN RECORD */
00054  2   CALL DLEOUT (48); /* DLE 0 = PRINTER OFF */
00055  2   CALL DLEOUT (54); /* DLE 6 = CASS1 PLY,CASS2 RECORD */
00056  2   CALL DLEOUT (60); /* DLE < = STATUS REQUEST */
00  7  2   STATUS = TTYIN;/* VERIFY TERMINAL STATUS */
00058  2   VERIFICA$STATO : IF (STATUS AND 1FH)<> 1FH THEN DO;/* STATUS NOT OK */
00059  2        CALL DLEOUT (57); /* DLE 9 = PRINTER ON */
00060  3        RETURN 0;
```

```
00061  3        END;
00062  2    ELSE RETURN 0FFH;
00063  2    END PREDISP;
00064  1    /* THE PROCEDURE RETURNS 0 IF THE TERMINAL IS NOT PREDISPOSED, 1 IF ALL OK*/
00065  1    /* PROCEDURE FOR CONFRONTIG THE CAR AND BLOCKS*/
00066  1    BLOCK$CONFR: PROCEDURE BYTE;
00067  2    DECLARE N BYTE;
00068  2    DO N=0 TO 85;
00069  2       IF CAR(N)<>BLOCK(N) THEN DO;
00070  3          IF CAR(N)=CR THEN RETURN 1;
00071  4                       ELSE RETURN 0;/*BLOCKS DIFFERENT*/
00072  4          END;
00073  3       ELSE IF CAR(N)=CR THEN RETURN 0FFH;/*IDENTICAL BLOCK*/
00074  3       END;
00075  2    RETURN 0FFH; /*IDENTICAL BLOCKS*/
00076  2    END BLOCK$CONFR;
00077  1    /* THE PROCEDURRE RETURNS TO THE CALLING POINT THE FOLLOWING VALUES:
00078  1            0 = BLOCK NOT EQUALS
00079  1            1 = BLOCK CAR IS A SUBSET OF BLOCK BLOCK
00080  1            0FFH = BLOCKS INDENTICAL */
00081  1    /* PROCEDURE FOR CONTROLLING AN ASR 733 TEXAS TERMINAL EQUIPED WITH RDC
00082  1    OPTION.
00083  1    SEARCH OF A LINE IN FILE CONTAINED INTO CASSETTE 1 WITH OR WITHOUT
00084  1    PARAMETER COPIA=0 : SEARCH WITHOUT COPYING; PARAMETER COPIA=1 ; SEARCH
00085  1    COPYING. */
00086  1    RICERCA$BLOCK: PROCEDURE (COPIA);
00087  2    DECLARE (COPIA,TEMP)BYTE;
00088  2    ENTER$BLOCK: CALL READ$BLOCK (.CAR);
00089  2    /* THE LINE TO BE FOUND COMES FROM THE KEYBOARD AND IS PUT IN THE MEMORY
00090  2    IN THE CAR POSITIONS. THE LINE CAN BE OF ANY LENGH FROM 1 86 CARACTERS
00091  2    THE CONVENCTION FOR AND OF FILE IS THE ASCII CARACTER $ IN COLUMN 1.
00092  2    WHEN THIS CARACTER IS FOUND IN COLUM 1 THE TERMINAL PRINTS <> AND THE
00093  2    PROGRAM RETURNS TO THE CALLING POINT.
00094  2    THE LINE TYPED IN NEEDS NOT TO BE OF THE SAME LENGH OF THE LINE TO BE FOUND
00095  2    FOR EXAMPLE:
00096  2    ------------------------------------------------------------------------
00097  2    FIRST,SECOND,THIRD
00098  2            (THIS IS THE LINE TO BE FOUND)
00099  2    FIRST
00100  2            (THIS IS A VAID INDICATION TO FIND THAT LINE)
00101  2    ------------------------------------------------------------------------
00102  2    */
00103  2    IF PREDISP=0 THEN RETURN;/* TERMINAL NOT PREDISPOSED*/
00104  2    IF COPIA =1 THEN CALL TTYOUT(18);/*DC2=RECORD ON */
00105  2    LOOP: IF ONEBLOCK=0 THEN DO;/*BLOCK NOT FOUND.FOUND EOF ($) PRINTS<> */
00106  2                  CALL TTYOUT (60);/* < */
00107  3                  CALL TTYOUT (62);/* > */
00108  3                  RETURN; END;
00109  2    /* BLOCK FOUND, START CONFRONT*/
00110  2    IF (TEMP:=BLOCK$CONFR)=0 THEN GO TO LOOP;/*BLOCKS DIFFERENT*/
00111  2    CALL TTYOUT (20); /*DC4=RECORD OFF*/
00112  2    CALL DLEOUT (41); /*DLE9=PRINTER ON*/
00113  2    CALL TTYOUT (61); /*=*/
00114  2    RETURN;
00115  2    END RICERCA$BLOCK;
00116  1    EOF
NO    OGRAM ERRORS
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AC8

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000

(use additional sheets if necessary)

Program Title

Intellec 8/MOD 80-Silent 700 Interface

Function

Required Hardware

This note describes an interface between the Intellec8/Mod 80 Microcomputer Development System and the Texas Instruments "Silent 700" Electronic Data Terminal.

Required Software

The interface consists of an Intellec/Silent 700 adapter. In addition, the "Silent 700" terminal and Intellec 8/Mod 80 must be modified. The adapter and equipment modifications are described in this note.

Input Parameters

The interface was designed for use with the basic Intellec configuration (imm8-84A) and a Model 733ASR terminal equipped with the 1200 Baud Transmission option and either the Automatic Device Control (ADC) option or Remote Device Control (RDC) option.

Output Results

Registers Modified:	**Assembler/Compiler Used:**
RAM Required:	**Programmer:** Phil Jensen
ROM Required:	**Company:**
Maximum Subroutine Nesting Level:	**Address:** 6530 LBJ Freeway, #178 Dallas, Texas 75240

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AD3

☐ 4004 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

Interrupt Service Routine

Function

Handles multiple-level interrupts, saving all registers and flags and outputting the status of the current interrupt to an external status latch. See additional sheet.

Required Hardware

Priority Interrupt Control Unit (8214) with status latch inputs connected to appropriate output port (#255 in sample program).

Required Software

INTX and ROUTX for each interrupt level to be handled, plus necessary routines to service the interrupts.

Input Parameters

A RESTART instruction is forced onto the data bus by the interrupt control unit.

Output Results

The status of the current interrupt routine is output on bits 0, 1, and 2 of output port 255. The service routine is executed and all registers and flags from the calling program are save and restored after the interrupt routine is complete.

Registers Modified: All registers are saved	**Maximum Subroutine Nesting Level:** 12 bytes per interrupt level
RAM Required: 1 byte plus stack	**Assembler/Compiler Used:** 8080 Macro Assembler V4.0
ROM Required: 28 bytes + 21 bytes per interrupt level.	**Programmer:** Donald E. Shorter
	Company: Telcom, Inc. 8027 Leesburg Pike Vienna, Va. 22180

Interrupt Service Routine

This routine is designed to be used with an external hardware interrupt control unit and interrupt status latch. A type 8214 Priority Interrupt Control Unit may be used to generate the appropriate RESTART instructions; the current status latch pins should be connected to bits 0, 1, and 2 of output port 255.

With any priority interrupt situation, a problem arises when one interrupt service routine is interrupted by a higher priority signal. When the higher priority routine begins, it must output its status to the current status latch; after it is finished, control is transferred to the lower priority routine which was interrupted. This requires that the lower priority status must be sent to the status latch so that any higher priority interrupt may take control again.

This routine uses one byte of RAM to store any current status (this byte should be set to the lowest priority at the beginning of the main program). When an interrupt occurs, this previous status is pushed into the stack and the current status is output to the status latch. At the end of the routine, the previous status is popped off the stack and output to the latch just before the program goes into the routine to restore the registers and return to the calling program. In this way, the status latch will always contain the current status, even if several interrupts are stacked one on top of another.

The interrupt service routine saves the contents of all registers and flags and restores them at the end of the service program. Each level of interrupt requires 12 bytes of RAM for stack operations, plus whatever is required to actually service the interrupt. Interrupts may be nested as far as the available stack will permit.

In the sample program, INTX is the routine which resides in an eight-byte section of the first 64 bytes of system memory; "X" is the number, 0-7, of the interrupt routine. One INTX is required for each interrupt level desired. INTX saves the contents of all registers and flags and then jumps to the service routine.

ROUTX is the service routine which saves the previous status in the stack and outputs the current status to the external latch. "X" is the number, 0-7, of the interrupt routine. The interrupt flip-flop is enabled and any instructions necessary to service the interrupt are executed. Then

ROUTX disables the interrupt flip-flop and restores the previous status to RAM and also to the status latch on port 255. One ROUTX is required for each interrupt level desired.

SAVE is the subroutine which performs the actual status updating and outputting operations. Only one SAVE is required.

RSTOR is the subroutine which restores the previous status to the latch and the RAM location (STATU). Only one RSTOR is required.

FINIS is the routine which restores all registers and flags to their original condition and returns to the calling program. One FINIS is required.

Test Program

A test program was developed to test the interrupt service routine; it requires the Monitor, Version 2.0 in the Intellec 8/Mod 80.

The six INTX routines (INT2 - INT7) are loaded in RAM starting at location 10H (the first 16 bytes are required for operation of the Monitor and cannot be used). The main program is then loaded in RAM starting at location 200H.

The main program continuously prints a message on the TTY; at any time during this message, a RST2 - RST7 may be entered on the data bus of the Intellec by setting up the appropriate code on the data switches and pressing the INT switch. When it receives the interrupt, the program will begin to print the message corresponding to the number of that interrupt. Another interrupt may then be inserted and the corresponding message will be printed. When a message has been completely sent, the program will revert back to the remainder of the previous message, etc., until it is back to the main program message.

A delay routine was incorporated to slow the printing so that the operator has time to set up and generate another interrupt signal. As each interrupt routine is active, the current status appears on the lower three bits of the front panel lights connected to output port 255. A RST1 will cause the program to be transferred back to the system Monitor.

Donald E. Shorter
Telcom, Inc.
8027 Leesburg Pike
Vienna, Virginia 22180

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AD4

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

Interrupt Handler Re-entrant

Function

On processor receipt of an interrupt instruction (RST 0-7), this program saves the machine state and previous interrupt level on the stack, transmits the new service level to the interrupt control unit (ICU), executes a subroutine corresponding to the level interrupt received, then restores the machine and ICU to their pre-interrupt state before resuming executing the interrupted program.

Required Hardware

Interrupt circuitry; test routine assumes TTY on 0 and 1; test interrupts were from Intellec 8/80 front panel.

Required Software

Service programs appropriate to the various interrupt levels expected background initialization; TTY test routine uses "CO" and "CRLF" in monitor or equiv.

Input Parameters

Restart instructions, RST 0-7, either from ICU or programmed.

Output Results

Interrupt routine causes branches to routines according to pointers: SERV1, SERV2, etc. supplied by user for his system. Test program types the level of interrupt received on the assigned monitor console device. See attached sheet.

Registers Modified: None; state restored on exit	**Assembler/Compiler Used:** Intellec 8 Macro Assembler
RAM Required: 6 byte variables, plus 10 bytes/nesting level on stack, to stack limit	**Programmer:** John M. Mills, Lecturer
ROM Required: bytes 0H through 3FH, plus 44H bytes elsewhere = 84H	**Company:** University of Wisconsin
Maximum Subroutine Nesting Level: Fully re-entrant;stack limit	**Address:** Dept. of Mechanical Engr. Madison, Wisconsin 53706

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB20

☐ 4004 ☐ 4040 ☒ 8008 ☐ 8080

(use additional sheets if necessary)

Program Title

8008 DISASSEMBLER

Function

This program is used to obtain an assembly language listing of a machine coded program in memory.

Required Hardware

INTELLEC 8 with console I/O device.
12K of RAM Memory (ORG commands can be changed for less).

Required Software

INTEL 8008 MONITOR program ver 2.1. (Monitor subroutine addresses are listed at the start of the program for updating to other monitor versions.)

Input Parameters

The Monitor 'G' command is used to enter the program with a response of a carriage return and line feed. The low address and the high address is then entered in hexadecimal format, separated by a comma, and followed by a carriage return.

Output Results

A listing on the console output device of the hexadecimal address, the hexadecimal machine instruction, and the assembly language instruction.

Invalid machine instruction codes are listed as hexadecimal numbers.

Registers Modified:	**Assembler/Compiler Used:** INTELLEC 8 MACRO ASSEMBLER
RAM Required:	**Programmer:** Hal King
ROM Required:	**Company:** Brooks Research & Mfg
Maximum Subroutine Nesting Level:	**Address:** 5612 Brighton 64130 Kansas City, Missouri

98-034B

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 ☒8080 ☐8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

8080 DIS-ASSEMBLER (SEE NOTE BELOW)

Function

This program inputs an ISIS-II file in HEX format and generates a symbolic assembly language program suitable for editing and/or assembling.

Required Hardware

MDS-800, Dual Diskette Drives
32K to operate
64K to compile

Required Software

PLM-80 Diskette Resident Compiler (to compile the source)
ISIS-II (to execute the program)

Input Parameters

ISIS-II file (disk file, :HR:, etc.) in Intel's HEX format
(use OBJHEX if it's originally in ISIS-II, 8080 object module format)

REVISED 10/78
by Gary Carleton
Intel Corp.

Output Results

Assembly language source program

NOTE: Program source is in PLM-80 Resident compiler form.

Available on diskette only

Registers Modified: All	Programmer: Erick Serdahl
RAM Required: 32K to operate 64K to compile	Company: Acurex Corp. Icore Div. R+D Dept.
ROM Required: N/A	Address: 555 Clyde Ave.
Maximum Subroutine Nesting Level: 8	City: Mountain View
Assembler/Compiler Used: PLM-80	State: CA

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

DISASM (8080 Disassembler)

Function

DISASM is intended as a software development and debugging aid. Operating on resident object code, it produces an assembly language equivalent which is printed on a TTY terminal. In this present form, the program starts at a given memory address and steps sequentially through memory until manually halted.

Required Hardware

TTY on parts 0 and 1 (as Intellec system configuration)

Required Software

Intellec 8/80 Monitor console output routine (or equivalent)

Input Parameters

Output Results

See reference #AB 154, Page 4-725 for SYMBOL TABLE INSERTER

Registers Modified:	A, B, C, D, H, L	Programmer:	S. N. Brunner
RAM Required:	2 Bytes (pluss Stack)	Company:	General Electric
ROM Required:	791 Bytes	Address:	Bldg 50-1 2901 East Lake Road
Maximum Subroutine Nesting Level:	2 levels (4 bytes)	City:	Erie
Assembler/Compiler Used:	8080 Macro Assembler, V 3.0	State:	PA 16531

```
                        ;REF. NO. AB22
                        ;PROGRAM NAME DISASM (8080 DISASSEMBLER)
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
1B00                    ORG     1B00H
3809                CO  EQU     3809H

1B00 2A081D     RDBYTE: LHLD    PC
1B03 7E                 MOV     A,M
1B04 23                 INX     H
1B05 22081D             SHLD    PC
1B08 C9                 RET

1B09 3C         RGPRNT: INR     A
1B0A E607               ANI     07
1B0C FE06               CPI     06
 B0E DA131B             JC      RGP1
1B11 C603               ADI     03
1B13 FE05       RGP1:   CPI     05
1B15 DA1A1B             JC      RGP2
1B18 C602               ADI     02
1B1A C641       RGP2:   ADI     41H
1B1C 4F                 MOV     C,A
1B1D C30938             JMP     CO

1B20 47         DECODE: MOV     B,A
1B21 E6F0               ANI     0F0H
1B23 0F                 RRC
1B24 0F                 RRC
1B25 0F                 RRC
1B26 0F                 RRC
1B27 C690               ADI     90H
1B29 27                 DAA
1B2A CE40               ACI     40H
1B2C 27                 DAA
1B2D 4F                 MOV     C,A
1B2E CD0938             CALL    CO
1B31 78                 MOV     A,B
1B32 E6F0               ANI     0F0H
1B34 C690               ADI     90H
1B36 27                 DAA
'B37 CE40               ACI     40H
 B39 27                 DAA
1B3A 4F                 MOV     C,A
1B3B C30938             JMP     CO
```

```
1B3E 0604      PRINT:  MVI     B,4
1B40 4E        P1:     MOV     C,M
1B41 CD0938            CALL    CO
1B44 23                INX     H
1B45 05                DCR     B
1B46 C2401B            JNZ     P1
1B49 0E20              MVI     C,20H
1B4B C30938            JMP     CO
1B4E 7A        XTRACT: MOV     A,D
1B4F E638              ANI     38H
1B51 0F                RRC
1B52 0F                RRC
1B53 0F                RRC
1B54 C9                RET

1B55 CD4E1B    CCPRNT: CALL    XTRACT
1B58 87                ADD     A
1B59 4F                MOV     C,A
1B5A 21081E            LXI     H,CCODE
1B5D 09                DAD     B
 B5E 4E                MOV     C,M
1B5F CD0938            CALL    CO
1B62 23                INX     H
1B63 4E                MOV     C,M
1B64 CD0938            CALL    CO
1B67 0E20              MVI     C,20H
1B69 CD0938            CALL    CO
1B6C C30938            JMP     CO

1B6F CD4E1B    RPPRNT: CALL    XTRACT
1B72 E606              ANI     06
1B74 FE06              CPI     06
1B76 C2091B            JNZ     RGPRNT
1B79 0E53              MVI     C,53H
1B7B CD0938            CALL    CO
1B7E 0E50              MVI     C,50H
1B80 C30938            JMP     CO

1B83 0E0D      DISASM: MVI     C,0DH
1B85 CD0938            CALL    CO
1B88 0E0A              MVI     C,0AH
1B8A CD0938            CALL    CO
1B8D 2A081D            LHLD    PC
1B90 7C                MOV     A,H
1B91 CD201B            CALL    DECODE
 B94 7D                MOV     A,L
 B95 CD201B            CALL    DECODE
1B98 0E20              MVI     C,20H
1B9A CD0938            CALL    CO
```

```
1B9D CD0938          CALL    CO
1BA0 CD001B          CALL    RDBYTE
1BA3 57              MOV     D,A
1BA4 210A1D          LXI     H,TABLE
1BA7 011100          LXI     B,11H
1BAA BE        D1:   CMP     M
1BAB CAFA1C          JZ      TG1
1BAE 23              INX     H
1BAF 0D              DCR     C
1BB0 C2AA1B          JNZ     D1
1BB3 0E0A            MVI     C,0AH
1BB5 BE              CMP     M
1BB6 BE        D2:   CMP     M
1BB7 CAE61C          JZ      TG2
1BBA 23              INX     H
1BBB 0D              DCR     C
1BBC C2B61B          JNZ     D2
1BBF 0E06            MVI     C,6
1BC1 BE        D3:   CMP     M
1BC2 CACA1C          JZ      TG3
1BC5 23              INX     H
 3C6 0D              DCR     C
1BC7 C2C11B          JNZ     D3
1BCA E6C0            ANI     0C0H
1BCC FE40            CPI     40H
1BCE CAB01C          JZ      MG0
1BD1 FE80            CPI     80H
1BD3 CAA11C          JZ      MG1
1BD6 7A              MOV     A,D
1BD7 E6C7            ANI     0C7H
1BD9 D604            SUI     04
1BDB CA921C          JZ      MG2
1BDE 3D              DCR     A
1BDF CA8C1C          JZ      MG3
1BE2 3D              DCR     A
1BE3 CA781C          JZ      MG4
1BE6 7A              MOV     A,D
1BE7 E6C0            ANI     0C0H
1BE9 CA4C1C          JZ      MG5
1BEC 7A              MOV     A,D
1BED E6C7            ANI     0C7H
1BEF D6C0            SUI     0C0H
1BF1 CA411C          JZ      MG6
1BF4 D602            SUI     02
1BF6 CA361C          JZ      MG7
1BF9 D602            SUI     02
 BFB CA2B1C          JZ      MG8
 FE D603             SUI     03
1C00 CA1F1C          JZ      MG9
1C03 7A              MOV     A,D
```

```
1C04 E607              ANI     07
1C06 4F                MOV     C,A
1C07 21FF1D            LXI     H,PPOP-1
1C0A 09                DAD     B
1C0B CD3E1B            CALL    PRINT
1C0E CD4E1B            CALL    XTRACT
1C11 FE06              CPI     06
1C13 C29B1C            JNZ     D6
1C16 21FC1D            LXI     H,PPSW
1C19 CD3E1B            CALL    PRINT
1C1C C3831B            JMP     DISASM
1C1F 21F81D    MG9:    LXI     H,PRST
1C22 CD3E1B            CALL    PRINT
1C25 CD201B            CALL    DECODE
1C28 C3831B            JMP     DISASM
1C2B 0E43      MG8:    MVI     C,43H
1C2D CD0938            CALL    CO
1C30 CD551B            CALL    CCPRNT
1C33 C3D51C            JMP     D7
1C36 0E4A      MG7:    MVI     C,4AH
1C38 CD0938            CALL    CO
1C3B CD551B            CALL    CCPRNT
1C3E C3D51C            JMP     D7
1C41 0E52      MG6:    MVI     C,52H
1C43 CD0938            CALL    CO
1C46 CD551B            CALL    CCPRNT
1C49 C3831B            JMP     DISASM
1C4C 21E01D    MG5:    LXI     H,PLXI
1C4F 7A                MOV     A,D
1C50 E60F              ANI     0FH
1C52 3D                DCR     A
1C53 CA6A1C            JZ      MG51
1C56 FE04              CPI     04
1C58 DA5D1C            JC      D4
1C5B D605              SUI     05
1C5D 87        D4:     ADD     A
1C5E 87                ADD     A
1C5F 4F                MOV     C,A
1C60 09                DAD     B
1C61 CD3E1B            CALL    PRINT
1C64 CD6F1B            CALL    RPPRNT
1C67 C3831B            JMP     DISASM
1C6A CD3E1B    MG51:   CALL    PRINT
1C6D CD6F1B            CALL    RPPRNT
1C70 0E2C              MVI     C,2CH
1C72 CD0938            CALL    CO
1C75 C3D51C            JMP     D7
1C78 21DC1D    MG4:    LXI     H,PMVI
1C7B CD3E1B            CALL    PRINT
1C7E CD4E1B            CALL    XTRACT
```

```
1C81 CD091B          CALL    RGPRNT
1C84 0E2C            MVI     C,2CH
1C86 CD0938          CALL    CO
1C89 C3F11C          JMP     D8
1C8C 21D81D    MG3:  LXI     H,PDCR
1C8F C3951C          JMP     D5
1C92 21D41D    MG2:  LXI     H,PINR
1C95 CD3E1B    D5:   CALL    PRINT
1C98 CD4E1B          CALL    XTRACT
1C9B CD091B    D6:   CALL    RGPRNT
1C9E C3831B          JMP     DISASM
1CA1 7A        MG1:  MOV     A,D
1CA2 E638            ANI     38H
1CA4 0F              RRC
1CA5 4F              MOV     C,A
1CA6 21B41D          LXI     H,PADD
1CA9 09              DAD     B
1CAA CD3E1B          CALL    PRINT
1CAD C3C11C          JMP     D9
1CB0 21B01D    MG0:  LXI     H,PMOV
1CB3 CD3E1B          CALL    PRINT
 CB6 CD4E1B          CALL    XTRACT
1CB9 CD091B          CALL    RGPRNT
1CBC 0E2C            MVI     C,2CH
1CBE CD0938          CALL    CO
1CC1 7A        D9:   MOV     A,D
1CC2 E607            ANI     07
1CC4 CD091B          CALL    RGPRNT
1CC7 C3831B          JMP     DISASM
1CCA 79        TG3:  MOV     A,C
1CCB 87              ADD     A
1CCC 87              ADD     A
1CCD 4F              MOV     C,A
1CCE 21941D          LXI     H,TAB3-4
1CD1 09              DAD     B
1CD2 CD3E1B          CALL    PRINT
1CD5 CD001B    D7:   CALL    RDBYTE
1CD8 57              MOV     D,A
1CD9 CD001B          CALL    RDBYTE
1CDC CD201B          CALL    DECODE
1CDF 7A              MOV     A,D
1CE0 CD201B          CALL    DECODE
1CE3 C3831B          JMP     DISASM
1CE6 79        TG2:  MOV     A,C
1CE7 87              ADD     A
1CE8 87              ADD     A
 CE9 4F              MOV     C,A
 CEA 216C1D          LXI     H,TAB2-4
1CED 09              DAD     B
1CEE CD3E1B          CALL    PRINT
```

```
1CF1 CD001B   D8:      CALL    RDBYTE
1CF4 CD201B            CALL    DECODE
1CF7 C3831B            JMP     DISASM
1CFA 79       TG1:     MOV     A,C
1CFB 87                ADD     A
1CFC 87                ADD     A
1CFD 4F                MOV     C,A
1CFE 21271D            LXI     H,TAB1-4
1D01 09                DAD     B
1D02 CD3E1B            CALL    PRINT
1D05 C3831B            JMP     DISASM
1D08          PC:      DS      2
1D0A 00070F17 TABLE:   DB      000H,007H,00FH,017H
1D0E 1F272F37          DB      01FH,027H,02FH,037H
1D12 3F76C9E3          DB      03FH,076H,0C9H,0E3H
1D16 E9EBF3F9          DB      0E9H,0EBH,0F3H,0F9H
1D1A FBC6CED3          DB      0FBH,0C6H,0CEH,0D3H
1D1E D6DBDEE6          DB      0D6H,0DBH,0DEH,0E6H
1D22 EEF6FF22          DB      0EEH,0F6H,0FFH,022H
1D26 2A323AC3          DB      02AH,032H,03AH,0C3H
1D2A CD                DB      0CDH
 D2B 45492020 TAB1:    DB      'EI  ','SPHL','DI  ','XCHG'
1D2F 20535048
1D33 4C444920
1D37 20584348
1D3B 47


1D3C 5043484C         DB      'PCHL','XTHL','RET ','HLT '
1D40 5854484C
1D44 52455420
1D48 484C5420


1D4C 434D4320         DB      'CMC ','STC ','CMA ','DAA '
1D50 53544320
1D54 434D4120
1D58 44414120


1D5C 52415220         DB      'RAR ','RAL ','RRC ','RLC '
1D60 52414C20
1D64 52524320
1D68 524C4320


 D6C 4E4F5020         DB      'NOP '
1D70 43504920 TAB2:   DB      'CPI ','ORI ','XRI ','ANI '
1D74 4F524920
```

```
1D78 58524920
1D7C 414E4920


1D80 53424920           DB      'SBI ','IN  ','SUI ','OUT '
1D84 494E2020
1D88 53554920
1D8C 4F555420


1D90 41434920           DB      'ACI ','ADI '
1D94 41444920
1D98 43414C4C TAB3:     DB      'CALL','JMP ','LDA ','STA '
1D9C 4A4D5020
1DA0 4C444120
1DA4 53544120


1DA8 4C484C44           DB      'LHLD','SHLD'
1DAC 53484C44

 DB0 4D4F5620 PMOV:     DB      'MOV '
1DB4 41444420 PADD:     DB      'ADD ','ADC ','SUB ','SBB '
1DB8 41444320
1DBC 53554220
1DC0 53424220


1DC4 414E4120           DB      'ANA ','XRA ','ORA ','CMP '
1DC8 58524120
1DCC 4F524120
1DD0 434D5020


1DD4 494E5220 PINR:     DB      'INR '
1DD8 44435220 PDCR:     DB      'DCR '
1DDC 4D564920 PMVI:     DB      'MVI '
1DE0 4C584920 PLXI:     DB      'LXI ','STAX','INX ','DAD '
1DE4 53544158
1DE8 494E5820
1DEC 44414420


1DF0 4C444158           DB      'LDAX','DCX '
1DF4 44435820

 DF8 52535420 PRST:     DB      'RST '
 DFC 50535720 PPSW:     DB      'PSW '
1E00 504F5020 PPOP:     DB      'POP ','PUSH'
1E04 50555348
```

```
1E08 4E5A5A20 CCODE:  DB      'NZ','Z ','NC','C '
1E0C 4E434320

1E10 504F5045         DB      'PO','PE','P ','M '
1E14 50204D20

0000                  END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB23

☐ 4004 ☐ 8008 ☒ 8080 ☐ 4040

(use additional sheets if necessary)

Program Title

BINLB -- 8080 System Loader

Function

Loads Hex Format Paper Tape produced by Macro-Assembler on GE Time Sharing into 8080 system. Also provides TTY input and output subroutines. BINLD can also produce a binary dump of itself for bootstrap loading.

Required Hardware

TTY -- This program uses a serial TTY interface and a hardware timer. This can be easily changed.

Required Software

NONE

Input Parameters

Hex format paper tape

Output Results

Loaded program

Registers Modified:	**Maximum Subroutine Nesting Level:** N/A
RAM Required: 256 Bytes	**Assembler/Compiler Used:** MACRO/8080
ROM Required: NONE	**Programmer:** George Miler
	Company: MILERTRONICS 525-A Airport Rd. Greenville, S.C. 29607

```
                ;REF. NO. AB23.
                ;PROGRAM NAME  8080 SYSTEM LOADER
                ;
                ;
                ;
                ;
                ;
                ; 4,22,75
                ;WRITTEN BY
                ;MILERTRONICS DIV.
                ;GEORGE MILER INC.
                ;525-A AIRPORT RD.
                ;GREENVILLE, S.C. 29607
                ;(803)242-9232
                ;BINLD  LOADS INTEL HEX FORMAT
                ;OBJECT TAPE ASSEMBLER
                ;REG B USED TO GENAERATE CHARACTER
                ;REG C USED TO FOR BUT IT COUNT 8SERIAL I/O)
                ;REG D USED FOR TEMP STORAGE
                ;REG E USED FOR CHARACTER COUNT
                ;
                ;
                ;
                ;
0E00                    ORG     0E00H
0E00 3EC9               MVI     A,0C9H          ;STORE A RET AT THE TTY
0E02 321800             STA     18H             ;INTERRUTPT ADDRESS-18H
0E05 F3         START:  DI                      ;DISABLE INTERRUPT
0E06 31100F             LXI     SP,0F10H        ;LOAD STACK POINTER
                ;BEGINNING OF MAIN BINLD PROGRAM
0E09 AF         BINLD:  XRA     A               ;CLEAR SUM
0E0A 32D10E             STA     CHECK           ;CHECK LOCATION.
0E0D CDA80E             CALL    TTY             ;GET CHARACTER
0E10 FE3A               CPI     3AH             ;CHECK FOR COLON
0E12 C2090E             JNZ     BINLD           ;GET NEXT CHARACTER IN NOT :
0E15 CDA80E             CALL    TTY             ;GET FIRST LENGTH HEX
0E18 50                 MOV     D,B             ;SAVE FIRST HEX
0E19 CDA80E             CALL    TTY             ;GET SECOOND LENGTH
0E1C CD600E             CALL    PACK            ;PACK ;LENGTH
0E1F B7                 ORA     A               ;STOP IF END
0E20 CAC90E             JZ      STOP
0E23 58                 MOV     E,B             ;STORE LENGTH IN REG E
0E24 CDA80E             CALL    TTY             ;GET 4 HEX ADDRESSES
0E27 50                 MOV     D,B
0E28 CDA80E             CALL    TTY             ;GET SECOND HEX
[E2B CD600E             CALL    PACK            ;AND PACK
 E2E 60                 MOV     H,B             ;MOVE LSB TO H
0E2F CDA80E             CALL    TTY             ;GET THIRD HEX
0E32 50                 MOV     D,B             ;
```

```
0E33 CDA80E             CALL    TTY             ;GET LAST ADDRESS HEX
0E36 CD600E             CALL    PACK
0E39 68                 MOV     L,B             ;MSB TO L
0E3A CDA80E             CALL    TTY             ;IGNORE NEXT
0E3D CDA80E             CALL    TTY             ;TWO CHARACTERS
                        ;
                        ;PACK AND STORE DATA IN MEMORY
                        ;
0E40 CDA80E     STORE:  CALL    TTY             ;READ
0E43 50                 MOV     D,B             ;DATA
0E44 CDA80E             CALL    TTY             ;PACK
0E47 CD600E             CALL    PACK            ;AND
0E4A 70                 MOV     M,B             ;STORE IN MEMORY
0E4B 23                 INX     H
0E4C 1D                 DCR     E               ;DECREMENT LENGTH
0E4D C2400E             JNZ     STORE           ;GET NEXT CHARACTER
0E50 CDA80E             CALL    TTY             ;READ SUM CHECK
0E53 50                 MOV     D,B
0E54 CDA80E             CALL    TTY
0E57 CD600E             CALL    PACK
0E5A C27B0E             JNZ     ERROR           ;HALT IF ERROR IS DETECTED
 E5D C3090E             JMP     BINLD           ;LOOK FOR NEXT FILELINE
                        ;

                        ;PACK TAKES DATA IN REG B AND D
                        ;AND PACKS INTO REG B
                        ;REG D CONTAINS MOST SIGNIFICANT HEX
                        ;
0E60 7A         PACK:   MOV     A,D             ;FIRST HEX TO ACC
0E61 FE3A               CPI     3AH             ;CHECK FOR COLON
0E63 CA7B0E             JZ      ERROR           ;ERROR IF CPOLON
0E66 07                 RLC                     ;SHIFT
0E67 07                 RLC                     ;LEFT
0E68 07                 RLC                     ;FOUR
0E69 07                 RLC                     ;BITS
0E6A 57                 MOV     D,A             ;SAVE HEX
0E6B 78                 MOV     A,B
0E6C FE3A               CPI     3AH             ;GO TO
0E6E CA7B0E             JZ      ERROR           ;ERROR IF COLON
0E71 B2                 ORA     D               ;PACK FIRST HEX.
0E72 47                 MOV     B,A             ;PACKED RESULTS INTO B
0E73 3AD10E             LDA     CHECK           ;COMPUTE CHECK SUM
0E76 80                 ADD     B
0E77 32D10E             STA     CHECK
0E7A C9                 RET
                        ;
                        ;ERROR HALT
                        ;HALT IF A COLON IS DETECTED IN THE DATA
                        ;HALT IF AN ERROR IS DETECTED
```

```
                        ;
0E7B 0613       ERROR:  MVI     B,13H           ;STOP TAPE READER
0E7D CDD20E             CALL    OUTPT
0E80 76                 HLT                     ;AND HALT THE LOADER
                        ;
                        ;
                        ;TIMER LOADS DELAY FROM ACC TO CCM TIMMER
                        ;INTERRUPTS ARE NOT USED--TIME IS IN MS
                        ;
0E81 D301       TIMER:  OUT     1               ;SEND TIME TO CCM
0E83 DB03       LOOP1:  IN      3               ;READ TIME FROM CCM
0E85 B7                 ORA     A               ;TEST TIME
0E86 C2830E             JNZ     LOOP1           ;WATE FOR TIME
0E89 C9                 RET
                        ;
                        ;TTY READS CHARACTER FROM TTY
                        ;USES SERIAL I/O
                        ;RETURNS ONLY HEX OR COLON
                        ;CONVERTS ASCII TO HEX
                        ;REG C USED FOR BIT COLOUNT
                        ;
 :8A FB         INPUT:  EI                      ;ENABLE INTERRUPTS
0E8B 76                 HLT                     ;AND WATE FOR TTY
0E8C F3                 DI                      ;DISABLE INTERRUPTS
0E8D 010900             LXI     B,9             ;CLEAR B, SET BIT COUNT TO 9
0E90 3E0D               MVI     A,13            ;SKIP 1.5 BIT TIMES
0E92 C39C0E             JMP     JUMP3
0E95 DB17       LOOP2:  IN      17H             ;INPUT FROM TTY
0E97 B0                 ORA     B               ;OR BIT INTO BYTE
0E98 0F                 RRC                     ;ROTATE RIGHT
0E99 47                 MOV     B,A             ;SAVE BYTE
0E9A 3E09               MVI     A,9             ;WATE FOR
0E9C CD810E     JUMP3:  CALL    TIMER           ;NEXT BIT
0E9F 0D                 DCR     C               ;DECREMENT BIT COUNTER
0EA0 C2950E             JNZ     LOOP2
0EA3 78                 MOV     A,B             ;SAVE CHARACTER
0EA4 E67F               ANI     7FH             ;REMOVE PARITY
0EA6 47                 MOV     B,A
0EA7 C9                 RET
0EA8 CD8A0E     TTY:    CALL    INPUT           ;GET NEXT CHARACTER
0EAB 78                 MOV     A,B             ;RETURN CHARACTER TO ACC
0EAC FE2F               CPI     2FH             ;IS CHARACTER LESS THEN 0
0EAE DAA80E             JC      TTY             ;LOOK FOR NEXT CHARACTER
0EB1 FE3A               CPI     3AH             ;CHECK FOR 0 TO 9
0EB3 DAC50E             JC      JUMP1
0EB6 CAC70E             JZ      JUMP2           ;JUMP2 IF COLON
 EB9 FE41               CPI     41H             ;CHECK FOR A
 EBB DAA80E             JC      TTY             ;GET NEXT CHARACTER
0EBE FE47               CPI     47H             ;GET NEXT CHARACTER
0EC0 D2A80E             JNC     TTY             ;IF GREATER THAN F
```

```
0EC3 C609              ADI     9               ;CORRECT HEX
0EC5 E60F     JUMP1:   ANI     0FH             ;REMOVE LEFT 4 BITS
0EC7 47       JUMP2:   MOV     B,A
0EC8 C9                RET
                ;
                ;
                ;PROGRAM LOADED CORRECTLY
                ;STOP TTY AND JUMP TO PROGRAM
                ;
0EC9 0613     STOP:    MVI     B,13H           ;STOP TTY
0ECB CDD20E            CALL    OUTPT
0ECE C30001            JMP     100H            ;STARTING ADDRESS OF PROGRAM
0ED1          CHECK:   DS      1               ;CHECK SUM STORAGE
                ;TTY OUTPUT SUBROUTINE
                ;
0ED2 0E09     OUTPT:   MVI     C,9             ;SET BIT COUNTER
0ED4 AF                XRA     A               ;CLEAR ACC
0ED5 D317              OUT     17H             ;SEND 0
0ED7 3E09     RT1:     MVI     A,9             ;SET TIMMER
0ED9 CD810E            CALL    TIMER
0EDC 78                MOV     A,B             ;DATA TO ACC
  :DD D317             OUT     17H             ;SEND TO TTY
0EDF 0F                RRC                     ;ROTATE DATA
0EE0 47                MOV     B,A             ;SAVE
0EE1 0D                DCR     C
0EE2 C2D70E            JNZ     RT1
0EE5 3E01              MVI     A,1             ;SEND MARK
0EE7 D317              OUT     17H             ;TO TTY
0EE9 3E1C              MVI     A,28
0EEB CD810E            CALL    TIMER
0EEE C9                RET
                ;BINARY DUMP PROGRAM
                ;PUNCHES BINARY MEMORY IMAGE
                ;OUTPT ROUTINE PUNCHES BINARY ON TTY
                ;MEMORY IMAGE IS IN REVERSE ORDER
                ;
0EEF 31FE0F   DUMP:    LXI     SP,0FFEH        ;SET STACK POINTER
0EF2 21FD0E            LXI     H,ENDPG ;START OF DUMP
0EF5 16FF              MVI     D,0FFH          ;LENGTH OF DUMP
0EF7 46       LOOP3:   MOV     B,M             ;LOAD BYTE
0EF8 CDD20E            CALL    OUTPT           ;PUNCH BYTE
0EFB 2B                DCX     H               ;DECREMENT ADDRESS
0EFC 15                DCR     D               ;DECREMENT COUNT
0EFD C2F70E   ENDPG:   JNZ     LOOP3           ;GET NEXT BYTE
0000                   END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB24

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

Boot

Function

To allow for bootstrap loading of programs and for patching of programs or data in memory via the teletype. The program uses less than 200 bytes of memory and may be placed in ROM or entered manually.

Required Hardware

8080, teletype and interface.

Required Software

None.

Input Parameters

The input format is described in comments at the beginning of the program listing.

Output Results

Code or data may be loaded into any locations in memory. Loading or patching of memory locations need not be in sequential order.

Registers Modified: All	**Assembler/Compiler Used:** MAC 80
RAM Required: x'9D' (157 bytes)	**Programmer:** Jeff Kravitz
ROM Required: Same as above	**Company:** Grumman Data Systems
Maximum Subroutine Nesting Level: 5	**Address:** 197 Fairhaven Blvd. Woodbury, N.Y. 11797

```
;REF. NO. AB24
;PROGRAM NAME BOOT
;
;*****************************************************************
;       BOOTLOADER & DEBUGGING TOOL FOR INTEL 8080 WITH TELETYPE
;=================================================================
;
;
;       THIS PROGRAM ACCEPTS INPUT FROM THE TTY KEYBOARD OR READ
;       IN HEXADECIMAL FORMAT.  IT WILL ENTER PROGRAMS OR DATA IN
;       ANY LOCATION IN MEMORY AND BRANCH TO ANY LOCATION.
;
;       THE FOLLOWING CHARACTERS HAVE SPECIAL MEANING TO THE PRO
;
;       :       (COLON) INDICATES THAT THE NEXT FOUR HEX DIGITS
;                       ARE TO BE USED AS THE ADDRESS OF ANY SUB
;                       DATA TO BE LOADED.  THE COLON MAY BE USED
;                       OF TIMES TO ENTER PATCHES INTO DIFFERENT
;                       MEMORY OR IF LOADING FROM PAPER TAPE, TO
;                       SPECIFY THE INTIAL LOAD ADDRESS AND ANY
;                       ADDRESSES.
;
;       T               INDICATES THAT THE INPUT IS COMING FROM
;                       TAPE.  THIS TURNS OFF ECHOING OF CHARACTE
;                       PRINTER.  (THIS WILL NOT WORK IF THE PROG
;                       ONLY MEMORY).  IF A PAPER TAPE CONTAINS A
;                       ITS FIRST CHARACTED, THE TTY WILL NOT EC
;                       THE T MAY ONLY BE USED AS THE FIRST CHAR
;                       THE PROGRAM ONLY CHECKS FOR IT ONCE.
;
;       G               INDICATES THAT THE NEXT FOUR HEX DIGITS
;                       TO BE USED AS A BRANCH ADDRESS.  THE PROG
;                       BRANCH TO THE SPECIFIED LOCATON.
;
;       X               INDICATES THAT THE LOADING OR PATCHING O
;                       IS COMPLETED.  THE PROGRAM WILL HALT WITH
;                       ENABLED.
;
;                       BLANKS AND CARRIAGE RETURNS MAY BE FREEL
;                       FOR CONVENIENCE AND READABLTY.  CARRIAGE
;                       WILL ECHO AS CARRIAGE RETURN, LINE FEED.
;
;
;=================================================================
;
;               A SAMPLE INPUT MIGHT BE AS FOLLOW:
;
;
;
```

```
                ;       :0100 A2F4 A345 9786 98 45 0909 4343 G 0400
                ;
                ;       THIS WILL PLACE 'A2E4 A354... ETC.' AT LOCATION 100
                ;       AND THEN BRANCH TO LOCATION 400 (ALL NUMBERS IN HEX)
                ;
                ;               ANOTHER EXAMPLE;
                ;
                ;       :0304 F2 :0309 46 :0807 87 X
                ;
                ;       THIS WOULD PLACE 'F2' AT 304, '46' AT 309, '87' AT 807 A
                ;
                ;
                ;*****************************************************************
0000 31B800     INIT:   LXI     SP,ENDP+10H             ; INITIALIZE THE STACK P
0003 210001             LXI     H,START                 ; LOAD DEFAULT STARTING
0006 CD5900             CALL    TESTX                   ; TEST FOR C'T' (NO ECHO
0009 CD6900     LOOP1:  CALL    TREAD                   ; READ, TEST, CONVERT, E
000C 77                 MOV     M,A                     ; STORE THE BYTE IN MEMO
000D 23                 INX     H                       ; INCREMENT THE LOAD ADD
000E C30900             JMP     LOOP1                   ; CONTINUE
                ;
                ;
0100            START   EQU     100H                    ; DEFAULT LOAD ADDRESS
                ;*****************************************************************
                ;*              GO ROUTINE
                ;*
                ;*****************************************************************
0011 CD7B00     GO:     CALL    GETHL                   ; GET H&L FROM TTY
0014 E9                 PCHL    ;BRANCH TO USER ADDRESS
                ;
                ;
                ;
                ;*****************************************************************
                ;*                                      ERROR ROUTINE
                ;*
                ;*****************************************************************
0015 063F       ERROR:  MVI     B,'?'                   ; TYPE A QUESTION MARK
0017 CD3200             CALL    OUTT
001A FB         ERR2:   EI
001B 76                 HLT
                ;*****************************************************************
                ;*
                ;*              READ AND ECHO SUBROUTINES
                ;*
                ;*****************************************************************
                ; NOTE THIS PROGRAM ASSUMES THE TTY IS AT 70 AND 71
                ; THE ADDRESSES IN THE NEXT TWO ROUTINES MUST BE MODIFIED
                ; IF THIS IS NOT THE CASE.
001C DB71       READ:   IN      71H             ;READ TTY STATUS BYTES BYTE
001E E6FB               ANI     0FFH-TBE        ;REMOVE THE 'TRANSMIT BUFFER EMP
```

```
0020 FE01           CPI     01H             ;TEST FOR DATA AVAILABLE BIT
0022 F21500         JP      ERROR           ;OTHER BITS ON, ERROR FROM TTY
0025 FA1C00         JM      READ            ;DATA NOT YET AVAIL.  KEEP TRYING
0028 DB70           IN      70H             ;READ A DATA BYTE
002A FE00           CPI     00H             ;SKIP BLANK LEADER TAPE
002C CA1C00         JZ      READ
                    ANI     7FH             ;TURN OFF 'PARITY' BIT
0031 47             MOV     B,A             ;PUT CHAR IN B FOR ECHO ROUTINE
                    ;
                    ;*****************************************************
                    ;*
                    ;*              ECHO SUBROUTINE
                    ;*
                    ;*****************************************************
0032 DB71           OUTT:   IN      71H             ;GET TTY STATUS
0034 E604           ANI     TBE             ;TEST FOR TRANSMIT BUFFER EMPTY
0036 CA3200         JZ      OUTT            ;NOT EMPTY, TEST AGAIN
0039 78             MOV     A,B             ;GET DATA BYTE IN A
003A D370           NOPE:   OUT     70H             ;OUT THE BYTE
003C FE20           CPI     ' '             ;TEST FOR A BLANK
003E CA1C00         JZ      READ            ;YES, READ AGAIN
0041 FE0A           CPI     LF              ;TEST FOR A LINE FEED
0043 CA1C00         JZ      READ            ;YES READ AGAINE FEED
0051 FE0D           CPI     CR              ;TEST FOR CARRIAGE RETURN
0053 C0             RNZ     ;NO, RETURN TO CALLER
0054 060A           MVI     B,LF            ;ECHO A LINE FEED
0056 C33200         JMP     OUTT
                    ;
0004                TBE     EQU     04H             ;TRANSMIT BUFFER EMPTY
000D                CR      EQU     0DH             ;CARRIAGE RETURN
000A                LF      EQU     0AH             ;LINE FEED
                    ;*****************************************************
                    ;*
                    ;*          SPECIAL CHARACTER TESTING ROUTINE
                    ;*
                    ;*****************************************************
0059 CD1C00         TESTX:  CALL    READ            ;READ A BYTE FROM THE TTY & ECHO
005C FE54           CPI     CHART           ;IS THE THE 'NO ECHO' CHAR?
005E C26C00         JNZ     TEST            ;NO, CONTINUE
0061 3E00           MVI     A,NOPI          ;NOP THE ECHO ROUTINE
0063 323A00         STA     NOPE

0066 323B00         STA     NOPE+1
0069 CD1C00         TREAD:  CALL    READ            ;READ ANOYHER THER CHARACTER

006C FE47           TEST:   CPI     CHARG           ;IS THIS THE "GO" CHARACTER
006E CA1100         JZ      GO              ;YES  'GO'
  71 FE58           CPI     CHARX           ;IS THIS THE 'DONE' CHARACTER
0073 CA1A00         JZ      ERR2            ;YES, HALT
0076 FE3A           CPI     CHARA           ;IS THIS THE 'ADDRESS' CHARACTER
```

```
0078 C28700             JNZ     CONV1           ;NO TREAT AS HEX DIGIT
007B CD8400     GETHL:  CALL    CONV            ;READ A 4 HEX DIGIT ADDRESS IN P
007E 67                 MOV     H,A             ;HI ORDER BYTE OF ADDRESS
007F CD8400             CALL    CONV            ;GET LOW ORDER ADDRESS
0082 5F                 MOV     L,A
0083 C9                 RET
                ;
                ;
0000            NOPI    EQU     00H             ;NOP INSTRUCTION
0054            CHART   EQU     54H             ;C'T' 'NO ECHO' CHAR
0047            CHARG   EQU     47H             ;C'G' 'GO' CHAR
0058            CHARX   EQU     58H             ;C'X'  'DONE' CHAR
003A            CHARA   EQU     3AH             ;C':' 'ADDRESS' CHAR
                ;*******************************************************************
                ;*
                ;*              ASCII TO HEX ASSEMBLY ROUTINE
                ;*
                ;*******************************************************************
0084 CD1C00     CONV:   CALL    READ            ;READ A BYTE FROM THE TTY & ECHO
0087 CD9500     CONV1:  CALL    TLATE           ;TRANSLATE THE BYTE FROM ASCII T
008A 87                 ADD     A               ;SHIFT LEFT 2
008B 87                 ADD     A               ;SHIFT LEFT ANOTHER 2
008C 5F                 MOV     E,A             ;SAVE IN B
008D CD1C00             CALL    READ            ;READ ANOTHER BYTE
0090 CD9500             CALL    TLATE           ;TRANSLATE FROM ASCII TO HEX
0093 B0                 ORA     B               ;COMBINE TWO HALF BYTES
0094 C9                 RET     ;RETURN TO CALLER
                ;
                ;
                ;*******************************************************************
                ;*
                ;*              ASCII TO HEX TRANSLATION ROUTINE
                ;*
                ;*******************************************************************
0095 D630       TLATE:  SUI     '0'             ;SUBTRACT THE LOWEST PERMISSABLE
0097 FA1500             JM      ERROR           ;TOO LOW, BAD HEX CHAR
009A FE0A               CPI     '9'-'0'+1       ;TEST FOR A 9
009C F8                 RM      ;BETWEEN ZERO AND NINE, DONE
009D D611               SUI     'A'-'0'         ;SUBTRACT THE DIFFERENCE TO A
009F FA1500             JM      ERROR           ;BETWEEN '0' AND 'A', ERROR
00A2 FE06               CPI     'F'-'A'+1       ;TEST FOR HIGHER THAN 'F'
00A4 F8                 RM      ;BETWEEN 'A' AND 'F', OK
00A5 C31500             JMP     ERROR           ;GREATER THAN 'F', ERROR
00A8            ENDP    EQU     $
0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB25

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

Octal PROM Programming

Function

See below

Required Hardware

Standard INTELLEC 8/MOD 80 with teletype

Required Software

System Monitor

Input Parameters

This program accepts sets of 3 octal numbers. The fourth character (unless it is a rubout) will cause the BYTES to be placed in memory starting at 100H. Any invalid octal character input in the first 3 positions will cause a carriage return and line feed to be output to the teletype and the line to be ignored. Any number of sets may be input (up to the practical limit of 100H). Whenever a "bell" is typed, the address of the last valid byte on Page 1 will be displayed on the register/flag lights and control will pass to the system monitor. To program the PROM, type P100, 1NN,0 where NN is the HEX number displayed on the lights.

Output Results

The program may be tested by inputting your choice of octal numbers and verifying that the PROM accepted them.

Registers Modified: All	**Assembler/Compiler Used:** INTELLEC 8/MOD 80 VER. 3.0
RAM Required: 6E	**Programmer:** Bob Gill
ROM Required: None	**Company:** Applied Physics Laboratory
Maximum Subroutine Nesting Level: 1 - plus whatever is in C1 and C0	**Address:** 1013 NE 40th Seattle, Washington 98195

```
                ;REF.  NO.  AB 25
                ;PROGRAM NAME OCTAL PROM PROGRAMMING
                ;
                ;
                ;
                ;
0010                    ORG     10H
0010 31F001             LXI     SP,1F0H         ;SET STACK POINTER
0013 1600               MVI     D,0                     ;ZERO # OF BYTES COUNTER
0015 210001             LXI     H,100H                  ;PLACE DATA BEGINNING AT
0018 0600       BB:     MVI     B,0                     ;ZERO ACCUMULATOR
001A 1E00               MVI     E,2-2                   ;SET LOOP COUNTER
001C CD8003     AA:     CALL    380H                    ;READ A CHR FROM TTY
001F 4F                 MOV     C,A
0020 77                 MOV     M,A                     ;SAVE IT
0021 CD0938             CALL    3809H                   ;ECHO IT
0024 7E                 MOV     A,M                     ;RELOAD CHAR
0025 E67F               ANI     7FH                     ;MASK TO 7 BITS
0027 FE7F               CPI     7FH
0029 CA5700             JZ      CRLF                    ;IGNORE LINE IF RUBOUT
 32C FE07               CPI     07H                     ;IS IT A BELL
002E CA6400             JZ      OUTY                    ;GO TO PROGRAMMING PORTI
0031 FE30               CPI     30H
0033 FA5700             JM      CRLF                    ;IGNORE LINE IF #<0
0036 FE38               CPI     38H
0038 F25700             JP      CRLF                    ;IGNORE LINE IF #>7
003B E607               ANI     7                       ;CONVERT ASCII TO BINARY
003D 80                 ADD     B
003E 47                 MOV     B,A                     ;ADD CHAR TO ACCUM
003F 7B                 MOV     A,E
0040 FE00               CPI     0
0042 CA4E00             JZ      FIN                     ;IF LOOP=0,HAVE LOOKED A
0045 78                 MOV     A,B
0046 07                 RLC
0047 07                 RLC
0048 07                 RLC
0049 47                 MOV     B,A                     ;MULT ACCUM BY 7
004A 1C                 INR     E                       ;INCREMENT LOOP COUNTER
004B C31C00             JMP     AA
004E CD0338     FIN:    CALL    3803H                   ;GET 4TH CHAR
0051 E67F               ANI     7FH                     ;MASK TO 7 BITS
0053 78                 MOV     A,B                     ;IF NON-RUBOUT, STORE AC
0054 77                 MOV     M,A
0055 14                 INR     D                       ;INCREMENT # OF BYTES CO
0056 23                 INX     H                       ;MOVE ADDRESS POINTTER-G
 357 0E0D       CRLF:   MVI     C,0DH                   ;OUTPUT CR
-059 CD0938             CALL    3809H
005C 0E0A               MVI     C,0AH                   ;OUTPUT  LF
005E CD0938             CALL    3809H
```

```
0061 C31800             JMP     BB
0064 7A         OUTY:   MOV     A,D
0065 3D                 DCR     A
0066 D3FF               OUT     255
                ;DISPLAT LAST ADDRESS ON PAGE 1 ON REGISTER/FLAG LIGHTS
0068 C30038             JMP     3800H
0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB26

☐ 4004 ☐ 4040 ☐ 8008 ☐ 8080 (use additional sheets if necessary)

Program Title

8080 IDLE Analyzer for Approximating CPU utilization

Function

Displays amount of time 8080 would have spent in an Idle loop. When RUN time is compared with Idle time, the percent of CPU utilization can be calculated. Time display is in memory, in ASCII.

Required Hardware

8080

Required Software

Routine to call "IDLE", Routine to restore Registers after an interrupt, Routine to call "Time Display" when job is complete

Input Parameters

NONE

Output Results

ASCII time, format:

bIDLEbTIMEbMMbSS.S

where m = minutes
s = seconds

Registers Modified: IDLE: A, B, H, L Time Display: ALL	**Assembler/Compiler Used:** 8080 MACRO ASSEMBLER Ver 1.0
RAM Required: 16 BYTES (Hex)	**Programmer:** J. William McGonigal
ROM Required: 99 BYTES (Hex)	**Company:** Sycor, Inc.
Maximum Subroutine Nesting Level: Uses 2 BYTES of stack N/A	**Address:** 100 Phoenix Drive Ann Arbor, MI 48103

```
                        ;REF. NO. AB26.
                        ;PROGRAM 8080 IDLE ANALYZER FOR APPROXIMATING CPU UTILIZATION
                        ;
                        ;


                        ;
                        ;
                        ;       8080 IDLE ANALYZER FOR APPROXIMATING
                        ;          CPU UTILIZATION
                        ;
                        ;               J W MCGONIGAL
                        ;                  MAY 7, 1975
                        ;       THE FIRST ROUTINE (IDLE), SHOULD BE CALLED WHENEVER
                        ;       THE 8080 IS WAITING FOR AN EVENT (INTERRUPT) TO OCCUR.
                        ;
                        ;       WHEN THE JOB HAS COMPLETED PROCESSING, CALL
                        ;       "TIME DISPLAY" (TIMED).  IDLE TIME WILL BE STORED IN
                        ;       MEMORY, IN ASCII, STARTING AT LOCATION "ITIME".
                        ;       MAXIMUM TIME IS 54 MINUTES, 36.7 SECONDS.
                        ;       RESOLUTION IS 0.1 SECONDS.
                        ;
                        ;       USER IS RESPONSIBLE FOR SAVING REGISTERS AGTER AN
                        ;       INTERRUPT, AND IS ALSO RESPONSIBLE FOR RESTORING THEM
                        ;       BEFORE AGAIN CALLING "IDLE".    NEITHER "IDLE" NOR
                        ;       "TIMED" REQUIRE INITIAL PARAMETERS.    STACK POINTER IS
                        ;       ASSUMED TO BE PRE-INITIALIZED FOR "TIMED".
                        ;
                        ;
0100                            ORG     100H            ;MAY BE AT ANY LOCATION
                        ;       *** IDLE ROUTINE ***
                        ;          R(A) = 1.92 MILLISECOND COUNTER
                        ;          R(B) = 7.5 MICROSECOND COUNTER
                        ;          R(H,L) = 50 MILLISECOND COUNTER
                        ;          OTHER REGISTERS, STACK, AND STACK POINTER ARE
                        ;             NEITHER USED NOR MODIFIED.
                        ;
                        ;       *** TIME DISPLAY ***
                        ;          CALLS CLOCK CONVERSION ROUTINE (INCLUDED)
                        ;          R(A) = WORK REGISTER
                        ;          R(B,C) = MASKS
                        ;          R(D,E) = BINARY COUNTER
                        ;          R(H,L) = DATA POINTER
                        ;          USES STACK POINTER AND THE STACK
                        ;          REQUIRES TWO BYTES OF STACK SPACE
                        ;
                        ;
                        ;          THIS SECTION MAY BE IN ROM
0100 3A9001     IDLE:   LDA     FLAG1           ;INITIALIZE
0103 C600               ADI     0               ;ACTIVATE CONDITION FLAGS
```

```
0105 C21501            JNZ     IDLE1           ;JUMP IF NOT FIRST PASS
0108 3C                INR     A
0109 329001            STA     FLAG1           ;TURN FLAG ON
010C 0600              MVI     B,0             ;ZERO R(B)
010E 210000            LXI     H,0             ;ZERO R(H,L)
0111 229101            SHLD    BINCT           ;ZERO THE BINARY COUNTER
0114 AF                XRA     A               ;CLEAR THE ACCUMULATOR
0115 04        IDLE1:  INR     B               ;INCREMENT R(B)
0116 C21501            JNZ     IDLE1           ;LOOP FOR 1.92 MILLISECONDS
0119 3C                INR     A               ;INCREMENT THE ACCUMULATOR
011A FE1A              CPI     26D             ;CHECK FOR 26 PASSES
011C C21501            JNZ     IDLE1           ;LOOP IF NOT 49. 92 MILLISECONDS
011F 23                INX     H
0120 229101            SHLD    BINCT           ;INCR 50 MILLISECOND COUNTER
0123 C3FF00            JMP     IDLE -1         ;LOOP
                ;
                ;       TIME DISPLAY
                ;
0126 3E30      TIMED:  MVI     A,'0'           ;RESET DISPLAY TO ASCII ZERO
0128 219D01            LXI     H,IMINT         ;SET H,L TO TENS OF MINUTES
012B 77                MOV     M,A             ;ZERO TENS OF MINUTES
 12C 23                INX     H               ;BUMP POINTER TO MINUTES
012D 77                MOV     M,A             ;ZERO MINUTES
012E 23                INX     H               ;SKIP THE COLON
012F 23                INX     H               ;POINT TO TENS OF SECOND
0130 77                MOV     M,A             ;ZERO
0131 23                INX     H
0132 77                MOV     M,A             ;ZERO SECOND
0133 23                INX     H               ;SKIP THE PERIOD
0134 23                INX     H
0135 77                MOV     M,A             ;ZERO TENTH OF SECONDS
                ;           START TIME CONVERSION
0136 2A9101            LHLD    BINCT           ;GET BINARY TIME
0139 EB                XCHG                    ;PUT IN R(D,E)
013A 219D01            LXI     H,IMINT         ;SET POINTER TO TENS OF MINUTES
013D 01E02E            LXI     B,2EE0H         ;TENS OF MINUTES MASK
0140 CD7701    TIME1:  CALL    CCONV           ;CALL CLOCK CONVERTER


0143 CA4B01            JZ      TIME2           ;JUMP IF NO TENS OF MINUTES
0146 86                ADD     M               ;INCREMENT DISPLAY COUNT
0147 77                MOV     M,A             ;STORE
0148 C34001            JMP     TIME1           ;LOOP
014B 23        TIME2:  INX     H               ;ADVANCE TO MINUTES
014C 01B004            LXI     B,04B0H         ;MASK FOR MINUTES
014F CD7701    TIME3:  CALL    CCONV           ;CALL THE CLOCK CONVERTER
 152 01C800            LXI     B,00C8H         ;MASK FOR TENS OF SECONDS
 155 CD7701    TIME5:  CALL    CCONV           ;CALL CLOCK CONVERTED
0158 CA6001            JZ      TIME6           ;JUMP IF NO TENS
015B 86                ADD     M
```

```
015C 77                 MOV     M,A             ;INCREMENT DISPLAY COUNT
015D C35501             JMP     TIME5           ;LOOP
0160 23         TIME6:  INX     H               ;ADVANCE TO SECONDS
0161 011400             LXI     B,0014H         ;MASK FOR SECONDS
0164 CD7701     TIME7:  CALL    CCONV           ;CALL CLOCK CONVERTER
0167 CA6F01             JZ      TIME8           ;JUMP IF NO SECONDS
016A 86                 ADD     M
016B 77                 MOV     M,A             ;INCREMENT DISPLAAYY  COUNT
016C C36401             JMP     TIME7           ;LOOP
016F 23         TIME8:  INX     H
0170 23                 INX     H               ;ADVANCE TO TENTHS OF SECONDS
0171 AF                 XRA     A               ;CLEAR ACCUMULATOR AND THE CARRY
0172 7B                 MOV     A,E             ;GET REMAINING BINARY TIME
0173 1F                 RAR                     ;SHIFT OFF HUNDRETHS
0174 86                 ADD     M
0175 77                 MOV     M,A             ;STORE TENTHS OF SECONDS SECONDS
0176 C9                 RET                     ;RETURN TO USER
                ;
                ;           CLOCK CONVERTER
                ;
                ;       RETURN: A=0 IF DATA LESS THAN MASK
                ;               A=1 IF DATA GR.  THAN MASK, ALSO PERFORMS
                ;                SUBTRACTION
                ;
                ;       RETURN/ENTER WITH: R(B,C) = 16 BIT MASK
                ;                          R(D,E) = 16 BIT DATA
                ;
                ;
0177 7A         CCONV:  MOV     A,D
0178 B8                 CMP     B
0179 DA8401             JC      CCON1           ;RETURN A=0 IF A LT.  B
017C C28601             JNZ     CCON2           ;JUMP TO SUBTRACT IF A GT.  B
017F 7B                 MOV     A,E             ;A = B,  CHECK LOWER BYTE
0180 B9                 CMP     C
0181 D28601             JNC     CCON2           ;JUMP IF A GT.  OR EQ.  C
0184 AF         CCON1:  XRA     A               ;ZERO THE ACCUMULATOR
0185 C9                 RET                     ;RETURN
0186 AF         CCON2:  XRA     A               ;CLEAR A AND CARRY BIT
0187 7B                 MOV     A,E             ;GET DATA
0188 99                 SBB     C               ;SUBTRACT MASK
0189 5F                 MOV     E,A             ;SAVE IT
018A 7A                 MOV     A,D
018B 98                 SBB     B               ;DO UPPER BYTE
018C 57                 MOV     D,A             ;STORE IT
018D AF                 XRA     A               ;ZERO ACCUMULATOR
018E 3C                 INR     A               ;SET A=1
018F C9                 RET                     ;RETURN
                ;
                ;.
                ;       THIS SECTION MUST BE IN RAM AND CAN BE RELOCATED
```

```
                        ;       BY AN ORG STAEMENTTEMENT OF INCLUSION WITH OTHER CODE.
                        ;
0190 00         FLAG1:  DB      0               ;FIRST PASS FLAG
0191 00         BINCT:  DB      0               ;BINARY 50 MILLISECOND COUNTER
0192 00                 DB      0
0193 2049444C   ITIME:  DB      ' IDLE TIME'    ;(MAY BE MODIFIED OR OMITTED)
0197 45205449
019B 4D45
019D 20         IMINT:  DB      ' '             ;TENS/UNITS OF MINUTES
019E 20                 DB      ' '             ;SPACE
019F 20         ISECS:  DB      ' '             ;TENS/UNITS OF SECONDS
01A0 20                 DB      ' '             ;PERIOD
01A1 20         ITENTH: DB      ' '             ;TENTHS OF SECONDS
01A2 20                 DB      ' '             ;BLANK PADDING
0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB27

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

REAL TIME EXECUTIVE

Function

PERFORMS PROCESSOR INITIALIZIATION, PERIODIC AND DEMAND SCHEDULING, ROUTINE TERMINATION, AND WAITS DURING IDLE TIME.

Required Hardware

8080, TIMER THAT CAUSES A PERIODIC INTERRUPT (i.e. 1ms), ROM, AND RAM

Required Software

NONE

Input Parameters

Output Results

ROUTINE	*PARAMETER*	*REGS*	*RESULT*
?CLKI	*CLOCK INTERRUPT*	----	*?CLOK IS INCREMENTED & PERIODIC ROUTINES MAY BE ADDED TO DISPATCH QUEUE*
?PERQ	*ROUTINE ADDRESS* *TIME*	*D,E* *A*	*THE PASSED ROUTINE IS ADDED TO THE TIME QUEUE TO BE ACTIVATED AT TIME IN A.*
?INIT	*ROUTINE ADDRESS*	*D,E*	*ADDS ROUTINE TO LIFO DISPATCH QUEUE*
?DSPC	----	----	*EXITS TO NEXT TASK ON QUEUE*
?WAIT	----	----	*WAITS FOR AN INTERRUPT (SYSTEM IDLE)*
?INTP	----	----	*CLEARS RAM AND SETS UP QUEUES AND STACKS*

Registers Modified: *ALL*	**Assembler/Compiler Used:** *8080 MDS MACRO ASSEMBLER -1.0*
RAM Required: *22 BYTES MIN → 42 BYTES REC.*	**Programmer:** *TED J. CLOWES*
ROM Required: *0FEH = 254 BYTES*	**Company:** *CUBIC CORP.*
Maximum Subroutine Nesting Level: *2*	**Address:** *4285 PONDEROSA SAN DIEGO, CA. 92123*

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AC9

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

Read/write Routines for Interchange Tapes

Function

Subroutines read and write blocks and characters for any common audio cassette recorder. Variable redundancy allows high-speed or highly reliable operation.

Required Hardware

8080 with one-chip cassette tape interface

Required Software

Self-sufficient

Input Parameters

Block I/O: Specify array address and block length
Character I/O: Specify word size

Output Results

Writer: Recorded cassette tape
Reader: Blocks of data in memory

Registers Modified: All	**Assembler/Compiler Used:** Macro Assembler
RAM Required: User's data	**Programmer:** J.L. Ogdin
ROM Required: 306 bytes	**Company:** Microcomputer Technique, Inc.
Maximum Subroutine Nesting Level: 3	**Address:** 11227 Handlebar Road Reston, Va. 22091

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB28

☐ 4004 ☐ 4040 ☒ 8008 ☐ 8080

(use additional sheets if necessary)

Program Title

PROPORTIONAL POWER CONTROL IMAGE BUILDER

Function

This program builds an "ON-OFF" image in RAM to allow proportional power control using zero crossing solid state relays. After an image is built using this program, consecutive image words are accessed by an interupt routine (not shown here) that causes solid state zero crossing relays to turn on and off as the interupt program steps through the RAM image. Assuming a 60 Hz line interupt frequency, one complete image cycle takes 256/60 = 4.4 sec. The image is built so that power is applied nearly evenly throughout the cycle. For example, the most significant value bit controls every other word; the next most significant bit in a value word controls every other remaining word and so on. Each value word controls one output bit stream. The present program is set to take 32 value words of 8 bits each, and produce a switching image: 32/8 x 100H = 400H locations long to control 4 output ports or 32 relays.

Registers Modified:	**Assembler/Compiler Used:** PLM1, PLM2
RAM Required: 130 + p (100H) p = # of ports used	**Programmer:** R. Yodlowski/617-890-2000 X494
ROM Required: 15AH	**Company:** LFE Corporation
Maximum Subroutine Nesting Level: 0	**Address:** 1601 Trapelo Rd. Waltham, Mass. 02154

98-034B

```
00001  1
00002  1   /*REF. NO. AB28 */
00003  1   /*PROGRAM TITLE PROPORTIONAL POWER CONTROL IMAGE BUILDER */
00004  1
00005  1   200H: DECLARE VALUE (32) BYTE;
00006  1
00007  1   DECLARE MSB$USED LITERALLY '7';
00008  1
00009  1   DECLARE PWRIMBRET ADDRESS;
00010  1
00011  1
00012  1
00013  1   /*  PWRIMB.RY  PWRIMB.RY    APRIL 9, 1975*/
00014  1
00015  1
00016  1   DO;    /*****POWER IMAGE BUILDER****/
00017  1
00018  1
00019  1   /* THIS PROGRAM SEGMENT DEVELOPS A BIT-BY-BIT RAM IMAGE
00020  1
00021  1   OF THE POWER PATTERNS TO BE APPLIED TO SOLID STATE
00022  1
00023  1   RELAYS TO ACHIEVE SYNCHRONUS SWITCHING PROPORTIONAL
00024  1
00025  1   POWER CONTROL.  THE DESIRED POWER LEVELS ARE MADE
00026  1
00027  1   AVAILABLE IN A BYTE VECTOR:  VALUE (N) .  EACH VALUE
00028  1
00029  1   SPECIFIES THE NUMBER OF  ON  POWER CYCLES OUT OF A
00030  1
00031  1   POSSIBLE FULL POWER MAXIMUM OF -1+2**(MSB$USED+1)
00032  1
00033  1   CYCLES.  */
00034  1
00035  1
00036  1
00037  1   DECLARE (  /*POWER IMAGE BUILDER VARIABLES*/
00038  2
00039  2      STARTOFPORT, /* POINTS TO 1ST VALUE OF PORTS. */
00040  2
00041  2    BITSEL,  /* SELECTS BITS FROM VALUE BYTES  */
00042  2
00043  2    BITNBR,  /* NUMBER OF BIT WITHIN VALUE BYTE */
00044  2
00045  2    BIT$HOLDER,  /* HOLDS BITS SELECTED FROM VALUES */
00046  2
00047  2    PORTNBR,   /* NUMBER OF PORT  */
00048  2
00049  2    RLYOFPORT,  /* WHICH RLY OF CURRENT PORT */
00050  2
00051  2    I,           /* HOLDER VARIABLE  */
00052  2
00053  2    STRT$CYC,     /*  CYCLES FROM IMAGE TO START */
00054  2
00055  2    INCR$CYC)     /*  OF CYCLES BETWEEN PATTERNS */
00056  2
00057  2   BYTE;
00058  2
00059  2   DECLARE CYCSCTR ADDRESS;  /* CYCLE COUNT DOWN IN IMAGE*/
00060  2
```

```
00061  2    DECLARE IMAGE (1024) BYTE;
00062  2
00063  2
00064  2    /* FOR EACH PORT */
00065  2
00066  2    DO STARTOFPORT = 0 TO LAST(VALUE) BY 8;
00067  2
00068  2     PORTNBR = SHR (STARTOFPORT, 3);
00069  3
00070  3     BITSEL = 1000$0000B;
00071  3
00072  3
00073  3     /* FOR EACH BIT OF VALUE USED */
00074  3
00075  3     DO BITNBR = 0 TO MSB$USED;
00076  3
00077  3                BIT$HOLDER = 0;
00078  4
00079  4                BITSEL = ROL (BITSEL, 1) ;
00080  4
00081  4
00082  4                /* FOR EACH OUTPUT RELAY OF THE PORT */
00083  4
00084  4                DO RLYOFPORT = 0 TO 7;
00085  4
00086  4                    BIT$HOLDER = ROL  (BIT$HOLDER, 1);
00087  5
00088  5
00089  5                    /* COLLECT THE CORESP. BITS INTO BIT HOLDER */
00090  5
00091  5                    BIT$HOLDER = BIT$HOLDER OR (BITSEL
00092  5
00093  5                        AND VALUE (STARTOFPORT +
00094  5
00095  5                        RLYOFPORT));
00096  5
00097  5     END;
00098  4
00099  4     /* CORRECT BIT POSITIONS OF BIT$HOLDER */
00100  4
00101  4     IF BITNBR <>0
00102  4
00103  4                      THEN BIT$HOLDER = ROR (BIT$HOLDER, BITNBR);
00104  4
00105  4     /* COMPUTE START CYCLE NUMBER */
00106  4
00107  4           I = 7 - MSB$USED + BITNBR;
00108  4
00109  4     STRT$CYC = SHR (255, 1 +1);
00110  4
00111  4           /* COMPUTE CYCLE INCREMENT */
00112  4
00113  4           IF I <> 0
00114  4
00115  4                     THEN INCR$CYC = 1 + SHR (255,I);
00116  4
00117  4                     ELSE INCR$CYC = 255;
00118  4
00119  4
00120  4     /* PLACE BIT$HOLDER INTO THE (STRT$CYC) TH IMAGE
```

```
00121   4
00122   4                          AND EVERY (INCR$CYC) IMAGES THEREAFTER*/
00123   4
00124   4     DO CYC$CTR = STRT$CYC TO 255 BY INCR$CYC;
00125   4
00126   4                         IMAGE ((256*PORTNBR) + CYC$CTR) = BIT$HOLDER;
00127   5
00128   5     END; /* PLACEMENT OF BIT$HOLDER */
00129   4
00130   4          END; /* BITS OF VALUE USED */
00131   3
00132   3    END; /* PORTS  */
00133   2
00134   2    END; /****POWER IMAGE BUILDER****/
00135   1
00136   1    GO TO PWRIMBRET;
00137   1
00138   1
00139   1    EOF;
NO PROGRAM ERRORS
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB29

☐ 4004 ☐ 4040 ☒ 8008 ☐ 8080

(use additional sheets if necessary)

Program Title

FLAG PROCESSING ROUTINE

Function

A Routine for contact closure debouncing and processing

1. Simultaneously check 8 input flags to find those flags which have been present for 3 successive iterations of routine.
2. Report flags identified above.
3. Clear reported flags that have been serviced.
4. Remember serviced flags until input flags are cleared.

Required Hardware

Input Interface (s) to bring in Contact Closures (Flags)

Required Software

Routine to assemble up to 8 flags into an 8 bit word before each iteration of this routine.
Routine to recycle thru this routine as frequently as desired.

Input Parameters

8 bit word at RAM location "Flags" assembled with latest State of Contact Closures.
8 bit word at RAM location "Flags + 4" showing actions to be deleted from "Do" list.

Output Results

8 bit word at RAM location "Flags +5" showing actions "to do". This is a normally used output.
8 bit word at RAM locations "Flags + 3" showing actions needed.

Registers Modified: ALL	**Assembler/Compiler Used:** Intellec 8/mode Macro assembler Ver 2.0
RAM Required: 6 bytes	**Programmer:** B. Weston
ROM Required: 49 bytes	**Company:** TRANSCOM INCORPORATED
Maximum Subroutine Nesting Level: none	**Address:** 580 Spring Street Windsor Locks, Comm. 06096

```
                ;REF. NO. AB29
                ;PROGRAM TITLE FLAG PROCESSING ROUTINE
                ;
                ;
                ;
                ;
                ;
                ;
                ; INSTRUCTIONS
                ;   1. ASSEMBLE INPUT FLAGS INTO MEMLOC "FLAGS"
                ;   2. "OR" IN FLAGS THAT HAVE BEEN SERVICED
                ;      INTO MEMLOC "FLAGS + $"
                ;      INTO MEMLOC "FLAGS + 4"
                ;   3. RUN ROUTINE
                ;   4. WORD IN MEMLOC "FLAGS + 5" REPORTS BIT
                ;      POSITION OF FLAGS WHICH NEED TO BE
                ;      SERVICED
                ;   5. WORD IN MEMLOC "FLAG + 3" REPORTS BIT
                ;      POSITION OF FLAGS WHICH HAVE BEEN
                ;      PRESENT FOR 3 OR MORE SUCCESSIVE
                ;      ROUTINE ITERATIONS AND ARE STILL PRESENT
                ;      OR HAVE NOT BEEN REPORTED TO HAVE BEEN
                ;      SERVICE.

0041            FLAG    EQU     41H
0141            FLAGS   EQU     141H
002E                    ORG     002EH
                ;
                ;       FLAG PROCESSING ROUTINE
                ;
002E 214101             LXI     H,FLAGS         ;POINTER TO FLAG
0031 7E                 MOV     A,M             ;STSORE CURRENT WORD IN A
0032 47                 MOV     B,A             ;AND ALSO IN B
0033 2C                 INR     L               ;POINTER TO PAST WORD
0034 AE                 XRA     M               ;FIND PAST/CURRENT CHANGES
0035 A6                 ANA     M               ;FIND DELETED FLAGS
0036 4F                 MOV     C,A             ;SAVE DELETED FLAGS IN C
0037 AE                 XRA     M               ;REMOVE DELETED FLAGS FROM
                                                ;PAST WORD
0038 77                 MOV     M,A             ;RE'STORE' CORRECTED PAST WORD
0039 2C                 INR     L               ;POINTER TO PREVIOUS PAST WORD
003A 79                 MOV     A,C             ;BRING IN DELETED FLAGS
003B EEFF               XRI     0FFH            ;INVERT
003D A6                 ANA     M               ;DELETE FROM PREVIOUS PAST
                                                ;WORD
003E 57                 MOV     D,A             ;SAVE RESULT IN D
003F 2E45               MVI     L,FLAG+4        ;POINT TO ACTION TAKEN
0041 7E                 MOV     A,M             ;STORE IN A
0042 2D                 DCR     L               ;POINT TO ACTION NEEDED
```

```
                ;     RUN ROUTINE AGAIN, MEMORY WILL BE
                ;CHANGED TO-

                ;         F8 F8 F8 7E 08 76

                ;         REPEAT ROUTINE AGAIN, RESULTING IN-

                ;         F8 F8 F8 FE 08 F6

                ;         CONTINUED EXECUTIONS WILL RESULT IN
                ;NO FURTHER CHANGES.
0000                    END
```

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/4040 ☒ 8008 ☐ 8080 ☐ 8048 ☐ 8085 ☐ Other _____ (use additional sheets if necessary)

Program Title

SOFTWARE STACK ROUTINES FOR 8008

Function

Subroutines provided for PUSH, POP, and EXCHange A with top of stack, and to save processor state on stack in case of an interrupt, and to restore it again.

Required Hardware

NONE

Required Software

NONE

Input Parameters

E register must be initialized to empty stack (e.g. 255) at beginning of main program.

Output Results

H and L registers left pointing to top of stack.

Registers Modified: A, (D), E, H, L	Programmer: Tom Pittman
RAM Required: 4-255 Bytes (for stack)	Company: MicroComputer Consultant
ROM Required: 84_{10} Bytes	Address: PO 23189
Maximum Subroutine Nesting Level: -0-	City: San Jose
Assembler/Compiler Used: Macro 8	State: California 95153

SOFTWARE STACK ROUTINES FOR 8008

To permit interrupts in a 8008 microcomputer system requires either extra hardware for saving and restoring the processor status external to the CPU, or two dedicated registers, used only in the service of the interrupt (for holding the H and L registers, so that the processor state may be stored in memory).

The subroutines presented here are designed to get extra mileage out of one of those dedicated registers by designating it as a stack pointer for a stack which may be used by the main routine as well as the interrupt service routine. If the system has the additional hardware to disable interrupts during the save and restore sequences, the interrupt service routines themselves may also be interruptible, to the maximum subroutine nest depth allowed by the main routine and all nested interrupts.

Two categories of subroutines are presented here. The first is a set of subroutines to push, pop, and exchange the A register with the top of the stack. These routines are re-entrant, and are designed to be callable with a RST instruction, though of course they may be placed anywhere in memory to be called with the CAL instruction.

The second category consists of the processor state save and restore routines, which may be called from or straight line coded into the beginning and end of the interrupt service routine. These routines are not re-entrant, since they use the dedicated registers. If interrupts may be nested, they must be disabled during these save and restore routines. These routines assume an arbitrary processor state, and save/restore the flags, and the A, H, and L registers in the stack. If it is desired to save the B and C registers in the stack also, the instructions to do this are easily added.

In these subroutines, the D register is dedicated for the interrupt save/restore routines, and may not be used for any other purpose while interrupts are enabled. The E register is designated the stack pointer, and always points to the first available empty cell in memory adjacent to the top of the stack. It is assumed that the stack is wholely contained within a defined page in RAM, designated "PAGE" in the source code; this value is loaded into the H register for all stack operations. All stack operations (except the restore sequence) leave the H and L registers pointing to the top of the stack (i.e. the item last pushed or next to be popped), to facilitate stack arithmetic.

Using the stack for temporary storage effectively compensates for the loss of the D and E registers for general programming.

```
                        ;REF.  NO.  AD5
                        ;PROGRAM NAME SOFT STACK ROUTINE FOR 8008
                        ;
                        ;
                        ;
                        ;
                        ;    SOFTWARE STACK SUBROUTINES FOR 8008
                        ;
0010                    PAGE    SET     16              ; PAGE IN RAM OF STACK
0028                            ORG     40              ; FOR USE AS RST 5
0028 00                         NOP
0029 00                         NOP
                        ;
                        ;    STACK PUSH  (RE-ENTRANT)
                        ;
002A 2610               PUSH1:  MVI     H,PAGE          ; SET PAGE OF STACK
002C 6B                         MOV     L,E             ; GET POINTER
002D 1D                         DCR     E               ; DECREMENT STACK POINTER
002E 77                         MOV     M,A             ; STORE A IN STACK
002F C9                         RET
                        ;
                        ;    STACK POP  (RE-ENTRANT)
                        ;
0030 2610               POP1:   MVI     H,PAGE          ; SET PAGE OF STACK
0032 6B                         MOV     L,E             ; GET POINTER
0033 2C                         INR     L               ; POINT TO STACK TOP
0034 7E                         MOV     A,M             ; FETCH IT INTO A
0035 5D                         MOV     E,L             ; UPDATE STACK POINTER
0036 2C                         INR     L               ; LEAVE L AT STACK TOP
0037 C9                         RET
                        ;
                        ;    EXCHANGE A WITH STACK TOP  (RE-ENTRANT)
                        ;
0038 2610               XCHA:   MVI     H,PAGE          ; SET PAGE OF STACK
003A 6B                         MOV     L,E             ; GET POINTER
003B 1D                         DCR     E               ; DECREMENT STACK POINTER
003C 77                         MOV     M,A             ; SAVE OLD A
003D 2C                         INR     L               ; POINT TO OLD TOP
003E 7E                         MOV     A,M             ; FETCH IT
003F 6B                         MOV     L,E             ; PUSH INTO STACK.
0040 1D                         DCR     E               ; (AT NEW TOP)
0041 77                         MOV     M,A
0042 2C                         INR     L               ; RETRIEVE OLD A
0043 7E                         MOV     A,M             ;
0044 2C                         INR     L               ; STORE IN OLD TOP
0045 77                         MOV     M,A
  46 2D                         DCR     L               ; RECOVER NEW A
0047 2D                         DCR     L
0048 7E                         MOV     A,M
```

```
0049 2C              INR     L               ; DISCARD TEMP CELLS
004A 5D              MOV     E,L             ; AT STACK TOP
004B 2C              INR     L               ; LEAVE L AT STACK TOP
004C C9              RET
                ;
                ;    PUSH A,H,L, & FLAGS INTO STACK
                ;
004D 55         INTS:   MOV     D,L             ; SAVE L IN (DEDICATED) D
004E 6B              MOV     L,E             ; GET STACK POINTER
004F 5C              MOV     E,H             ; SAVE H IN E
0050 2610            MVI     H,PAGE          ; SET PAGE OF STACK
0052 77              MOV     M,A             ; PUSH A
0053 3E00            MVI     A,0             ; ENCODE FLAGS:
0055 CA5F00          JZ      $+10            ; ZERO
0058 3EA0            MVI     A,160           ; NONZERO
005A FA5F00          JM      $+5             ; MINUS
005D 3E60            MVI     A,96            ; PLUS
005F 1F              RAR                     ; CARRY (SHIFTED IN )
0060 EA6500          JPE     $+5             ; EVEN PARITY
0063 F608            ORI     8               ; ODD PARITY
0065 2D              DCR     L               ; PUSH INTO STACK
0066 77              MOV     M,A
0067 2D              DCR     L               ; NOW PUSH H
0068 73              MOV     M,E             ; (WAS IN E)
0069 2D              DCR     L               ; *PUSH L*
006A 72              MOV     M,D             ; *REQUIRED ONLY IF NESTED*
006B 5D              MOV     E,L             ; PUT STACK POINTER
006C 1D              DCR     E               ; BACK INTO E
006D C9              RET
                ;
                ;    POP A,H,L, & FLAGS FROM STACK
                ;
006E 2610       INTR:   MVI     H,PAGE          ; SET PAGE OF STACK
0070 6B              MOV     L,E             ; GET POINTER
0071 2C              INR     L
0072 56              MOV     D,M             ; *POP L VALUE INTO D*
0073 2C              INR     L               ; *OMIT IF NOT PUSHED*
0074 5E              MOV     E,M             ; POP H VALUE INTO E
0075 2C              INR     L
0076 7E              MOV     A,M             ; POP FLAGS INTO A
0077 2C              INR     L
0078 87              ADD     A               ; SET FLAGS
0079 7E              MOV     A,M             ; POP A FROM STACK
007A 63              MOV     H,E             ; RE-ARRANGE REGISTERS:
007B 5D              MOV     E,L             ; STACK POINTER TO E FROM L
007C 6A              MOV     L,D             ; H&L FROM E&D
007D C9              RET

0000                 END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB30

☐ 4004 ☐ 4040 ☒ 8008 ☐ 8080

(use additional sheets if necessary)

Program Title

Symbol Table List Routine

Function

This program will print the user's symbols in alphabetical order followed by the address the 8008 Macro Assembler has assigned to each symbol.

Required Hardware

It can be run after pass 1 of the Assembler. The remaining passes, if any, can then be run without rerunning pass 1. Either the V2.0 or the V2.2 equate statements can be set equal to 1 or 0 to assemble the program for use with either version 2. or 2.2 of the Intellec 8 Monitor.

ASR 33 TTY, teletype interface to Intellec 8 Mod 8

Required Software

Intellec 8 Assembler and Monitor

Input Parameters

None

Output Results

After running pass 1 of the Assembler when this routine is called, it will print a table of user's symbol addresses in alphabetical order.

Registers Modified: All	**Assembler/Compiler Used:** MAC 8
RAM Required: None	**Programmer:** Robert Uleski
ROM Required: 177 Bytes	**Company:** Gilford Instrument Labs
Maximum Subroutine Nesting Level: 4	**Address:** 132 Artino, Oberlin, OH. 44074

```
                        ;REF. NO. AB30.
                        ;PROGRAM NAME SYMBOL TALBLE LIST ROUTINE
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
3300                            ORG     3300H

0001                    V20     EQU     1;          TRUE FOR MONITOR VER 32.0
0000                    V22     EQU     0;          TRUE FOR MONITOR VER 2.2

3300 CD9533                     CALL    CRLF
3303 0E16                       MVI     C,22
3305 CDA933                     CALL    BLK0;       PRINT THE NUMBER OF BLANKS
                        ;                           IN REG C
3308 218733                     LXI     H,MSG;      PRINT "SYMBOL TABLE"
330B 0E0E                       MVI     C,14
                        SYM0:
330D 46                         MOV     B,M
330E CDEB3D                     CALL    INCHL
3311 CD573C                     CALL    TTY
3314 0D                         DCR     C
3315 C20D33                     JNZ     SYM0
3318 11D01D                     LXI     D,1DD0H;    POINT TO BEGINING OF SYMBOL
                        LOOP:
331B CD2D33                     CALL    LOOP0;      PRINT 4 SYMBOL PER LINE
331E CD2D33                     CALL    LOOP0
3321 CD2D33                     CALL    LOOP0
3324 CD2D33                     CALL    LOOP0
3327 CD9533                     CALL    CRLF
332A C31B33                     JMP     LOOP
                        LOOP0:
332D 2602                       MVI     H,D;        PRINT A SYMBOL AND ADDRESS SR
332F 7B                         MOV     A,E
3330 C605                       ADI     5
3332 6F                         MOV     L,A
3333 7E                         MOV     A,M
3334 FE02                       CPI     02H;        CHECK FOR USER SYMBOL
3336 C24438                     JNZ     3844H;      RETURN TO MONITOR AT END OF T
3339 6B                         MOV     L,E
333A 0E05                       MVI     C,5
                        LOOP1:
333C 46                         MOV     B,M;        LOAD AND PRINT LABEL
333D CDEB3D                     CALL    INCHL
3340 CD573C                     CALL    TTY
```

```
3343 0D                 DCR     C
3344 C23C33             JNZ     LOOP1
3347 0620               MVI     B,20H;          PRINT TWO SPACES
3349 CDEB3D             CALL    INCHL
334C CD573C             CALL    TTY
334F 0620               MVI     B,20H
3351 CD573C             CALL    TTY
3354 CDEB3D             CALL    INCHL
3357 CD6A33             CALL    BYTE;           PRINT MSB OF ADDRESS
335A CD9F33             CALL    DECHL
335D CD6A33             CALL    BYTE;
3360 CD6A33             CALL    BYTE;           PRINT LSB OF ADDRESS
3363 CD8033             CALL    NEX
3366 CDA733             CALL    BLANK;          PRINT 6 SPACES
3369 C9                 RET

                BYTE:
336A 7E                 MOV     A,M;            PRINT A BYTE AS 2 ASCII CHARS
336B 0F                 RRC
336C 0F                 RRC
336D 0F                 RRC
 36E 0F                 RRC
336F E60F               ANI     0FH
3371 CD7A33             CALL    HXD
3374 7E                 MOV     A,M
3375 E60F               ANI     0FH
3377 C37A33             JMP     HXD

                HXD:
337A CD8C3C             CALL    CONV;           CALLS MONITOR CONVERT SR
337D C3573C             JMP     TTY

                NEX:
3380 7B                 MOV     A,E;            SET D&E TO NEXT SYMBOL
3381 C608               ADI     8
3383 5F                 MOV     E,A
3384 D0                 RNC
3385 14                 INR     D
3386 C9                 RET

3387 53594D42 MSG:      DB      'SYMBOL TABLE',CR,LF
338B 4F4C2054
338F 41424C45
3393 0D0A


                CRLF:
3395 060D               MVI     B,CR;           PRINT CR, LF
3397 CD573C             CALL    TTY
```

```
339A 060A               MVI     B,LF
339C C3573C             JMP     TTY

                DECHL:
339F 2D                 DCR     L;          DECREMENT H & L SR
33A0 2C                 INR     L
33A1 C2A533             JNZ     DECL
33A4 25                 DCR     H
                DECL:
33A5 2D                 DCR     L
33A6 C9                 RET

                BLANK:
33A7 0E06               MVI     C,6;        PRINT 6 BLANKS
                BLK0:
33A9 0620               MVI     B,20H;      PRINT THE NUMBER OF BLANKS IN
33AB CD573C             CALL    TTY
33AE 0D                 DCR     C
33AF C8                 RZ
33B0 C3A933             JMP     BLK0

                        IF      V20
3C8C            CONV    EQU     3C8CH
3C57            TTY     EQU     3C57H
3DEB            INCHL   EQU     3DEBH
                        ENDIF

                        IF      V22
                CONV    EQU     3CC1H
                TTY     EQU     3C5CH
                INCHL   EQU     3DF0H
                        ENDIF
000D            CR      EQU     0DH;        ASCII CARRIAGE RETURN
000A            LF      EQU     0AH;        ASCII LINE FEED

0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

intel®

Ref. No. AA6

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 (use additional sheets if necessary)

Program Title

Diagnostic 1003

Function

To verify that a section of memory is correct by calculating a checksum. The checksum is an ordinary 16-bit sum of the 8-bit bytes contained in the storage region examined.

The routine is explicitly designed to operate entirely in PROM and the CPU without ever using RAM. Thus, the stack cannot be used for anything, including subroutine calls and saving the machine state.

Required Hardware

8080 Chip and ROM Chip.

Required Software

None.

Input Parameters

The starting ROM address, the ending ROM address and the location to which the routine exits.

Output Results

The A register contains zero if the checksum is correct and is non-zero otherwise.

Registers Modified:	Assembler/Compiler Used:
All except SP	8080 Assembler, Version 3.0
RAM Required:	**Programmer:**
None	M. J. Gralia
ROM Required:	**Company:**
43 bytes	Applied Physics Laboratory
Maximum Subroutine Nesting Level:	**Address:**
Zero	Laurel, Maryland 20810

```
                ;REF. NO. AA6
                ;PROGRAM TITLE DIAGNOSTIC 1003
                ;
                ;
                ;
                ;
                ;
                ;       DIAGNOSTIC 1003

                ;PURPOSE:
                ;       TO VERIFY THAT A SECTION OF MEMORY IS
                ;CORRECT BY CALCULATING A CHECKSUM.  THE CHECKSUM
                ;IS AN ORDINARY 16-BIT SUM OF THE EIGHT-BIT
                ;BYTES CONTAINED IN THE STORAGE REGION EXAMINED.
                ;       THE ROUTINE IS EXPLICITLY DESIGNED TO
                ;OPERATE IN PROM AND THE CPU WITHOUT EVER USING
                ;RAM.  THUS, THE SDACK CANNOT BE USED FOR ANYTHING,
                ;INCLUDING SUBROUTINE CALLS AND SAVING THE MACHINE
                ;STATE.

                ;INPUTS:
                ;       THE ROUTINE IS ENTERED VIA A "JUMP"
                ;INSTRUCTION.  THE "CALL" INSTRUCTION MUST NOT BE
                ;USED BECAUSE OF ITS ACCESS OF RAM.
                ;       THE MEMORY REGION TO BE VALIDATED IS
                ;CONTAINED IMPLICITLY WITHIN THE PROGRAM; IT
                ;IS SPECIFIED AT ASSEMBLY TIME.
                ;       A PRE-COMPUTED VALUE FOR THE REGION'S
                ;CHECKSUM IS ALSO CONTAINED WITHIN THE CODE, AND
                ;FIXED AT ASSEMBLY TIME.

                ;OUTPUTS:
                ;       THE ROUTINE TERMINATES VIA A "JUMP"
                ;INSTURCTION.  THE "CALL/RETURN" INSTRUCTION COUPLE
                ;MUST NOT BE USED BECAUSE OF ITS ACCESS TO RAM.
                ;       UPON RECIEPT OF CONTROL, THE "A" REGISTER
                ;IS ZERO IF NO ERRORS WERE DETECTED.
                ;       THE SP REGISTER IS NOT MODIFIED BY THIS
                ;ROUTINE, BUT ALL OTHER REGISTERS AND FLAGS ARE
                ;MODIFIED.  NOTE: THEY CANNOT BE TEMPORARILY
                ;SAVED BECAUSE RAM MUST NOT BE USED.

010B                    ORG 010BH

                ;       DIAGNOSTIC 1003
                ;CHECKSUM PROM.  DON'T USE RAM.
                ;INPUTS ALL IMPLICIT
                ;OUTPUTS: ONLY A=0 IF NO ERRORS, A=0FH IF ERRORS
```

```
                        ;REGS:
                        ;       A  - UTILITY
                        ;       BC - MEMORY POINTER
                        ;       DE - MEMORY CELL CONTENTS
                        ;       HL - CHECKSUM
                        DI1003:
                        ;INITIALIZE
010B 210000                     LXI H,0;        0 -> SUM
010E 010038                     LXI B,START;    FIRST MEM LOC -> MEM PNTR
0111 1600                       MVI D,0
                        DOWHIL:
                        ;DO-WHILE (CURRENT LOCATION) <= (FINISH LOCATION)
0113 3E00                       MVI A,((NOT FINISH) AND 0FFH)
0115 81                         ADD C
0116 3EC0                       MVI A,((NOT FINISH) SHR 8)
0118 88                         ADC B
0119 DA2301                     JC DOEND
                        ;SUM = SUM + MEMORY(BC)
011C 0A                 CONT:   LDAX B
011D 5F                         MOV E,A
011E 19                         DAD D
                        ;BC = BC + 1
011F 03                         INX B
0120 C31301                     JMP DOWHIL
                        DOEND:
                        ;END DO-WHILE
                        ;IF (EXPECT.NE.SUM) THEN A=0FH ELSE A=0
0123 3E65                       MVI A,(EXPECT SHR 8)
0125 BC                         CMP H
0126 C23301                     JNZ THEN
0129 3ED4                       MVI A,(EXPECT AND 0FFH)
012B BD                         CMP L
012C C23301                     JNZ THEN
012F AF                 ELSE:   XRA A
0130 C30000                     JMP XXXX
0133 3E0F               THEN:   MVI A,0FH
0135 C30000                     JMP XXXX
                        ;               @ARAMETERS
65D4                    EXPECT  EQU 065D4H;     EXPECTED CHECKSUM
3FFF                    FINISH  EQU 03FFFH;     LAST MEMORY LOCATION SUMMED
3800                    START   EQU 03800H;     FIRST MEMORY LOCATION SUMMED
0000                    XXXX    EQU 0;  EXIT LOCATION: NEXT ROUTINE
                                                ;AFTER DI1003

0000                            END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AA7

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000

(use additional sheets if necessary)

Program Title

LIST DEVICE TEST PROGRAM

Function

Tests selected list device and its interface.

Required Hardware

Minimum system with list device
CRT is used with enclosed program

Required Software

System monitor

Input Parameters

None

Output Results

Registers Modified: All except H & L	**Assembler/Compiler Used:** 8080 MDS Macro Assembler Version 1.0
RAM Required: 256 x 8	**Programmer:** Leland B. Myers
ROM Required: System monitor	**Company:** North Electric Co. Paul H. Henson Research Center
Maximum Subroutine Nesting Level: None	**Address:** Box 20345 Columbus, Ohio 43220

98-034C

```
                ;REF. NO. AA7
                ;PROGRAM TITLE LIST DEVICE TEST PROGRAM
                ;
                ;
                ;
                ;
                ;
                ; NORTH ELECTRIC CO., DELAWARE, OHIO
                ; PAUL H HENSON RESEARCH CENTER
                ; L. B. MYERS NOVEMBER 21, 1975
                ;
                ; LIST DEVICE TEST PROGRAM
                ;
000A            TAB     EQU     0AH     ; LEFT MARGIN
00F7            INAD    EQU     0F7H    ; CRT
00F6            OUTAD   EQU     0F6H    ; CRT
0001            MASK    EQU     01H     ; CRT
                ;
0030                    ORG     030H
                ;
0030 1E40               MVI     E,040H  ; 64 LINES.
0032 0E1F               MVI     C,01FH  ; " "
0034 060A       LOOP1:  MVI     B,TAB   ; MARGIN.
0036 3E20               MVI     A,020H
0038 CD6100     TABS:   CALL    PRINT
003B 05                 DCR     B
003C C23800             JNZ     TABS
003F 0641               MVI     B,041H  ; 65 CHAR'S.
0041 79                 MOV     A,C
0042 3C         LOOP0:  INR     A
0043 FE60               CPI     060H
0045 C24A00             JNZ     CONT
0048 3E20               MVI     A,020H
004A 4F         CONT:   MOV     C,A
004B CD6100             CALL    PRINT
004E 05                 DCR     B       ; COUNT CHAR'S.
004F C24200             JNZ     LOOP0
0052 3E0D               MVI     A,0DH   ; CR
0054 CD6100             CALL    PRINT
0057 3E0A               MVI     A,0AH   ; LF
0059 CD6100             CALL    PRINT
005C 1D                 DCR     E       ; COUNT LINES.
005D C23400             JNZ     LOOP1
0060 C7                 RST     0       ; FINISHED WHEN E=0.
0061 57         PRINT:  MOV     D,A     ; SAVE CHAR.
 062 DBF7       LOOP:   IN      INAD
0064 E601               ANI     MASK
0066 CA6200             JZ      LOOP    ; LOOP UNTIL READY.
```

```
0069 7A             MOV     A,D
006A D3F6           OUT     OUTAD   ; PRINT CHAR.
006C C9             RET
0000                END
```

intel®

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM Ref. No. AB31

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 (use additional sheets if necessary)

Program Title

PUNCH BINARY TAPE

Function

To transfer the contents of a given section of memory to paper tape in binary form.

The paper tape format is that used by DATA I/O and other PROM programming devices.

Required Hardware

TTY on Port 0 and 1

TTY Printer/Punch on output Port 0
TTY Printer/Punch status on input Port 1

Required Software

The program contains the necessary TTY output subroutine. This subroutine assumes that the data will be inverted between the accumulator output and the TTY serial output connector, as in the case of the Intellec 8/MOD 80 using the imm8-61 INPUT/OUTPUT card.

Input Parameters

Data to be output to paper tape must be stored in consecutive memory locations starting at 100 Hex.

The number of bytes to be punched must be entered in the body of the program at location 0031 Hex.
The default value is 32 decimal or 1F Hex.

Output Results

The execution of this program will cause a paper tape to be punched by the TTY.
After punching six inches of NULL characters as LEADER the program will punch one ASCII "RUBOUT" to indicate that the next tape location contains the first data byte from memory location 100H. The selected number of bytes will then be punched on paper tape. After the data has been punched the program will cause six inches of NULL characters to be punched as TRAILER.

Registers Modified: A,B,C,H & L	**Assembler/Compiler Used:** Intellec 8/MOD 80 Macro Assembler
RAM Required: 512 Bytes	**Programmer:** Robert W. Rea
ROM Required: None	**Company:** Atlantic Research Corp.
Maximum Subroutine Nexting Level: Zero	**Address:** 5390 Cherokee Ave. Alexandria, Va. 22314

```
                                ;REF. NO. AB31
                                ;PROGRAM TITLE PUNCH BINARY TAPE
                                ;
                                ;
                                ;
                                ;
                                ;NAME: PUNCH BINARY TAPE
                                ;
                                ;FUNCTION: PUNCH A BINARY
                                ;TAPE IN FORMAT NEEDED BY
                                ;CERTAIN PROM PROGRAMMERS
                                ;
                                ;DATA TO BE OUTPUT MUST BE
                                ;STORED IN CONSECUTIVE
                                ;LOCATIONS STARTING AT 100H
                                ;
                                ;THE NUMBER OF BYTES TO BE
                                ;PUNCHED MUST BE ENTERED IN
                                ;THE BODY OF PROGRAM AT
                                ;LOCATION 0031
                                ;THE DEFAULT VALUE IS 32
                                ;DECIMAL OR 1F HEX
                                ;
                                ;PROGRAM WILL PUNCH SIX
                                ;INCHS OF NULL CHARACTERS
                                ;AND THEN PUNCH ONE ASCII
                                ;RUBOUT CHARACTER IN THE
                                ;TAPE LOCATION IMMEDIATELY
                                ;PRECEDING THE FIRST EIGHT
                                ;BIT BINARY WORD
                                ;
                                ;AFTER THE SELECTED NUMBER
                                ;OF BYTES HAVE BEEN
                                ;PUNCHED, THE PROGRAM WILL
                                ;PUNCH SIX INCHS OF NULLS
                                ;AS TRAILER AND THEN CAUSE
                                ;THE CPU TO ENTER THE HALT
                                ;STATE
                                ;
0010                            ORG 10H         ;START AT 10 HEX
                                ;
0010 063C                       MVI B,60        ;SET B AS NULL COUNTER
0012 0E00       LEADR:          MVI C,00        ;00 HEX = ASCII NULL
0014 DB01       LOOP1:          IN 01           ;READ TTY STATUS
 916 E604                       ANI 04          ;TEST TTY STATUS
J018 C21400                     JNZ LOOP1       ;LOOP TIL TTY READY
001B 79                         MOV A,C         ;MOVE C TO A
001C 2F                         CMA             ;COMPLEMENT A
```

```
001D D300               OUT 00          ;OUTPUT A TO TTY
001F 05                 DCR B           ;DECREMENT NULL COUNTER
0020 C21200             JNZ LEADR       ;LOOP TO LEADR IF NOT
                                        ;SIXTY NULLS
                                        ;
0023 0EFF               MVI C,0FFH      ;SET C TO ASCII RUBOUT
0025 DB01       LOOP2:  IN 01           ;READ TTY STATUS
0027 E604               ANI 04          ;TEST TTY STATUS
0029 C22500             JNZ LOOP2       ;LOOP TIL TTY READY
002C 79                 MOV A,C         ;MOVE C TO A
002D 2F                 CMA             ;COMPLEMENT A
002E D300               OUT 00          ;OUTPUT A TO TTY
                                        ;
0030 061F       PROM:   MVI B,1FH       ;SET BYTE COUNTER
0032 00         SIZE:   NOP             ;  32 BYTE = 1F HEX
                                        ;  64 BYTE = 3F HEX
                                        ;128 BYTE = 7F HEX
                                        ;256 BYTE = FF HEX
                                        ;
0033 210001             LXI H,100H      ;SET H & L TO 100 HEX
0036 4E                 MOV C,M         ;MOVE MEMORY TO C
 037 DB01       LOOP3:  IN 01           ;READ TTY STATUS
0039 E604               ANI 04          ;TEST TTY STATUS
003B C23700             JNZ LOOP3       ;LOOP TIL TTY READY
003E 79                 MOV A,C         ;MOVE C TO A
003F 2F                 CMA             ;COMPLEMENT A
0040 D300               OUT 00          ;OUTPUT A TO TTY
                                        ;
0042 23         PDATA:  INX H           ;INCREMENT H & L
0043 4E                 MOV C,M         ;MOVE MEMORY TO C
0044 DB01       LOOP4:  IN 01           ;READ TTY STATUS
0046 E604               ANI 04          ;TEST TTY STATUS
0048 C24400             JNZ LOOP4       ;LOOP TIL TTY READY
004B 79                 MOV A,C         ;MOVE C TO A
004C 2F                 CMA             ;COMPLEMENT A
004D D300               OUT 00          ;OUTPUT A TO TTY
004F 05                 DCR B           ;DECREMENT BYTE COUNTER
0050 C24200             JNZ PDATA       ;GET ANOTHER CHARACTER
                                        ;IF BYTE COUNTER NOT ZERO
                                        ;
0053 063C               MVI B,60        ;SET B AS NULL COUNTER
0055 0E00       TRALR:  MVI C,00        ;00 HEX = ASCII NULL
0057 DB01       LOOP5:  IN 01           ;READ TTY STATUS
0059 E604               ANI 04          ;TEST TTY STATUS
005B C25700             JNZ LOOP5       ;LOOP TIL TTY READY
005E 79                 MOV A,C         ;MOVE C TO A
 35F 2F                 CMA             ;COMPLEMENT A
 060 D300               OUT 00          ;OUTPUT A TO TTY
0062 05                 DCR B           ;DECREMENT NULL COUNTER
0063 C25500             JNZ TRALR       ;LOOP TO TRALR IF NOT
```

```
                                    ;SIXTY NULLS
                                    ;
0066 76                 HLT         ;PUT CPU IN HALT STATE
0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB32

☐ 4004 ☐ 8008 ☒ 8080 ☐ 4040

(use additional sheets if necessary)

Program Title

MEMORY COMPARE

Function

This program extends the version 3.0 monitors functions to include a memory to memory compare. The program as written occupies locations 3000 to 30FF but may be reassembled to occupy any 256 bytes of memory.

Required Hardware

MCS Intellec 8/MOD 80

Required Software

Intellec 8/MOD 80 Monitor, Version 3.0, 14 April 1975.

Input Parameters

When started, the program types "C" to show it is active. There are two legal inputs that may be entered at this time, "C" or "H". "C" means the user wants to compare 256 bytes of memory (The C stands for Chip). "H" means the user wants to compare 4096 bytes of memory (The H stands for WHole). The program then waits for two arguments to be entered. The arguments are the address at which the comparisons are to begin. If a mistake is made, type Rubout and the program will return control to the monitor. The arguments may be separated by a space or a comma.

Output Results

The output results are similar to the results of the compare memory to PROM function as supplied by the monitor. If no differences are found, nothing is printed. If differences are found, the first address followed by its contents followed by the second address followed by its contents are printed. This printout occurs for all locations which do not match.

Registers Modified: *ALL (A, B, C, D, E, H, L)*	**Maximum Subroutine Nesting Level:** *Intellec 8/MOD 80 VER 3.0*
RAM Required: *STACK USE ONLY*	**Assembler/Compiler Used:** *R. E. Considine*
ROM Required: *MAY BE PLACED IN 256 BYTE ROM*	**Programmer:** *SRL MEDICAL, INC.*
2 levels	**Company:** *2676 Indian Ripple Rd. Dayton, Ohio 45440*

```
                        ;REF. NO. AB32
                        ;PROGRAM TITLE MEMORY COMPARE
                        ;
                        ;
                        TITLE   'MEMORY COMPARE ROUTINE'

                                ;MEMORY COMPARE ROUTINE
                                ;FOR 8080
                                ;10-NOV-75

3CCD                            CRLF    EQU     3CCDH
3C55                            CO      EQU     3C55H
3F73                            TI      EQU     3F73H
3C53                            BLK     EQU     3C53H
3DBF                            LADR    EQU     3DBFH
3DC7                            LBYTE   EQU     3DC7H
3D70                            EXPR    EQU     3D70H

3000                            ORG     3000H

3000 CDCD3C     COMPR:          CALL    CRLF
3003 0E43                       MVI     C,'C'
3005 CD553C                     CALL    CO
3008 CD733F                     CALL    TI
300B FE43                       CPI     'C'
300D CA1D30                     JZ      PART
3010 FE48                       CPI     'H'
3012 CA2230                     JZ      WHOLE
3015 0E3F                       MVI     C,'?'
3017 CD553C                     CALL    CO
301A C30030                     JMP     COMPR

301D 0E01       PART:           MVI     C,1
301F C32430                     JMP     HOLE1

3022 0E10       WHOLE:          MVI     C,16
3024 C5         HOLE1:          PUSH    B
3025 CD3430                     CALL    GETPR
3028 CD3C30     HOLE2:          CALL    CM256
302B C1                         POP     B
302C 0D                         DCR     C
302D C5                         PUSH    B
302E C22830                     JNZ     HOLE2
3031 C30030                     JMP     COMPR

3034 0E02       GETPR:          MVI     C,2
3036 CD703D                     CALL    EXPR
3039 D1                         POP     D
303A E1                         POP     H
```

```
303B C9                 RET

303C 0E00       CM256:  MVI     C,0
303E C5         CMLUP:  PUSH    B
303F 7E                 MOV     A,M
3040 EB                 XCHG
3041 46                 MOV     B,M
3042 EB                 XCHG
3043 B8                 CMP     B
3044 CA6330             JZ      NDIFF
3047 CDCD3C     DIFF:   CALL    CRLF
304A CDBF3D             CALL    LADR
304D CD533C             CALL    BLK
3050 7E                 MOV     A,M
3051 CDC73D             CALL    LBYTE
3054 CD533C             CALL    BLK
3057 EB                 XCHG
3058 CDBF3D             CALL    LADR
305B CD533C             CALL    BLK
305E 7E                 MOV     A,M
305F CDC73D             CALL    LBYTE
3062 EB                 XCHG
3063 23         NDIFF:  INX     H
3064 EB                 XCHG
3065 23                 INX     H
3066 EB                 XCHG
3067 C1                 POP     B
3068 0C                 INR     C
3069 C23E30             JNZ     CMLUP
306C C9                 RET

0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AA8

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 (use additional sheets if necessary)

Program Title

TRACE

Function

Provide a program debug trace facility by dumping the address of the calling location and the contents of the registers.

Required Hardware

TTY on ports 0 and 1.

Required Software

The routine calls four TTY utilities in the monitor. However, they are very short and could be included when the monitor is not available.

Input Parameters

All of the registers and the first location of the stack(return address).

Output Results

A header description line and the dump below. For example, if trace were called from location 100H the console output would be:

```
*T*  A F  B C  D E  H L  M   SP
0103 0A02 1B23 0001 5678 0101 0088
```

Note that the return address is three greater than the calling location and that the M reference is considered 2 bytes.

Registers Modified:	Assembler/Compiler Used:
None	Intellec 8 macro assembler
RAM Required:	**Programmer:**
66 bytes	Lorne Douglas
ROM Required:	**Company:**
None	Telaid Systems
Maximum Subroutine Nesting Level:	**Address:**
None	6725 Variel,Canoga Park,Cal,91303

```
                        ;REF.  NO.  AA8
                        ;PROGRAM TITLE TRACE
                        ;
                        ;
                        ;
                        ;         **************
                        ;         * * TRACE * *
                        ;         **************
                        ;
                        ; THIS ROUTINE DUMPS THE ADDRESS OF
                        ; THE CALLING LOCATION & THE VALUES
                        ; OF THE REGISTERS.  IT IS INTENDED
                        ; A DEBUGGING AID POSSIBLY STORED
                        ; IN PROM.
                        ;
                        ;
                        ; UPON ENTRY ALL OF THE VALUES TO BE
                        ; DUMPED ARE PUSHED INTO THE STACK.
                        ; THE HEADER IS THEN PRINTED AND THE
                        ; ACTUAL VALUES ARE TAKEN FROM THE
                        ; STACK.  ALL REGISTERS ARE THEN
                        ; RESTORED AND CONTROL IS RETURNED.
                        ;
                        ; THIS VERSION WAS INTENDED FOR USE
                        ; WITH AN INTELLEC 8/MOD.80 WITH
                        ; MONITOR VERSION 3.0.  THE FORMAT OF
                        ; THE DISPLAY WAS DESIGNED FOR USE
                        ; WITH A CRT WHICH HAS A PRINT LINE
                        ; OF 40 CHR.
                        ;
                        ; THE FOLLOWING ARE MONITOR UTILITIES
                        ; THAT WRITE TO THE CONSOLE,  IF USING
                        ; A DIFFERENT MONITOR CHANGE TO
                        ; APPROPRIATE ADDRESSES.
                        ;;
3C53                    BLK     EQU  3C53H    ; WRITE BLANK
3CCD                    CRLF    EQU  3CCDH    ; END OF LINE
3C55                    CO      EQU  3C55H    ; WRITE 1 CHR
3DC7                    LBYTE   EQU  3DC7H    ; WRITE HEX BYTE
                        ;
07E0                            ORG  07E0H
07E0                    HEAD    EQU  $        ; HEADER
07E0 2A542A20                   DB   '*T*  A F  B C  D'
07E4 20412046
 7E8 20204220
-7EC 43202044
07F0 20452020                   DB   ' E  H L  M   SP'
07F4 48204C20
```

```
07F8 204D2020
07FC 20205350
0020            TITCH   EQU   $-HEAD
                ;
0800            TRACE   EQU   $         ; * ENTRY
                ;
0800 F5                 PUSH  PSW       ; THE DATA TO BE
0801 C5                 PUSH  B         ; DISPLAYED IS ALL
0802 D5                 PUSH  D         ; PUSHED INTO THE
0803 E5                 PUSH  H         ; STACK IN THE
0804 46                 MOV   B,M       ; ORDER OF THE DUMP
0805 23                 INX   H         ; M IS CONSIDERED A
0806 4E                 MOV   C,M       ; 16 BIT LOCATION
0807 C5                 PUSH  B
0808 210C00             LXI   H,00CH    ; STACK VALUE PRIOR
080B 39                 DAD   SP        ; TO TRACE CALL
080C E5                 PUSH  H
080D 2B                 DCX   H         ; CALLING ADDR
080E E5                 PUSH  H         ; SAVE FOR DUMP
                ;
                ;
                ; NOW DUMP THE TRACE TITLE LINE
                ;
080F 0620               MVI   B,TITCH ; NUMBER OF CHR
0811 21E007             LXI   H,HEAD  ; CHR STRING
                LOOP:
0814 4E                 MOV   C,M       ; PICK UP CHR
0815 CD553C             CALL  CO        ; PRINT IT
0818 05                 DCR   B         ; SEE IF DONE
0819 CA2008             JZ    DONE
081C 23                 INX   H         ; ADDR OF NEXT CHR
081D C31408             JMP   LOOP
                DONE:
                ;
                ;
0820 CDCD3C             CALL  CRLF      ; NEXT LINE

                ; NOW THAT THE TITLE IS PRINTED ITS
                ; TIME TO DUMP THE DATA IN THE STACK
0823 E1                 POP   H         ; RESTORE POINTER
                ;
0824 0607               MVI   B,7       ; # BYTE PAIRS
0826 C5         LOOP2:  PUSH  B         ; SAVE COUNT
0827 7E                 MOV   A,M       ; PICK UP FIRST 1
0828 CDC73D             CALL  LBYTE     ; & DUMP IT
082B 2B                 DCX   H         ; NEXT 1
 92C 7E                 MOV   A,M
082D CDC73D             CALL  LBYTE
0830 2B                 DCX   H         ; 4 NEXT TIME
0831 CD533C             CALL  BLK       ; PRINT BLANK
```

```
0834 C1              POP  B        ; LOOP COUNT
0835 05              DCR  B        ; TEST IF DONE
0836 C22608          JNZ  LOOP2
0839 CDCD3C          CALL CRLF     ; NEW LINE
                     ;
                     ; ALL DONE NOW RESTPRE REGISTERS
                     ; AND EXIT
                     ;
083C E1              POP  H        ; THESE 2 SYNC TO
083D E1              POP  H        ; THE TRUE H
                     ;
083E E1              POP  H        ; RESTORE'M
083F D1              POP  D        ;
0840 C1              POP  B
0841 F1              POP  PSW      ;
0842 C9              RET           ; ** ADIOS **
                     ;
                     ;
                     ;
                     ;* * * THAT'S ALL FOLKS * * *
                     ;
                     ;
                     ; TEST TEST TEST TEST TEST TEST TEST
                     ; **** **** **** **** **** **** ****
                     ;
0200                 ORG  200H
0200 3E0A            MVI  A,0AH
0202 012C1B          LXI  B,1B2CH
0205 114E3D          LXI  D,3D4EH
0208 217856          LXI  H,5678H
020B 310001          LXI  SP,0100H
020E CD0008          CALL TRACE
0211 CD0008          CALL TRACE

0214 C30000          JMP  0
                     ;
                     ;* * * * * * * * * * * * * * * * * * * *
                     ;
0000                 END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AC10

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000

(use additional sheets if necessary)

Program Title

Handler for Tally paper tape punch

Function

The program TALLY is a handler that interfaces to the 8080 monitor. The Tally punch requires a 4.5 msec drive pulse and has a maximum punch rate of 60 cps. To obtain the long drive pulse the step line of the Tally was wired to the drive direction line of the MDS. The Tally punch has high true logic which requires software inversion of the signals. On entry to the subroutine the C register contains the data to be punched. The data is output to device F8H. The drive bit (bit 4) on device F9H is set and a 4.5 msec time delay is generated. The drive bit is removed and a 12.2 msec delay is genereated. The data lines are cleared and a return is made. The PSW, and B register pairs are saved on entry and restored before return.

Required Hardware

Tally model PD-420 paper tape punch and MDS-800

Required Software

MDS-800 monitor

Input Parameters

C register contains data to be punched

Output Results

Punch on paper tape

Registers Modified:	Assembler/Compiler Used:
None	Intellec MDS Assembler Ver 1.0
RAM Required:	**Programmer:**
2DH bytes	S. Graf
ROM Required:	**Company:**
None	Sentrol Systems Ltd.
Maximum Subroutine Nesting Level:	**Address:** 4401 Steeles Avenue West Toronto, Ontario M3N 2S4
None	CANADA

```
                ;REF. NO. AC10
                ;PROGRAM TITLE HANDLER FOR TALLY PTP
                ;
                ;
                ;
                ;         TALLY PUNCH HANDLER
                ;
                ;
                TALLY:
0000 F5                 PUSH    PSW     ;SAVE
0001 C5                 PUSH    B
0002 79                 MOV     A,C
0003 2F                 CMA             ;POSITIVE TRUE
0004 D3F8               OUT     0F8H    ;OUTPUT DATA
0006 AF                 XRA     A       ;CLEAR A
0007 D3F9               OUT     0F9H    ;SET DRIVE BIT
0009 0602               MVI     B,2
000B 3EFF       UP1:    MVI     A,0FFH
000D 3D         UP2:    DCR     A       ;DELAY 4.5 MS
000E C20D00             JNZ     UP2
0011 05                 DCR     B
0012 C20B00             JNZ     UP1
0015 3E10               MVI     A,10H   ;REMOVE DRIVE BIT
0017 D3F9               OUT     0F9H
0019 0607               MVI     B,7
001B 3EFF       UP3:    MVI     A,0FFH  ;DELAY (16.7-4.5) MS
001D 3D         UP4:    DCR     A
001E C21D00             JNZ     UP4
0021 05                 DCR     B
0022 C21B00             JNZ     UP3
0025 3EFF               MVI     A,0FFH
0027 D3F8               OUT     0F8H    ;CLEAR DATA LINES
0029 C1                 POP     B       ;RESTORE
002A F1                 POP     PSW
002B C9                 RET
                ;
                ;
0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB33

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000

(use additional sheets if necessary)

Program Title

CRCC-16

Function

This program generates a 2-byte cyclic redundancy check character, obtained as remainder of the division of the serial data stream by a selected polynomial. This character is stored in register pair B (most significant byte) and C.

Required Hardware

8251 as input device.

Required Software

None

Input Parameters

The selected polynomial is given in the form of two constants stored in instructions 410 and 480. The selected polynomial for this case is

$$x^{16} + x^{15} + x^2 + 1$$

coming as constants 01000000B, and 00000001B, respectively. Any polynomial of type $x^{16} + \sum_{i=15}^{1} a_i x^i + 1$ (the a's being boolean), has to be introduced as the sequences of bits $a_1, a_2, \ldots a_7$, Ø and $a_8, a_9, \ldots, a_{15}$ as immediate fields of instructions 410 and 480 respectively.

Output Results

Registers B and C store the check character. After the read-out of one block of data followed by this character, the content of B & C has to be Ø.

Registers Modified: A,B,C,D,E	**Assembler/Compiler Used:** 8080 Macroassembler
RAM Required: Ø	**Programmer:** M. Nobile
ROM Required: 4Ø bytes	**Company:** Honeywell Information Systems Italia
Maximum Subroutine Nesting Level:	**Address:** I-20010 Pregnana Milanese, Italy

```
                        ;REF. NO. AB33
                        ;PROGRAM TITLE CRCC-16
                        ;
                        ;
                        ;
                        ;  THIS PROGRAM GEREATES A TWO BYTE CYCLIC
                        ;  REDUNDANCY CHECK CHARACTER, OBTAINED AS REMAINDER
                        ;  OF THE DIVISION OF THE SERIAL DATA STREAM BY A
                        ;  SELECTED POLYNOMIAL
                        ;  THE DATA STREAM ISSREAD BYTE AFTER BYTE THROUGH
                        ;  INPUT DEVICE #0 ASSUMED TO BE AN 8251.
                        ;  THE CRCC IS STORED IN REGISTER PAIR B (MORE
                        ;  SIGNIFICANT BYTE) AND C (L.S. BYTE).
                        ;  THE SELECTED POLYNOMIAL IS GIVEN IN THE FORM OF
                        ;  IMMEDIATE GONSTANTS STORED IN INSTRUCTIONS
                        ;  410 AND 480
                        ;  THE PRESENT PROGRAM SELECTS POLYNOMIAL
                        ;       X**16 + X**15 + X**2 + 1
                        ;  COMING AS CONSTANTS 40H AND 01H.
                        ;  ANY POLYNOMIAL OF TYPE
                        ;       X**16 + A(15)*X**15 + .... + A(1)*X +1
                        ;  THE A'S BEING BOOLEAN, HAS TO BE INTRODUCED AS TWO
                        ;  CONSTANTS, NAMELY THE SEQUENCE OF BINARY DIGITS
                        ;  A(1),A(2), ..... ,A(7),0   AND THE SEQUENCE
                        ;  OF BINARY DIGITS A(8),A(9), ... ,A(15) AS
                        ;  IMMEDIATE FIELD OF INSTRUCTIONS
                        ;  410 AND 480 RESPECTIVELY.
                        ;
                        ;
                        ;
                        ;
0100                            ORG     100H
0100 010000                     LXI     B,0H        ;INITIALIZATION - CLEAR B AND C
0103 DB00               W1:     IN      0H          ;INPUT DATA
0105 1E08                       MVI     E,8H        ;SET BYTE COUNTER
0107 57                 BITF:   MOV     D,A
0108 A9                         XRA     C           ;CHECK BIT CALCULUS
0109 1F                         RAR
010A 78                         MOV     A,B
010B D22101                     JNC     BITO        ;CONDITIONAL JUMP ON CHECK BIT
010E EE40                       XRI     40H         ;UPDATING DUE TO THE CHOSEN POLYNOM
                                                    ;IN MS BYTE CRC REGISTER (B)
0110 37                         STC
0111 1F                         RAR
0112 47                         MOV     B,A
 113 79                         MOV     A,C
 114 1F                         RAR
0115 EE01                       XRI     01H         ;UPDATING DUE TO THE CHOSEN POLYNOM
                                                    ;IN LS BYTE CRC REGISTER (C)
```

```
0117 4F        BITE:   MOV     C,A
0118 7A                MOV     A,D
0119 1F                RAR
011A 1D                DCR     E
011B C20701            JNZ     BITF    ;TO NEW DATA BIT
011E C30301            JMP     W1      ;TO NEW DATA BYTE
0121 1F        BITO:   RAR             ;SUBROUTINE ENTRY FOR CHECK BIT = 0
0122 47                MOV     B,A
0123 79                MOV     A,C
0124 1F                RAR
0125 C31701            JMP     BITE
0000                   END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AC11

☐ 4004 ☐ 4040 ☐ 8008 XX 8080 ☐ 3000

(use additional sheets if necessary)

Program Title

Intellec MDS to TI Silent 700 Interface

Function

To interface a Ti Silent 700 Terminal with magnetic tape cassettes to an Intellec MDS and permit both the printer and cassettes to run at 30 characters per second speed. Both hardware and software modifications are required to the MDS and hardware changes are required for the TI Silent 700.

Required Hardware

Intellec MDS
One (1) MDS-406 6K PROM board
Nine (9) 1702A PROMs
TI Silent 700 ASR Terminal with current loop interface
UPP PROM Programmer or equivalent capability

Required Software

MDS Monitor Version 1 or Version 2
MDS Boot PROM V1.0 or V1.1

Input Parameters

Revised 8/8/77

Output Results

Program Listing Available Only.
Paper Tape is not offered.

Registers Modified:	**Assembler/Compiler Used:** ---
RAM Required:	**Programmer:** Richard Lonchar
ROM Required:	**Company:** North Electric Co.
Maximum Subroutine Nesting Level:	**Address:** P.O. Box 20345 Columbus, Ohio 43220

98-034C

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB34

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 (use additional sheets if necessary)

Program Title

COMPARE

Function

This program reads a HEX format tape and compares it to the corresponding memory locations. When the two don't match, the address is printed on the system console.
Included are three options:

1. Print contents of tape & addr.
2. Skip if address was originally Ø.
3. Restore memory to original value.

Required Hardware

8080

Required Software

Several utility routines available in the MCS 80 monitor were called. These could be substituted or rewritten for use on other systems (see listing).

Input Parameters

HEX format tape in reader.

Output Results

The program lists all locations that have been modified and, optionally, the original and modified values. Also, optionally, the program may restore the modified locations to the original value.

Registers Modified: A,B,C,H,L	**Assembler/Compiler Used:** Intellec 8 Macro Assembler
RAM Required: 16 BYTES	**Programmer:** LORNE DOUGLAS
ROM Required: 113 BYTES (can be in RAM)	**Company:** TELAID SYSTEMS, INC.
Maximum Subroutine Nesting Level: 16 BYTES IN STACK	**Address:** 6725 Variel Avenue Canoga Park, Calif. 91303

98-034C

```
                ;REF. NO. AB34
                ;PROGRAM TITLE COMPARE
                ;
                ;
                ; ****************************************
                ; *            COMPARE                   *
                ; ****************************************
                ;
                ; THIS PROGRAM READS A HEX FORMAT TAPE
                ; AND COMPARES IT TO THE CORRESPONDING
                ; MEMORY LOCATIONS. WHEN THE TWO DON'T
                ; MATCH THE ADDRESS IS PRINTED ON  THE
                ; SYSTEM CONSOLE.
                ;
                ;
0000            NO      EQU   0
0001            YES     EQU   1
                ;
                ;
                ; INCLUDED ARE THREE OPTIONS:
                ;
                ;       1 PRINT CONTENTS OF TAPE & ADDR.
                ;       2 SKIP IF ADDR WAS ORIGINALLY 0.
                ;       3 RESTORE MEM TO ORIGINAL VALUE.
                ;
                ;
0001            PRNTM   SET   YES
0001            SKIPM   SET   YES
0001            RSTRM   SET   YES
                ;
                ;
                ;
                ; THE BULK OF THIS PROGRAM IS AN EXACT
                ; COPY OF THE MONITOR READ ROUTINE.  IT
                ; WAS ASSEMBLED FOR USE ON AN INTELLEC
                ; 8 MOD 80 RUNNING ON VERSION 4.  THERE
                ; ARE SEVERAL MONITOR UTILITIES CALLED
                ; THAT COULD BE SUBSTITUTED FOR USE ON
                ; OTHER SYSTEMS.
                ;
3D70            EXPR    EQU   3D70H  ;GET OFFSET ADDR
3EFF            RIX     EQU   3EFFH  ;GET CHR FROM READR
3C82            BYTE    EQU   3C82H  ;READ TWO ASCII CHR
                                     ; DECODE TO BINARY
 C55            CO      EQU   3C55H  ;PRINT CHR IN C
3C53            BLK     EQU   3C53H  ;PRINT BLANK
3CCD            EOL     EQU   3CCDH  ;PRINT CR LF
3DBF            LADR    EQU   3DBFH  ;PRINT ADDR IN H&L
```

```
3DC7            LBYTE   EQU     3DC7H   ;CONVERT 1 BINARY
                                        ;  BYTE TO HEX & PRT
386B            START   EQU     386BH   ;MONITOR RESTART
3C43            LER     EQU     3C43H   ;ERROR RETURN
                ;
                ;
                ;
0300                    ORG     300H
                ;
                READ:
0300 0D                 DCR     C       ; GET ONE ADDR
0301 CD703D             CALL    EXPR
                REDO:
0304 E1                 POP     H       ; GET BIAS ADDR
0305 E5                 PUSH    H
0306 CDFF3E             CALL    RIX
0309 063A               MVI     B,':'
030B 90                 SUB     B
030C C20403             JNZ     REDO    ; SCAN TO REC MRK
030F 57                 MOV     D,A     ; CLEAR CHECKSUM
 310 CD823C             CALL    BYTE
0313 CA6003             JZ      RED2    ; ZERO RECORD L
                                        ; LENGTH ALL DONE
0316 5F                 MOV     E,A     ; E<-RECORD LENGTH
0317 CD823C             CALL    BYTE    ; MSB OF LOAD ADR
031A F5                 PUSH    PSW     ; SAVE IT
031B CD823C             CALL    BYTE    ; LSB OF LOAD ADR
031E C1                 POP     B       ; RETV MSB PUT IN B
031F 4F                 MOV     C,A
0320 09                 DAD     B       ; BIAS + LOAD -> HL
0321 CD823C             CALL BYTE       ; RECORD TYPE
                RED1:
0324 CD823C             CALL    BYTE
                ;
                ;
                ;** COMPARE INSERT **********************
                ;
                ;
                ;
                IF      SKIPM           ;
0327 B7                 ORA     A       ;TEST TO SEE IF 0
0328 CA5203             JZ      ENDIT   ; AND EXIT
                        ENDIF
                ;
032B F5                 PUSH    PSW     ;SAVE BYTE FROM TAP
 32C 96                 SUB     M       ;*TEST* IF MATCH
032D CA5003             JZ      POPIT   ; IF SO ADIOS
                ;
                ;THIS CODE IS EXECUTED IF THE LOCATION
```

```
                        ;HAS CHANGED
                        ;
0330 CDCD3C                     CALL    EOL     ;START A NEW LINE
0333 CDBF3D                     CALL    LADR    ;PRINT LOCATION
                        IF      PRNTM
0336 CD533C                     CALL    BLK     ;PRINT A BLANK
0339 0E54                       MVI     C,'T'   ;PRINT A 'T' TO
033B CD553C                     CALL    CO      ;  IDENTIFY TAPE
033E F1                         POP     PSW     ;GET BACK THE CHR
033F F5                         PUSH    PSW     ;  FROM THE TAPE
0340 4F                         MOV     C,A     ;  AND PRINT IT
0341 CDC73D                     CALL    LBYTE   ;
0344 CD533C                     CALL    BLK     ;PRINT A BLANK
0347 0E4D                       MVI     C,'M'   ;PRINT 'M' FOR
0349 CD553C                     CALL    CO      ;  MEMORY
034C 7E                         MOV     A,M     ;PRINT THE CONTENTS
034D CDC73D                     CALL    LBYTE   ;  OF MEMORY
                                ENDIF
                        ;
0350 F1                 POPIT:  POP     PSW     ;CHR FROM TAPE
                        IF      RSTRM
 351 77                         MOV     M,A     ;RESTORE VALUE
                                ENDIF
                        ;
                        ENDIT:
                        ;
                        ;
                        ;** END INSERT ************************
                        ;
                        ;
0352 23                         INX     H
0353 1D                         DCR     E
0354 C22403                     JNZ     RED1    ;  LOOP UNTIL DONE
0357 CD823C                     CALL    BYTE    ;  READ CHECKSUM
035A C2433C                     JNZ     LER     ;  CHECKSUM ERROR
035D C30403                     JMP     REDO    ;  ANOTHER RECORD

                        RED2:

0360 CD823C                     CALL    BYTE    ;  GET MSB TRANSFER
                                                ;  ADDRESS

0363 67                         MOV     H,A
0364 CD823C                     CALL    BYTE
0367 6F                         MOV     L,A
0368 B4                         ORA     H
 369 CA6D03                     JZ       RED3
036C E9                         PCHL
                        RED3:
036D E1                         POP     H
```

```
036E C36B38             JMP    START
                ;
                ;
0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB36

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

INPUT and OUTPUT COMMANDS FOR INTELLEC/MDS

Function

These two pseudo monitor commands allow the user to gain console control over "IN" and "OUT" ports in the MDS system. Data can be read into the console using the "I" command; data can be sent out using the "O" command.

Required Hardware

Standard MDS system with console.

Required Software

INTELLEC/MDS Monitor Version 1.0, resident on Prom.

Input Parameters

Letter I or O from console input and data:

<I><PORT NUMBER> (CONSOLE RETURNS INPUT PORT DATA)

<O><PORT NUMBER>, <DATA TO BE OUTPUTTED>

Output Results

The I command prints on the console the value of the chosen input port.

The O command causes the chosen data value to be sent to the output port.

Registers Modified: -	Assembler/Compiler Used: Intellec 80 MACRO Assembler
RAM Required: 3 BYTES	Programmer: Paul G. St. Amand
ROM Required: 41 (decimal) BYTES	Company: Bell Telephone Labs
Maximum Subroutine Nesting Level: -	Address: Room 3E-25 1600 Osgood Street No. Andover, MA. 01845

```
                ;REF. NO. AB36
                ;PROGRAM TITLE INPUT & OUTPUT COMMANDS FOR MDS
                ;
                ;
                ;
                ;       PAUL G. ST. AMAND       BELL TELEPHONE LABS  24 JAN 76
                ;
                ;       ADDITIONAL MONITOR COMMANDS FOR INTELLEC/MDS SYSTEM
                ;
                ;
                ;  THIS ROUTINE ALLOWS THE USER TO GAIN ACCESS TO THE INPUT AND
                ;  PORTS OF HIS MDS SYSTEM, THROUGH USE OF TWO MONITOR-LIKE COMM
                ;  THE USER=S EXISTING MONITOR SOFTWARE MUST BE MODIFIED TO ALLO
                ;  THE TWO NEW COMMANDS OF <IN> AND <OUT> TO BE RECOGNIZED.
                ;  THE MONITOR EPROM AT LOCATION 0F800H MUST BE CHANGED AS SHOWN
                ;
                ;                   LOCATION        NEW DATA
                ; INPUT -I-           F86F            00
                ;                     F870            F7
                ;
                ; OUTPUT -0-          F87B            15
                ;                     F87C            F7
                ;
                ;  THIS WILL CAUSE THE MONITOR TO BRANCH TO THE NEW PROM CODE AT
                ;  APPROPRIATE LOCATIONS.
                ;
                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                ;
                ;                   SYMBOL TABLE FOR LINK POINTS TO MDS MONITOR
                ;
FE7F            EXPR    EQU     0FE7FH  ;BRING IN DATA FROM CONSOLE
FEAA            LBYTE   EQU     0FEAAH  ;DATA OUTPUT ROUTINE TO CONSOLE
FECB            PARAM   EQU     0FECBH  ;BRINGS IN ONE BYTE OF DATA FROM CONSOLE
                ;
                ;                   SYMBOL TABLE FOR PROGRAM LOCATIONS
                ;
0010            PATCH   EQU     10H     ;USER RAM, TO BE USED BY ROUTINES.
F700            ROM     EQU     0F700H  ;FIRST PROM BELOW MONITOR. (ARBITRARY)
                ;
                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
F700                    ORG     ROM
                ;
                ;
                ;
                ;                   INPUT ROUTINE
                ;
                ;  THE USER WILL TYPE IN <I><NUMBER>< >.  THE SYSTEM WILL THEN
                ;  ANSWER WITH <-ANSWER>.  WHERE <NUMBER> IS THE INPUT PORT NUMBE
                ;  IN HEX, AND <-ANSWER> IS THE HEX VALUE OF THE SELECTED INPUT
```

```
                 ;  PORT. FOR EXAMPLE, IF INPUT PORT 08 CONTAINS THE VALUE 70 THE
                 ;  ROUTINE WILL RETURN:
                 ;                           .I08 -79
                 ;
                 INPORT:
F700 CDCBFE              CALL    PARAM    ;FETCH THE PORT NUMBER

F703 4D                  MOV     C,L      ;SAVE PORT NUMBER IN C REGISTER
F704 211000              LXI     H,PATCH  ;LOAD RAM PATCH LOCATION
F707 36D8                MVI     M,0D8H   ;LOAD <IN> COMMAND
F709 23                  INX     H
F70A 71                  MOV     M,C      ;LOAD PORT NUMBER
F70B 23                  INX     H
F70C 36C9                MVI     M,0C9H   ;LOAD <RETURN> COMMAND
                 ;
                 ;  RAM WILL NOW HAVE THE PATCH CODE:
                 ;           IN
                 ;           PORT NUMBER
                 ;           RETURN
                 ;
F70E CD1000              CALL    PATCH    ;EXECUTE THE CODE
 711 CDAAFE              CALL    LBYTE    ;DISPLAY THE DATA
F714 C9                  RET              ;RETURN TO MONITOR COMMAND LOOP
                 ;
                 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                 ;
                 ;
                 ;
                 ;                 OUTPUT PORT ROUTINE
                 ;
                 ;  THE USER TYPES IN <O><NUMBER>,<DATA>. WHEN THE CARRIAGE
                 ;  RETURN IS ACTIVATED, THE MONITOR WILL OUTPUT TO PORT NUMBER
                 ;  <NUMBER> THE VALUE <DATA>, WHERE BOTH <NUMBER> AND <DATA> ARE
                 ;  IN HEX FORMAT.
                 ;  AN EXAMPLE WOULD BE:
                 ;                           .O3C,42
                 ;  THIS WOULD CAUSE HEX 42 TO BE SENT TO PORT NUMBER 3C.
                 ;
                 OUTPORT:
F715 CD7FFE              CALL    EXPR     ;FETCH PORT NUMBER AND DATA
F718 211000              LXI     H,PATCH  ;LOAD ADDRESS FOR RAM PATCH
F71B 36D3                MVI     M,0D3H   ;LOAD <OUT> COMMAND
F71D 23                  INX     H
F71E C1                  POP     B        ;DATA VALUE TO OUTPUT IS IN C REGISTER
F71F D1                  POP     D        ;PORT NUMBER WILL BE IN E REGISTER
F720 73                  MOV     M,E      ;LOAD PORT NUMBER
F721 79                  MOV     A,C      ;MOVE DATA VALUE INTO REGISTER A
  22 23                  INX     H
F723 36C9                MVI     M,0C9H   ;LOAD <RETURN> INTO RAM
                 ;
```

```
                ;   THE RAM PATCH AREA WILL NOW HOLD:
                ;               OUT
                ;               PORT
                ;               RET
                ;
F725 CD1000             CALL    PATCH   ;EXECUTE THE CODE
F728 C9                 RET             ;RETURN TO MONITOR COMMAND LOOP
                ;
                ;                       END
0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB37

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

MLOAD

Function

To allow MDS-DOS users to load a binary file (created by the HEXBIN command) using ISIS, at an address modifed with a bias. MDS-DOS users cannot currently create, assemble, load, and transfer to PROM a program which uses memory addresses less than 12,000. The reason for this problem is that such programs, when loaded into MDS using the DEBUG command, attempt to overlay the ISIS control program. An error will result, aborting the attempted load. By using MLOAD, such programs may be loaded at an address above 12,000, and then transferred to PROM.

Required Hardware

Standard MDS-DOS configuration, with either 1 or 2 disk drives, 32K + memory, and console.

Required Software

ISIS control program.

Input Parameters

MLOAD is initiated by an ISIS-like command, as is EDIT,ASM80,etc.

While system is under ISIS control, and is expecting an ISIS-like command, enter:

MLOAD ⟨filename⟩$=⟨address⟩H

where:

⟨filename⟩ is a binary file, created by HEXBIN.
⟨address⟩ is the biased address of the load in HEX.

Output Results

The binary file, specified in ⟨filename⟩ is loaded into MDS RAM, beginning at the address biased by ⟨address⟩ .If no error occurrs, control is returned to the MDS Monitor to allow PROM programming. Otherwise, an error message will be output on the system console, and control is returned to ISIS. Error codes are standard ISIS error codes, with the exception of the MLOAD syntax error, which is error code 100.

-SEE ATTATCHED NOTE-

Registers Modified: SEE ATTATCHED NOTE	**Assembler/Compiler Used:** MDS 8080 Macro Assembler
RAM Required: SEE ATTATCHED NOTE	**Programmer:** William J. Long
ROM Required: NONE	**Company:** Sullivan, Long & Hagerty
Maximum Subroutine Nesting Level: SEE ATTATCHED NOTE	**Address:** P.O. Box 2247 Birmingham,Al.35201

MLOAD is designed to operate as an ISIS System command (e.g. EDIT, DEBUG, etc.). MLOAD is therefore called in the same way as an ISIS command, and is provided with parameters in the same way. MLOAD will use about 500 bytes of memory (between addresses 3000 and 31FF Hex.) and will alter all registers. The user must assure that programs loaded by MLOAD will not overlay this area of memory as unpredictable results will occur. After performing the MLOAD function, control is returned to the MDS Monitor if the load is successful, else control is returned to ISIS.

MLOAD does not perform a Relocate function, and the binary file which is loaded is not usually executable. MLOAD simply loads the binary file at an adjusted starting address, usually for the purposes of PROM programming.

No test program is provided because any ISIS binary file will suffice. However, a sample load procedure is shown below.

ABBREVIATED TEST PROGRAM LISTING

```
001 0000 F3            DI;
002 0001 06FF          MVI B,0FFH;
003 0003 0EFF   STLOP: MVI C,0FFH;
004 0005 0D     STLLP: DCR C;
005 0006 C20500        JNZ STLLP;
006 0009 05            DCR B;

        - ETC -
```

Now suppose that this program was assembled, and the output of the assembly, when passed through HEXBIN was contained in a file called :F1:BIN.FIL . If the user wanted to load this program onto PROM chips, a DEBUG command could not be used. MLOAD would be used in this case, and the following command would be entered on the MDS console:

```
MLOAD :F1:BIN.FIL$=3200H
```

This would cause the binary file to be loaded into MDS RAM, beginning at address 3200H (AAAA + BIAS, which in this case, AAAA = 0000H and BIAS = 3200H). Control would then pass to the MDS Monitor, where the following command :

```
D3200,3209
```

would produce the following list:

```
3200 F3 06 FF 0E FF 0D C2 05 00 05
```

The PROM could then be programmed by entering:

```
PTX3200,3209,00
```

```
                ;REF. NO. AB37
                ;PROGRAM TITLE MLOAD
                ;
                ;
                ;
                ;*********************************************
                ;*****           MDS/DOS                 *****
                ;*****    MEMORY LOAD FROM DISK          *****
                ;*****      WITH BIAS OFFSET             *****
                ;*********************************************
0000            OPEN    EQU     0;
0001            CLOSE   EQU     1;
0003            READ    EQU     3;
0006            LOAD    EQU     6;
000C            ERROR   EQU    12;
0009            EXIT    EQU     9;
                ;
0040            ISIS    EQU    64;
                ;
3000                    ORG     03000H;
3000 319F31     START:  LXI     SP,SPADD;       LOAD STACK POINTER
3003 0E03               MVI     C,READ;         READ CONSOLE
3005 11B430             LXI     D,RDDCB;
3008 CD4000             CALL    ISIS;
300B 3A7B31             LDA     STATUS;
300E B7                 ORA     A;
300F C25A30             JNZ     ERRTN;          JUMP IF ERROR
3012 2A3E31             LHLD    ACTUAL;         GET BYTEYTE COUNT
3015 EB                 XCHG;
3016 21BE30             LXI     H,BUFFER;       TRANSFER FILE NAME TO LOAD CALL
3019 014A31             LXI     B,LDFIL;
301C 7E         FILNL:  MOV     A,M;
301D FE24               CPI     '$';            TEST FOR END OF FILE NAME
301F C22C30             JNZ     GTBIA;
3022 02                 STAX    B;              STORE IN LOAD CALL FILE NAME BUF
3023 23                 INX     H;              INCREMENT BUFFER ADDRESSES
3024 03                 INX     B;
3025 15                 DCR     D;              DECREMENT COUNT
3026 CA6530             JZ      STXER;          JUMP TO ERROR IF  COUNT EXCEDED
3029 C31C30             JMP     FILNL;
302C 3E20       GTBIA:  MVI     A,' ';          INSERT BLANK AFTER FILENAME
302E 02                 STAX    B;
302F 0606               MVI     B,006H;         LOAD COUNT
3031 23                 INX     H;
3032 7E                 MOV     A,M;
3033 FE3D               CPI     '=';            CHECK FOR =
3035 C26530             JNZ     STXER;          JUMP IF SYNTEX ERROR
3038 CD7630             CALL    CONVB;          CONVERT 2 HIGH ORDER HEX DIGITS
303B 324331             STA     LDBIA+1;        STORE IN LOAD CALL BIAS WORD
```

```
303E CD7630            CALL    CONVB;          CONVERT 2 LOW ORDER HEX DIGITS T
3041 324231            STA     LDBIA;
3044 23                INX     H;              CHECK FOR 'H' AFTER ADDRESS KEYI
3045 7E                MOV     A,M;
3046 FE48              CPI     'H';
3048 C26530            JNZ     STXER;          ERROR IF NOT 'H'
304B 0E06              MVI     C,LOAD;
304D 114031            LXI     D,LDDCB;        LOAD FILE AND TRANSFER CONTROL
3050 CD4000            CALL    ISIS;           TO MONITOR, ELSE ERROR
3053 3A7B31            LDA     STATUS;
3056 B7                ORA     A;
3057 CA6D30            JZ      FINISH;
305A 0E0C      ERRTN:  MVI     C,ERROR;
305C 117B31            LXI     D,ERDCB;
305F CD4000            CALL    ISIS;           OUTPUT ERROR MESSAGE
3062 C36D30            JMP     FINISH;         JUMP TO EXIT
3065 3E64      STXER:  MVI     A,100;          LOAD ERROR CODE
3067 327B31            STA     STATUS;
306A C35A30            JMP     ERRTN;
306D 0E09      FINISH: MVI     C,EXIT;
306F 114031            LXI     D,LDDCB;
3072 CD4000            CALL    ISIS;
3075 76                HLT;
3076 CDA430    CONVB:  CALL    GETBI;          GET HEX VALUE CONVERT
3079 17                RAL;
307A 17                RAL;
307B 17                RAL;
307C 17                RAL;
307D E6F0              ANI     0F0H;           MASK OUT LOW-ORDER 4 BITS
307F F5                PUSH    PSW;            SAVE
3080 CDA430            CALL    GETBI;
3083 E60F              ANI     00FH;           MASK OUT HIGH-ORDER 4 BITS
3085 4F                MOV     C,A;
3086 F1                POP     PSW;
3087 B1                ORA     C;
3088 C9                RET;
3089 F5        CVTHB:  PUSH    PSW;            CONVERSION ROUTINE HEX-TO-BINARY
308A AF                XRA     A;
308B 57                MOV     D,A;            ZERO BINARY COUNTER
308C 016A31            LXI     B,CONTA;        CONVERSION TABLE
308F F1                POP     PSW;
3090 5F                MOV     E,A;            GET HEX CHARACTER
3091 0A        CVTLP:  LDAX    B;              GET TABLE VALUE
3092 FE47              CPI     'G';            TEST FOR END OF TABLE
3094 CAA030            JZ      CVRER;          END OF TABLE? JUMP TO ERROR
3097 BB                CMP     E;              TEST FOR MATCH
3098 CAA230            JZ      CVRED;          JUMP IF MATCH
309B 03                INX     B;              INCREMENT TABLE POINTER
309C 13                INX     D;              INCREMENT BINARY COUNTER
309D C39130            JMP     CVTLP;          GO BACK AND TRY, TRY AGAIN
```

```
30A0 16FF      CVRER:  MVI     D,0FFH;         LOAD ERROR RETURN CODE
30A2 7A        CVRED:  MOV     A,D;            LOAD BINARY VALUE IN ACC.
30A3 C9                RET;
30A4 23        GETBI:  INX     H;              INCREMENT TABLE ADDRESS
30A5 7E                MOV     A,M;
30A6 CD8930            CALL    CVTHB;
30A9 FEFF              CPI     0FFH;           TEST FOR ERROR
30AB CAAF30            JZ      GETER;          JUMP IF ERROR
30AE C9                RET;
30AF F1        GETER:  POP     PSW;            STACK ADJUST
30B0 F1                POP     PSW;            STACK ADJUST
30B1 C36530            JMP     STXER;          JUMP TO SYNTAX ERROR ROUTINE
               ;
               RDDCB:
30B4 0100      RDAFT:  DW      1;              CONSOLE READ AFT
30B6 BE30              DW      BUFFER;
30B8 8000              DW      128;
30BA 3E31              DW      ACTUAL;
30BC 7B31              DW      STATUS;
30BE           BUFFER: DS      128;            READ BUFFER
313E           ACTUAL: DS      2;              BYTE COUNT
               ;
               LDDCB:
3140 4A31              DW      LDFIL;          POINTER TO FILENAME
3142 0000      LDBIA:  DW      0;              BIAS
3144 0200      LDRTS:  DW      2;              RETURN SWITCH
3146           LDENT:  DS      2;              ENTRY POINY SFFTRDD (IGNORED)
3148 7B31              DW      STATUS;
314A           LDFIL:  DS      32;             FILE NAME
               ;
               ;       ASCII CONVERSON TABLE
316A 30313233  CONTA:  DB      '0123456789ABCDEFG'
316E 34353637
3172 38394142
3176 43444546
317A 47


               ;
               ERDCB:
317B           STATUS: DS      2;
317D 7B31              DW      STATUS;
317F           FILLER: DS      32;             STACK AREA
319F 0000      SPADD:  DW      0;              HIGH STACK ADDRESS
               ;
0000           END;
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

intel

Ref. No. AB38

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

HEX TAPE LOADER FOR SDK

Function

LOADS HEX ASSEMBLER OBJECT TAPE INTO RAM OF SDK

Required Hardware

INTEL MCS-80 SDK, ASR-33 TTY

Required Software

SDK MONITOR PROM

Input Parameters

PAPER TAPE HEX TAPE FORMATTED BY INTEL OR TIMESHARE ASSEMBLERS.

INSTRUCTIONS: 1) LOAD TAPE IN READER

2) TYPE (AFTERMONITOR PROMPT) G 138Ø (CR)

3) TURN ON READER

Output Results

HEX INFORMATION IS LOADED INTO MEMORY AND CONTROL RETURNS TO MONITOR.

A CHECKSUM ERROR CAUSES THE SIGNON MESSAGE TO BE PRINTED. A GOOD LOAD RESULTS IN A MONITOR PROMPT "." FOLLOWED BY SOME "*" ERROR SYMBOLS UNTIL READER IS STOPPED.

Registers Modified: N/A	**Assembler/Compiler Used:** ISIS 8080 MACRO ASSM., V1.0
RAM Required: 256 BYTES (KIT SUPPLIED)	**Programmer:** A.C. MARSHALL
ROM Required: SDK MONITOR PROM	**Company:** PROTEON ASSOCIATES, INC.
Maximum Subroutine Nesting Level: NA	**Address:** 24 CRESCENT STREET WALTHAM, MA 02154

```
                ;REF. NO. AB38
                ;PROGRAM TITLE HEX TAPE LOADER FOR SDK
                ;
                ;A. C. MARSHALL, FEB 4, 1976
                ;
                ;HEX TAPE LOADER FOR SDK
                ;
                ; LOAD INTO MEMORY USING I COMMAND
                ; POSITION IN LEADER OF '0'S' , TYPE I,
                ; THEN TURN ON READER.
                ;  TO RUN TAPE LOADER TYPE G1380 CR.
                ;  POSITION TAPE WITHIN 1 INCH OF BEGIN
                ;  RECORD MARK THEN TURN ON READER.
                ;
                ;  A GOOD LOAD WILL RETURN TO MONITOR . WITH
                ;  SOME '*' FOR EACH BLANK CHAR READ FROM
                ;  TRAILER.
                ;  A CHECKSUM ERROR WILL RETURN THROUGH
                ;  MONITOR SIGNON MESSAGE.
                ;   TO SEE WHAT HAPPENED USE X COMMAND.
                ;    E REG CONTAINS ERROR ON READ.
                ;
                ;
1380                    ORG 1380H
1380 060A       ENTR:   MVI B,10    ;LEADER LENGTH 1"
1382 CD1B02     GET:    CALL GETCH  ;GET A CHAR
1385 05                 DCR B       ;1 "?
1386 CA2B00             JZ GETCM    ;NO RECORD FOUND, RET TO MON
1389 FE3A               CPI ':'     ;BEGIN REC?
138B C28213             JNZ GET     ;NO, GET ANOUHER
138E 1E00               MVI E,0     ;CLEAR E REG
1390 CDB313             CALL INPBY
1393 A7                 ANA A       ;CLEAR ACCUM
1394 CA2B00             JZ GETCM    ;RECORD LENG=00, RET TO MON
1397 42                 MOV B,D     ;SAVE RECORD SIZE IN B
1398 CDB313             CALL INPBY  ;GET HI ADDR
139B 62                 MOV H,D     ;PUT IN H REG
139C CDB313             CALL INPBY  ;GET LO ADDR
139F 6A                 MOV L,D     ;PUT IN L REG
13A0 CDB313             CALL INPBY  ;RECORD TYPE, IGNORE
13A3 CDB313     DATA:   CALL INPBY  ;GET HEX DATA
13A6 05                 DCR B       ;FINISHED REC?
13A7 72                 MOV M,D     ;STORE DATA
13A8 23                 INX H       ;INCREMENT MEMORY ADDR
13A9 C2A313             JNZ DATA    ;LAST ONE?
13AC CDB313             CALL INPBY  ;YES, GET CHECKSUM
13AF CA8013             JZ  ENTR    ;CHECKSUM IS OK, START OVER
13B2 CF                 RST 1       ;CHECKSUM ERROR RTN TO MON
```

```
                ;SUBROUTINE
13B3 CD1B02     INPBY:CALL GETCH  ;GET A CHAR
13B6 4F               MOV C,A     ;PUT IN A
13B7 CDDA01           CALL CNVBN ;CONVERT TO BINARY
13BA 0F               RRC
13BB 0F               RRC
13BC 0F               RRC
13BD 0F               RRC         ;MOVE IT TO MSD
13BE 57               MOV D,A    ;SAVE FIRST HALF
13BF CD1B02           CALL GETCH ;INP CHAR
13C2 4F               MOV C,A     ;PUT IN C
13C3 CDDA01           CALL CNVBN ;CONVERT TO BIN
13C6 82               ADD D       ;FORM BYTE
13C7 57               MOV D,A     ;CALC CKSUM
13C8 83               ADD E       ;SAVE NEW CKSUM
13C9 5F               MOV E,A     ;SAVE CKSUM
13CA C9               RET
                ;
                ;EQUATES
                ;
01DA            CNVBN EQU 1DAH   ;MONITOR ROUTINE
  21B           GETCH EQU 21BH   ;MONITOR ROUTINE
000F            LSD   EQU 0FH
002B            GETCM EQU 02BH   ;MONITOR ROUTINE
0000                  END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB39

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

HEX FORMAT PAPER TAPE DUMP

Function

DUMP SECTIONS OF STORED PROGRAM IN HEX FORMAT (AS IS PRODUCED BY INTEL ASSEMBLER). NO LEADER IS PRODUCED SO THAT MULTIPLE SECTIONS MAY BE CONCATENATED INTO ONE TAPE

NOTE: EACH "." MONITOR PROMPT MUST BE "RUB OUT" OVERPUNCHED TO LOAD WITH ATTACHED LOADER.

Required Hardware

INTEL MCS-80 SDK, ASR-33 TTY.

Required Software

SDK MONITOR PROM

Input Parameters

MEMORY START, FINISH ADDRESSES:

INSTRUCTIONS: IN MONITOR

1) TYPE .G 13ØØ (CR)

XXXX,YYYY (TURN ON PUNCH) (CR)

Output Results

HEX FORMAT PAPER TAPE. LEADERS SHOULD BE PRODUCED MANUALLY, OFF LINE.

Registers Modified: NA	**Assembler/Compiler Used:** ISIS 8080 MACRO ASSEMBLER, V1.0
RAM Required: SDK SUPPLIED (256 BYTE)	**Programmer:** A.C. MARSHALL
ROM Required: SDK MONITOR PROM	**Company:** PROTEON ASSOCIATES, INC.
Maximum Subroutine Nesting Level: NA	**Address:** 24 CRESCENT ST. WALTHAM, MA 02154

98-034B

```
                ;REF. NO. AB39
                ;PROGRAM TITLE HEX FORMAT PAPER TAPE DUMP
                ;
                ;
                ;
                ;
                ; A.C. MARSHALL, FEB 4, 1976
                ;
                ;HEX FORMAT FORMAT DUMP ROUTINE
                ; USES MONITOR IN SDK
                ;  COMMAND IS G1300
                ;    THEN GIVE TWO ADDRESSES IN STD MON
                ;    FORMAT.
                ;    TURN ON PUNCH THEN HIT CR
                ;
1300                    ORG     1300H
1300 0E02       DUMP:   MVI     C,2         ;INIT TO 2 ADDR
1302 CD5702             CALL    GETNM
1305 D1                 POP     D           ;END ADDR
1306 E1                 POP     H           ;START ADDR
 307 CDEE01     DCM05:  CALL    CROUT       ;RETN
130A 0E3A               MVI     C,':'       ;RECORD FORM
130C CDFA03             CALL    CO          ;OUTPUT IT
130F 7B                 MOV     A,E         ;LSD OF ADDR END
1310 95                 SUB     L           ;LSD OF PRLSENT START ADDR
1311 FE0F               CPI     NEWLN
1313 D24C13             JNC     MAX         ;>=?,YES
1316 3C                 INR     A           ;EQU MEANS ONE LOCATION
1317 47                 MOV     B,A         ;SAVE A
1318 AF         FULL:   XRA     A           ;CALEAR CKSUM
1319 326713             STA     TEMP        ;CLEAR CKSUM
131C 78                 MOV     A,B         ;RECALL A
131D CD5A13             CALL    CKSUM       ;OUTPUT REC LENGTH
1320 7C                 MOV     A,H         ;GET HI ADDR
1321 CD5A13             CALL    CKSUM       ;OUTPUT HI ADDR
1324 7D                 MOV     A,L         ;GET LO ADDR
1325 CD5A13             CALL    CKSUM       ;OUTPUT LO ADDR
1328 AF                 XRA     A           ;CLEAR A
1329 CD5A13             CALL    CKSUM       ;OUTPUT REC TYPE
132C 7E         DCM10:  MOV     A,M         ;GET BYTE
132D CD5A13             CALL    CKSUM       ;OUTPUT BYTE
1330 CD9C02             CALL    HILO        ;DONE?
1333 DA4313             JC      CKOUT       ;YES , FINISH OFF
1336 23                 INX     H           ;NO, NEW ADDR
1337 7D                 MOV     A,L         ;ADDR TEST
 338 E60F               ANI     NEWLN       ;NEW LINE?
133A C22C13             JNZ     DCM10       ;NO, NEXT BYTE
133D CD5113     RCEND:  CALL    CKEND       ;YES, SEND CKSUM
1340 C30713             JMP     DCM05       ;DO NEW RECORD
```

```
1343 CD5113    CKOUT:  CALL    CKEND   ;SEND CKSUM
1346 CDEE01            CALL    CROUT   ;
1349 C32B00            JMP     GETCM   ;DONE REENTER MONITOR
134C 0610      MAX:    MVI     B,10H   ;MAX RECORD SIZE
134E C31813            JMP     FULL
1351 3A6713    CKEND:  LDA     TEMP    ;GET CKSUM
1354 2F                CMA
1355 3C                INR     A       ;NEGATE
1356 CDC302            CALL    NMOUT   ;OUTPUT
1359 C9                RET
135A 47        CKSUM:  MOV     B,A     ;SAVE B
135B 3A6713            LDA     TEMP    ;GET OLD CKSUM
135E 80                ADD     B       ;ADD NEW TO OLD
135F 326713            STA     TEMP    ;SAVE NEW CKSUM
1362 78                MOV     A,B     ;RESTORE A
1363 CDC302            CALL    NMOUT   ;OUTPUT DIGIT
1366 C9                RET             ;RETURN
1367 00        TEMP:   DB      0       ;CKSUM STORAGE
0257           GETNM   EQU     0257H   ;MON ROUTINE
01EE           CROUT   EQU     1EEH    ;MON ROUTINE
03FA           CO      EQU     3FAH    ;MON ROUTINE
 2C3           NMOUT   EQU     2C3H    ;MON ROUTINE
029C           HILO    EQU     29CH    ;MON ROUTINE
000F           NEWLN   EQU     0FH     ;MASK
002B           GETCM   EQU     2BH     ;MON ENTRY POINT
0000                   END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB40

☐ 4004 ☐ 4040 ☐ 8008 XX8080

(use additional sheets if necessary)

Program Title

PAPER TAPE REFORMATTER

Function

REFORMATS THE "DUMP" (D INSTRUCTION) TAPE OF THE 80-SDK MONITOR INTO A COMPATIBLE FORMAT FOR RELOADING WITH THE "INSERT" (I) INSTRUCTION.

Required Hardware

MCS-80-SDK ASR-33 OR EQUIVALENT PAPER TAPE TERMINAL.
NO MODIFICATIONS OF KIT NECESSARY.

Required Software

MCS 80-SDK PROM MONITOR REVISION AS LISTED IN INTEL SDK USERS GUIDE PRELIMINARY EDITION (98-203A)

Input Parameters

PAPER TAPE FORMED BY USING D INSTRUCTION WITH LEADER.
PROGRAM LOOKS FOR 2 CR, LF CHARACTERS TO BEGIN EXECUTION.

INSTRUCTIONS:
1) TURN ON PUNCH
2) LOAD TAPE IN READER
3) TYPE IN (AFTER MONITOR PROMPT) .G 1300 (CR)
4) TURN ON READER

Output Results

PAPER TAPE IS OUTPUT WITH A LEADER OF ASCII "Ø". THIS ALLOWS READ AS FOLLOWS:

INSTRUCTIONS:
1) LOAD OBJECT TAPE SET TO NEAR END OF "Ø" LEADER
2) TYPE IN .I
3) TURN ON READER
4) REMAINDER OF INSERT INSTRUCTION IS ON TAPE

Registers Modified: NA	**Assembler/Compiler Used:** ISIS 8080 MACRO ASSEMBLER V1.0
RAM Required: KIT SUPPLIED 256 Bytes	**Programmer:** A.C. MARSHALL
ROM Required: SDK MONITOR	**Company:** PROTEON ASSOCIATES, INC.
Maximum Subroutine Nesting Level: NA	**Address:** 24 CRESCENT STREET WALTHAM, MA 02154

98-034B

```
                        ;REF. NO. AB40
                        ;PROGRAM TITLE PAPER TAPE REFORMATTER
                        ;
                        ;
                        ;
                        ;A. C. MARSHALL, FEB 4, 1976
                        ;
                        ;TAPE REFORMATTING ROUTINE
                        ; TAPE IN THE FORM OF A D COMMAND
                        ; FROM THE SDK IS REFORMATTED IN
                        ; THE FORM FOR AN INSERT 'I' COMMAND
                        ; FOR THE MONITOR
                        ;
                        ;     START THE OBJECT TAPE IN THE READER
                        ;     AT THE FIRST CHARACTER. WHEN THE '.'
                        ;     FROM THE MONITOR IS RECEIVED, TURN ON THE
                        ;     TAPE READER.
                        ;
01E3                    CO      EQU     01E3H   ;MONITOR ROUTINE
000A                    LF      EQU     0AH     ;MONITOR ROUTINE
 02B                    GETCM   EQU     02BH    ;MONITOR ROUTINE
01EE                    CROUT   EQU     1EEH    ;MONITOR ROUTINE
021B                    CI      EQU     021BH   ;MONITOR ROUTINE
021B                    GETCH   EQU     021BH   ;MONITOR ROUTINE
001B                    ESC     EQU     1BH
000D                    CR      EQU     0DH
0014                    LDRLN   EQU     20
                        ;
1300                            ORG     1300H
1300 1614               ENTRY:  MVI     D,LDRLN ;SIZE OF LEADER
1302 0E30                       MVI     C,'0'   ;ZERO FOR LEADER
1304 CDE301             LEADER: CALL    CO      ;OUTPUT LEADER
1307 15                         DCR     D       ;DONE?
1308 C20413                     JNZ     LEADER  ;NO, LOOP
130B CD1B02             START:  CALL    GETCH   ;INPUT CHARACTER
130E FE0A                       CPI     LF      ;LINE FEED?
1310 C20B13                     JNZ     START   ;NO, GET ANOTHER
1313 CD1B02                     CALL    GETCH   ;RECORD
1316 CD1B02                     CALL    GETCH   ;/LENGTH, IGNORE
1319 1E06                       MVI     E,6     ;6 CHAR
131B 1604                       MVI     D,4     ;YES, GET ADDR.
131D CD7113             ADRI:   CALL    CARECH  ;OUTPUT CHAR
1320 15                         DCR     D       ;ALL 4?
1321 C21D13                     JNZ     ADRI    ;NO, REPEAT
1324 CD1B02                     CALL    GETCH   ;BLANK- OMIT
 327 0E0D                       MVI     C,CR    ;SET UP CR
_329 CDE301                     CALL    CO      ;OUTPUT IT
132C CD7113             DATA:   CALL    CARECH  ;DIGIT 1
132F CD7113                     CALL    CARECH  ;DIGIT 2
```

```
1332 3E0D               MVI     A,CR    ;BLANK- OMIT
1334 3E0D               MVI     A,CR    ;3RD CHAR IN A
1336 B9                 CMP     C       ;CR?
1337 C22C13             JNZ     DATA    ;NO, REPEAT
133A 0E2C               MVI     C,','   ;YES, LINE DONE
133C CDE301             CALL    CO      ;OUTPUT ','
133F CD1B02             CALL    GETCH   ;LINE FEED - NO OUT
1342 CD1B02             CALL    GETCH   ;HX DGT OR '.'
1345 3E2E               MVI     A,'.'   ;CHECK END
1347 B9                 CMP     C       ;END?
1348 C26113             JNZ     NXTLN   ;NO, DO AGN
134B 0E1B               MVI     C,ESC   ;YES
134D CDE301             CALL    CO      ;OUTPUT ESC
1350 0E00       ENDTP:  MVI     C,0     ;BLANK FOR TRAILER
1352 1614               MVI     D,LDRLN ;SIZE OF LEADER
1354 CDE301     TRAIL:  CALL    CO      ;OUTPUT BLANK
1357 15                 DCR     D       ;DONE?
1358 C25413             JNZ     TRAIL   ;NO, CONTINUE
135B CDEE01             CALL    CROUT   ;YES, RETURN
135E C32B00             JMP     GETCM   ;/TO MONITOR
1361 1604       NXTLN:  MVI     D,4     ;SET COUNTER TO 4
 363 1D                 DCR     E       ;DONE 5 REC?
1364 CA5013             JZ      ENDTP   ;YES END
1367 CD1B02     NADR:   CALL    GETCH   ;NO, GET A CHAR
136A 15                 DCR     D       ;DONE?
136B C26713             JNZ     NADR    ;NO, LOOP
136E C32C13             JMP     DATA    ;DO NEXT RECORD
                ;SUBROUTINES:
                ;
1371 CD1B02     CARECH: CALL    GETCH   ;GET CHAR
1374 CDE301             CALL    CO      ;ECHO  IT
1377 C9                 RET             ;RETURN
0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

intel

Ref. No. AB41

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 (use additional sheets if necessary)

Program Title

DATA IO PROM TAPE PROCESSOR V1.0 DATA IO

Function

CREATE, EDIT AND PUNCH PAPER TAPE IN THE BINARY FORMAT REQUIRED FOR DATA IO PROM PROGRAMMERS

Required Hardware

CONSOLE, INTELLEC MDS-800, PAPER TAPE READER, PAPER TAPE PUNCH

Required Software

MDS-800 MONITOR

Input Parameters

HEX INPUT FROM CONSOLE, HEX OR BINARY TAPE INPUT

Output Results

PAPER TAPE IN BINARY IMAGE FORMAT, AND LISTINGS

Program offered on diskette only.

Registers Modified: ALL	**Assembler/Compiler Used:** ASM80 MACRO ASSEMBLER
RAM Required: FCBH - 4043D	**Programmer:** J. WILLOTT
ROM Required: NONE	**Company:**
Maximum Subroutine Nesting Level: LESS THAN 6	**Address:** 1487 BONGATE COURT SAN JOSE, CA. 95130

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AC13

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 (use additional sheets if necessary)

Program Title

CENTRONICS Assembler oriented
306 Line Printer Handler and Error Only Assembly

Function

Allows assemblies to be performed on Centronics 306 Line Printer without top of form option.

Two listings are available 1) errors only
2) normal

Required Hardware

See attached for interface detail.

Line printer on output port 7
input port 4

Required Software

Normal 8080 Monitor and Assembler

Input Parameters

G 37E0 Error Only Listing
B 37F0 Final Listing

These can be set or changed at any time by re-entering the monitor with a console reset.

Output Results

The routine can be modified to provide both listings on a teletype by altering the following instructions and reassembling.

```
Line 15          ANI 80H  to  ANI 04H
     29  Status  EQU 4    to  Status EQU 1
     30  LPRT    EQU 7    to  LPRT   EQU 0
```

Delete lines 20 thru 25 ADI 80H to out LPRT

Replace LINED: RET with LINEND: MVI A,0AH
OUT LPRT
RET

Registers Modified:	**Assembler/Compiler Used:** 8080
RAM Required: 1 page	**Programmer:** R. Perry and D. Rush
ROM Required: 1 page *may be located in RAM if desired	**Company:** Westinghouse Transportation
Maximum Subroutine Nesting Level:	**Address:** 2001 Lebanon Road West Mifflin, PA 15122

98-034B

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AA9

☐ 4004 ☐ 4040 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program Title

Memory Test 8080

Function

To perform a functional test of the ram memory within a reasonable time. Using modified walking ones and zeroes test pattern.

Required Hardware

MDS-800 with List output device... May use console.

Required Software

MDS-800 monitor list output subroutine for printing error messages.

Input Parameters

4 equate card describing the dimentionality of the memory chips, the start/end address of memory to be tested, the origin of the program and the address of the character output subroutine.

Output Results

Returns to monitor if no errors... else prints each memory location in error, the data received from memory, and the test data written. Test time for 4K bytes of memory is one minute.

Registers Modified: All	**Assembler/Compiler Used:** ISIS 8080 Macro Assembler, V1.0
RAM Required: 286 Bytes	**Programmer:** S. G. Thompson
ROM Required: Intel MDS-800 Monitor I/O Subroutines	**Company:** Harris Controls
Maximum Subroutine Nesting Level: 5 + 4 pushes	**Address:** P. O. Box 430 Melbourne, Florida 32901

```
                        ;REF. NO. AA9
                        ;PROGRAM TITLE MEMORY TEST 8080
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;       MEMORY TEST
                        ;
1000                    MEMS    EQU   1000H       ;MEMORY START ADR FOR TESTING
4000                    MEME    EQU   4000H       ;MEMORY END ADR
                        ;       MEMORY CHIP DIMENTIONALTY 64 BY 64
0040                    MDEM    EQU   64              ; 4K MEM CHIP SUCH AS 2107 INTEL
0000                    DATAS   EQU   000H        ;STARTING MEMORY DATA PATTERN (BACKGROU
0041                    MPINC   EQU   MDEM+1          ; POINTER INC. VALUE FOR NEXT ALT P
                        ;
                        ;               MDS-800   SUBROUTINES USED
                        ;                          L0     (LIST OUTPUT I/O)
                        ;                          START  (OF MONITOR)
 30F                    L0      EQU   0F80FH
F826                    START   EQU   0F826H          ; MONITOR ADR. . STACK RESET
                        ;
0100                            ORG   100H        ;PGM START ADR
0100 311E02                     LXI   SP,STACK        ; SET STACK ADR.
0103 0E00                       MVI   C,0         ;FIRST TIME FLAG
0105 0600                       MVI   B,DATAS     ;B=DATA
0107                    NPAT    EQU   $               ; DO NEXT PATTERN
0107 AF                         XRA   A               ; CLEAR OFSET
0108 32EB01                     STA   OFSET
010B 210010             BEGIN:  LXI   H,MEMS      ;HL= MEM START ADR
010E CD5001                     CALL  BASE            ; GO TO CALC BASE ADR FOR READ WRIT
0111 CD5C01             WRITE:  CALL  WCHK        ;WRITE PROPER PATTERN AT THIS ADR.
0114 23                         INX   H           ;NEXT MEM ADR
0115 7C                         MOV   A,H         ;CHECK FOR DONE WITH WRITE
0116 FE40                       CPI   MEME/256    ; HIDH ADR. CHECK
0118 C21101                     JNZ   WRITE       ;LOOP TIL DONE
011B 7D                         MOV   A,L
011C FE00                       CPI   MEME-(MEME/256)*256 ; LOW
011E C21101                     JNZ   WRITE
0121 210010                     LXI   H,MEMS      ;BACK TO START FOR READ
0124 CD5001                     CALL  BASE            ; GO CALC BASE ADR FOR READ
0127 CD6F01             READ:   CALL  RCHK        ;READ & CHECK FOR OK
012A 23                         INX   H
012B 7C                         MOV   A,H         ;CHECK FOR DONE WITH READ
012C FE40                       CPI   MEME/256    ; HIDH ADR. CHECK
012E C22701                     JNZ   READ
0131 7D                         MOV   A,L
0132 FE00                       CPI   MEME-(MEME/256)*256 ; LOW
```

```
0134 C22701             JNZ   READ
                ;       CHECK FOR DONE . . OFSET=MDEM
0137 3AEB01             LDA   OFSET
013A 3C                 INR   A
013B 32EB01             STA   OFSET
013E FE41               CPI   MDEM+1
0140 C20B01             JNZ   BEGIN           ; CONTINUE TILL DONE
0143            NXT     EQU   $
0143 79                 MOV   A,C
0144 B7                 ORA   A
0145 C226F8             JNZ   START       ;BACK TO MONITOR
0148 78                 MOV   A,B
0149 2F                 CMA
014A 47                 MOV   B,A             ; DATA NOW COMPLEMENTED
014B 0EFF               MVI   C,0FFH          ; SET SECOND TIME FLAG
014D C30701             JMP   NPAT            ; GO TO NXT PATTERN
                ;
                ;       SUBROUTINE TO CALC BASE ADR. FOR ALT BATTERN START
0150            BASE    EQU   $
0150 110010             LXI   D,MEMS          ; MEM START
0153 3AEB01             LDA   OFSET           ; PLUS OFSET
 156 83                 ADD   E
0157 5F                 MOV   E,A
0158 AF                 XRA   A
0159 8A                 ADC   D               ; ADD IN ANY CARRY
015A 57                 MOV   D,A             ; RESTORE
015B C9                 RET
                ;
                ;       SUBROUTINE TO WRITE DATA PATTERN INTO MEMORY
                ;
015C            WCHK    EQU   $
015C 7C                 MOV   A,H         ;CHECK FOR ALTERNATE DATA PATTERN ADR
015D BA                 CMP   D
015E C26D01             JNZ   W2          ;NOT ADR. . . WRITE BACKGROUND
0161 7D                 MOV   A,L
0162 BB                 CMP   E
0163 C26D01             JNZ   W2          ;NOT ADR. . . WRITE BAKGND ELSE. . .
0166 78                 MOV   A,B         ;COMPLEMENT PATTERN AND. . .
0167 2F                 CMA
0168 77                 MOV   M,A         ;WRITE
0169 CD9301             CALL  INCD            ; INC ALT DATA PATTERN POINTER
016C C9                 RET
016D 70         W2:     MOV   M,B         ;STORE BACKGROUND DATA PATTERN
016E C9                 RET
                ;
                ;       SUBROUTINE TO READ AND CHECK MEMORY DATA
                ;
 16F            RCHK    EQU   $
016F 7C                 MOV   A,H         ;CHECK ADR
0170 BA                 CMP   D
```

```
0171 C28A01             JNZ     R2
0174 7D                 MOV     A,L
0175 BB                 CMP     E
0176 C28A01             JNZ     R2
0179 C5                 PUSH    B               ;B = CALC. DATA WRITTEN
017A 78                 MOV     A,B
017B 2F                 CMA                     ;A = DATA FROM MEMORY
017C 47                 MOV     B,A
017D 7E                 MOV     A,M             ;GET DATA
017E B8                 CMP     B               ;CHECK CELL FOR ERROR
017F CA8501             JZ      $+6
0182 CD9B01             CALL    ERROR           ;     GO TO PRINT OUT
0185 C1                 POP     B               ;RESTORE REGS
0186 CD9301             CALL    INCD                ; INC. ALT DATA PATTERN POINTER
0189 C9                 RET
018A 7E         R2:     MOV     A,M             ;A = MEM DATA
018B B8                 CMP     B               ; B= DATA WRITTEN
018C CA9201             JZ      $+6
018F CD9B01             CALL    ERROR           ;IF ERROR PRINT ELSE BACK TO CALLER
0192 C9                 RET
                ;
                ;       SUBROUTINE TO INC. ALT DATA PATTERN POINTER
0193            INCD    EQU     $
0193 E5                 PUSH    H                   ; SAVE HL
0194 214100             LXI     H,MPINC             ;GET INC. VALUE
0197 19                 DAD     D                   ; HL <= ALT. PATTERN POINTER + INC.
0198 EB                 XCHG                        ; BACK IN DE
0199 E1                 POP     H                   ; RESTORE HL
019A C9                 RET
                ;
                ;       MEMORY ERROR PRINTING SUBROUTINE
                ;
                ; LINE FORMAT:
                ;AAAA XXM YYT
                ;           XX= MEMORY DATA IN ERROR
                ;           YY= TEST DATA PATTERN
                ;           AAAA= ADDRESS OF MEM FAILURE
                ; A, X, Y ALL IN HEX
019B            ERROR   EQU     $
019B C5                 PUSH    B               ;BC SAVED
019C F5                 PUSH    PSW             ;ACC SAVED
019D 0E0D               MVI     C,0DH           ;CARRIAGE RET
019F CD0FF8             CALL    L0
01A2 0E0A               MVI     C,0AH           ;LINE FEED
01A4 CD0FF8             CALL    L0
01A7 CDC701             CALL    DADR            ;MEM ADR PRINT
01AA 0E20               MVI     C,' '           ;BLANK
01AC CD0FF8             CALL    L0
01AF F1                 POP     PSW
01B0 CDCF01             CALL    DBYTE           ;PRINT MEM DATA IN ERROR
```

```
01B3 0E4D               MVI   C,'M'
01B5 CD0FF8             CALL  L0          ;FLAG AS 'M'
01B8 0E20               MVI   C,' '       ;BLANK
01BA CD0FF8             CALL  L0
01BD C1                 POP   B
01BE 78                 MOV   A,B         ;PRINT DATA WRITTEN
01BF CDCF01             CALL  DBYTE
01C2 0E54               MVI   C,'T'       ;T = TEST PATTERN
01C4 C30FF8             JMP   L0          ;
                ;
                ;       DUMP MEMORY ADR REG TO L0
                ;
01C7            DADR    EQU   $
01C7 7C                 MOV   A,H
01C8 CDCF01             CALL  DBYTE
01CB 7D                 MOV   A,L
01CC C3CF01             JMP   DBYTE
                ;
                ;       DISPLAY ON L0 DATA BYTE IN ACC
                ;                           REG-C DESTROYED
01CF F5         DBYTE:  PUSH  PSW
 1D0 0F                 RRC               ;SHIFT OVER
01D1 0F                 RRC
01D2 0F                 RRC
01D3 0F                 RRC
01D4 CDE101             CALL  CONV        ;CONVERT TO HEX ASCII
01D7 CD0FF8             CALL  L0          ;PRINT
01DA F1                 POP   PSW         ;RESTORE LSD
01DB CDE101             CALL  CONV        ;CONV TO ASCII HEX
01DE C30FF8             JMP   L0
                ;
                ;               CONVERT 4 BIT BINARY INTO ASCII HEX
                ;                       4 BITS IN LSD OF ACC, ASCII IN REG C
01E1            CONV    EQU   $
01E1 E60F               ANI   0FH
01E3 C690               ADI   90H
01E5 27                 DAA
01E6 CE40               ACI   40H
01E8 27                 DAA
01E9 4F                 MOV   C,A
01EA C9                 RET
01EB            OFSET   EQU   $
01EB                    DS    1           ; ALT PATTERN BASE ADR.  <= OFSET+ME
                ;               STACK MEMORY . .
01EC                    DS    50
021E            STACK   EQU   $
^900                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

intel

Ref. No. AB42

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000

(use additional sheets if necessary)

Program Title

SYM

Function

SYMBOL TABLE DUMP FOR INTELLEC 8/80

Required Hardware

INTELLEC MCS, PRINTER (CONSOLE OR LIST DEVICE)

Required Software

INTEL ASSEMBLER VERSION 4.0

Input Parameters

SYMBOL TABLE AS LEFT AFTER AN ASSEMBLY

Output Results

THE ASSEMBLED SYMBOLS FOLLOWED BY THE ASSIGNED ADDRESS (VALUE). THE OUTPUT IS SIMILAR TO THE MDS ASSEMBLER SYMBOL TABLE DUMP.

Registers Modified: ALL	**Assembler/Compiler Used:** INTEL MACRO ASSM /4.0
RAM Required: NONE	**Programmer:** LORNE DOUGLAS
ROM Required: 63_{10} BYTES	**Company:** TELAID SYSTEMS
Maximum Subroutine Nesting Level: 2	**Address:** 6725 Variel Canoga Park, CA. 91303

98-034C

```
                ;REF NO.  AB42
                ;PROGRAM TITLE SYMBOL TABLE DUMP FOR MOD 80
                ;
                ;
                ;
                ;
                ;
                ;
                ;   *
                ;  *** SYMBOL TABLE DUMP FOR INTELLECT
                ;   *
                ;
                ;
                ;THIS ROUTINE PRINTS THE SYMBOL TABLE
                ;AFTER AN ASSEMBLY IS COMPLETE.  THIS
                ;PARTICULAR VERSION RUN UNDER MONITOR
                ;VERSION 3.0 AND ASSEMBLER VERSION 4.0.
                ;THIS PROGRAM HAS BEEN ASSEMBLED TO
                ;RUN IN RAM AND I HAVE MODIFIED THE
                ;BRANCH TABLE TO PRINT SYMBOLS BY
                ;TYPING V UNDER THE MONITOR.
                ;
                ;
                ;TO ADAPT TO ANOTHER ASSEMBLER OR
                ;MONITOR ONE WOULD HAVE TO FIND THE
                ;START OF THE SYMBOL TABLE IN RAM AND
                ;SUBSTITUTE THE UTILITY ROUTINE ADDR.
                ;
                ;
                ;
                ;
3368                    ORG     3368H   ;FOR PROM
                SYM:
3368 218E1D             LXI     H,1D8EH;STARTING ADDR OF
                                        ;  SYMBOL TABLE
                NXTLN:
336B CDCD3C             CALL    CRLF    ;NEW LINE TO LIST
336E 16FD               MVI     D,-3    ;3 SYMBOLS PER LINE
                NXTSYM:
3370 CD9333             CALL    PRINT   ;PRINT SYMBOL CHR
3373 CD533C             CALL    BLK     ;
3376 23                 INX     H       ;INCREMENT TO HEX
3377 23                 INX     H       ;  VALUE AND PRINT
3378 7E                 MOV     A,M     ;
3379 CDC73D             CALL    LBYTE   ;
337C 2B                 DCX     H       ;
337D 7E                 MOV     A,M     ;
337E CDC73D             CALL    LBYTE   ;
```

```
3381 23              INX   H       ;NOW INCR TO NEXT
3382 23              INX   H       ; SYMBOL
3383 CD533C          CALL  BLK     ;
3386 CD533C          CALL  BLK     ;
3389 CD533C          CALL  BLK     ;
338C 14              INR   D       ;
338D FA7033          JM    NXTSYM  ;
3390 C36B33          JMP   NXTLN
                     ;
                     PRINT:
3393 1EFB            MVI   E,-5    ;5 CHR PER SYMBOL
                     PR2:
3395 7E              MOV   A,M     ;TEST FOR END OF
3396 FE0D            CPI   0DH     ; TABLE
3398 C29E33          JNZ   PR3     ;
339B C36B38          JMP   START   ;DONE - ADIOS
                     PR3:
339E 4E              MOV   C,M     ;PRINT SYMBOL
339F CD553C          CALL  CO      ;
33A2 23              INX   H       ; NEXT CHR
33A3 1C              INR   E       ; CHR COUNT
 3A4 C8              RZ            ;
⊿3A5 C39533          JMP   PR2     ;
                     ;
                     ;
                     ;   **  MONITOR UTILITIES
                     ;
                     ;
3C53                 BLK   EQU   3C53H  ;PRINT 1 BLANK
3C55                 CO    EQU   3C55H  ;PRINT 1 ASCII CHR
3CCD                 CRLF  EQU   3CCDH  ;END OF LINE SEQNCE
3DC7                 LBYTE EQU   3DC7H  ;PRINT HEX BYTE
386B                 START EQU   386BH  ;MONITOR RESTART
                     ;
0000                 END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB43

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

MON256

Function

Provides the most commonly used monitor debug functions in a single 256-byte EPROM: Go To, Substitute, Display, Hex Arithmetic, Find Byte, and Move.

Required Hardware

As assembled requires INTELLEC 8/MOD 80 with TTY on ports 0 and 1. Can easily be reassembled with different input/output assignments for use with any user designed hardware which has ASCII I/O capability.

Required Software

None

Input Parameters

ASCII Characters 0 - 9, A - F, G, H, M and S.
Addresses must be entered as four characters.
Bytes must be entered as two characters.
Leading zeros are significant.

Output Results

ASCII characters corresponding to hexadecimal memory addresses and contents.

Registers Modified: ALL	**Assembler/Compiler Used:** 8080 MACRO ASSEMBLER, VER 3.0
RAM Required: 11 BYTES FOR STACK	**Programmer:** FRANK FAFF
ROM Required: 256 BYTES	**Company:** ATLANTIC RESEARCH CORP.
Maximum Subroutine Nesting Level: 5	**Address:** 5390 CHEROKEE AVENUE ALEXANDRIA, VA. 22314

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB44

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

CHARACTER INTERPRETED MEMORY DUMP

Function

Prints contents of selected portion of memory. Each line contains beginning address followed by 16 bytes in hex format and same 16 bytes repeated in character format. Minimum dump contains 256 bytes. All dumps are spaced to place three 256 byte blocks on an 11 inch page with column identification at the top of each page.

Required Hardware

Intellec 8/Mod 80 with TTY or equivalent.

Required Software

Intellec Monitor Version 2.0

Input Parameters

Program prompts for two addresses with an '@'. Addresses are entered in hex separated by a comma or blank as in the Monitor dump command.

Output Results

Prints memory in 256 byte blocks as described above.

Registers Modified: Monitor is reinitialized at exit.	**Assembler/Compiler Used:** ISIS 8080 Macro Assembler, V1.0
RAM Required: Program requires 242 bytes. Will execute in either RAM or ROM.	**Programmer:** Jim Squires
ROM Required:	**Company:** Allan Hancock College
Maximum Subroutine Nesting Level:	**Address:** Santa Maria, Ca 93454

```
                        ;REF. NO. AB44
                        ;PROGRAM TITLE CHARACTER INTERPRET MEMORY DUMP
                        ;
                        ;
                        ;
                        ;MEMORY DUMP WITH CHARACTER INTERPRET
                        ; DEFINE EXTERNAL REFERENCES
                        ;
3D6B                    EXPR    EQU  3D6BH
3CC8                    CRLF    EQU  3CC8H
3DBA                    LADR    EQU  3DBAH
3C4E                    BLK     EQU  3C4EH
3DC2                    LBYTE   EQU  3DC2H
3D9F                    HILO    EQU  3D9FH
386B                    START   EQU  386BH
3C50                    CO      EQU  3C50H
3C3E                    LER     EQU  3C3EH
                        ;
                        ; INITIALIZATION
                        ;
 700                    DUMP:   ORG  7000H      ; ENTRY POINT
7000 0E40                       MVI  C,'@'      ; GET PROMPT CHAR
7002 CD503C                     CALL CO         ; PRINT IT
7005 CD4E3C                     CALL BLK        ; AND A BLANK
7008 0E02                       MVI  C,2        ; SET UP FOR TWO ADDRS
700A CD6B3D                     CALL EXPR       ; GET TWO ADDRESSESS
700D D1                         POP  D          ; ENDING ADDR
700E E1                         POP  H          ; START ADDR
700F 3EF0                       MVI  A,0F0H     ; STRIP RIGHT
7011 A5                         ANA  L          ;     4 BITS
7012 6F                         MOV  L,A        ; FROM START ADDR
7013 0602                       MVI  B,2        ; SKIP TWO LINES
7015 CD9970                     CALL PBOTM      ;  TO GET STARTED
                        ;
                        ; PAGE LOOP
                        ;
7018 E5                 DUMP2:  PUSH  H         ; SAVE DUMP POINTER
7019 0607                       MVI  B,7        ; 7 LINE SKIP
701B CDA270                     CALL LSKIP      ; DO IT
701E 21AA70                     LXI  H,HEAD     ; GET ADDR OF PRINT LINE
7021 0649                       MVI  B,73       ; GONNA PRINT 73 CHARS
7023 4E                 DUMP3:  MOV  C,M        ; GET NEXT CHAR
7024 CD503C                     CALL CO         ; PRINT IT
7027 23                         INX  H          ; BUMP TO NEXT CHAR
7028 05                         DCR  B          ; DECREMENT COUNTER
7029 C22370                     JNZ  DUMP3      ; DO ANOTHER IF NEED BE
 02C E1                         POP  H          ; RESTORE DUMP POINTER
                        ;
                        ; PRINT A PAGE OF MEMORY
```

```
                ;
702D CDC83C             CALL CRLF       ; EMPTY LINE
7030 0639               MVI  B,57       ; 57 LINES TO PAGE BOTTOM
                ;
                ; PRINT A LINE OF MEMORY (16 BYTES)
                ;
7032 CDBA3D     DUMP5:  CALL LADR       ; PRINT THE ADDR
7035 E5                 PUSH H          ; REMEMBER STARTING BYTE
                ;
7036 CD4E3C     DUMP6:  CALL BLK        ; A BLANK SPACE
7039 7E                 MOV  A,M        ; GET NEXT BYTE
703A CDC23D             CALL LBYTE      ;   PRINT AS HEX
703D 23                 INX  H          ; BUMP POINTER
703E 3E0F               MVI  A,0FH      ; CHECK FOR
7040 A5                 ANA  L          ;   MULTIPLE OF 16
7041 C23670             JNZ  DUMP6      ; JUMP NO
7044 CD4E3C             CALL BLK        ; PRINT A BLANK
7047 CD4E3C             CALL BLK        ;   AND ANOTHER
704A 0E2A               MVI  C,'*'      ; GET AN ASTERIK
704C CD503C             CALL CO         ;   PRINT IT
704F E1                 POP  H          ; GET BEGINNING ADDR AGAIN
                ;
                ;  INTERPRET BYTES
                ;
7050 4E         DUMP7:  MOV  C,M        ; GET NEXT BYTE
7051 3E1F               MVI  A,1FH      ; IF BLANK OR HIGHER
7053 B9                 CMP  C
7054 DA5970             JC   DUMP8      ;   THEN OK
7057 0E20               MVI  C,' '      ; ELSE MAKE IT BLANK
7059 3E5C       DUMP8:  MVI  A,'\'      ; IF \ OR LESS
705B B9                 CMP  C
705C D26170             JNC  DUMP9      ; THEN OK
705F 0E20               MVI  C,' '      ; ELSE FORCE A BLANK
7061 CD503C     DUMP9:  CALL CO         ; PRINT THE CHAR
7064 23                 INX  H          ; POINT TO THE NEXT ONE
7065 3E0F               MVI  A,0FH      ; CHECK FOR
7067 A5                 ANA  L          ;   END OF LINE
7068 C25070             JNZ  DUMP7      ; JUMP NOT YET
706B 0E2A               MVI  C,'*'      ; GET AN ASTERIK
706D CD503C             CALL CO         ; PRINT IT
7070 CDC83C             CALL CRLF
7073 05                 DCR  B          ; DECREMENT LINE COUNTER
                ;
                ; CHECK FOR 256 BYTE BREAK
                ;
7074 7D                 MOV  A,L        ; GET LOW BYTE
7075 B7                 ORA  A          ; SET THE FLAGS
7076 C23270             JNZ  DUMP5      ; JUMP NO BREAK
7079 CDC83C             CALL CRLF       ; ELSE A BLANK LINE
707C 05                 DCR  B          ; DECREMENT LINE COUNTER
```

```
707D 2B              DCX  H          ; ADJUST H
707E CD9F3D          CALL HILO       ; CHECK FOR FINISH
7081 D28D70          JNC  DUMPA      ; JUMP NOT YET
7084 CD9970          CALL PBOTM      ; GET TO BOTTOM OF PAGE
7087 CDC83C          CALL CRLF       ; SKIP A LINE
708A C33E3C          JMP  LER        ; RETURN TO MONITOR
                ;
                ; CHECK FOR BOTTOM OF PAGE
                ;
708D 3E13       DUMPA: MVI  A,19     ; NEED 19 MORE LINES
708F 90              SUB  B          ; TO CONTINUE ON THIS PAGE
7090 DA3270          JC   DUMP5      ; JUMP MORE THAN 19
7093 CD9970          CALL PBOTM      ; PUT A BOTTOM ON THE PAGE
7096 C31870          JMP  DUMP2      ; GET NEXT PAGE STARTED
                ;
7099 CDA270     PBOTM: CALL LSKIP    ; PAGE BOTTOM ROUTINE
709C 0E2D            MVI  C,'-'      ; SKIPS TO LAST LINE
709E CD503C          CALL CO         ;  AND PRINTS A DASH
70A1 C9              RET
                ;
70A2 CDC83C     LSKIP: CALL CRLF     ; SKIPS FORWARD
 )A5 05              DCR  B          ; NUMBER OF LINES
/0A6 C2A270          JNZ  LSKIP      ; IN REG B
70A9 C9              RET
                ;
70AA 41444452 HEAD:  DB   'ADDR     1  2  3  4  5  6  '
70AE 20202020
70B2 20312020
70B6 32202033
70BA 20203420
70BE 20352020
70C2 362020
70C5 37202038        DB   '7  8  9  A  B  C  D  E  F   '
70C9 20203920
70CD 20412020
70D1 42202043
70D5 20204420
70D9 20452020
70DD 46202020
70E1 302E2E2E        DB   '0...5...A...F'
70E5 2E352E2E
70E9 2E2E412E
70ED 2E2E2E46
70F1 0D              DB   0DH        ; CARRIAGE RETURN
70F2 0A              DB   0AH        ; LINE FEED
                ;
^900                 END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB45

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 (use additional sheets if necessary)

Program Title

ASCII Display

Function

This routine expects two or three hex parameters. The first and second (16 bits) specify the bounds of a memory area to be displayed on the console device. The contents of these memory locations are interpretted as ASCII characters (MSB ignored).

Required Hardware

Console device

Required Software

9 monitor routines

Input Parameters

Two or three hex numbers from the console device

Output Results

Display on the console device

Registers Modified: A,B,C,D,E,H,L	**Assembler/Compiler Used:** B6700/8080 Cross Assembler V 2.3
RAM Required: 3 bytes	**Programmer:** Gerhard Moertel, S.A. Nilsson
ROM Required: 421 bytes (or RAM)	**Company:** Universitaet Karlsruhe Institut fuer Informatik IV
Maximum Subroutine Nesting Level:	**Address:** Zirkel 2 7500 Karlsruhe 1, GERMANY

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/4040 ☐ 8008 ☒ 8080 ☐ 8048 ☐ 8085 ☐ Other _____ (use additional sheets if necessary)

Program Title

RAM CHECK

Function

Writes alternating 1's and 0's into all RAM locations then reads them and writes the two character result on a model 32 Teletype. Modifiable to ASR33 or LA-36, Decwriter. Can also check ROM read.

Required Hardware

8080A, 4k ROM, any amount of RAM, 8251 USART, selectable baud rate clock.

Required Software

The above program.

Input Parameters

Starting address of RAM (resident in ROM).

Output Results

Hex address and two 6's followed by two L's if address will read and write both 1's and 0's.

Registers Modified: A, B, C, D, E, H, L	Programmer: Scheer
RAM Required: Any amount	Company: Analog Precision, INC.
ROM Required: 4k	Address: 1620 N. Park
Maximum Subroutine Nesting Level: -0-	City: Tucson
Assembler/Compiler Used: MDS V1.0	State: Arizona 85719

Ref.# AA11

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/4040 ☐ 8008 ☒ 8080 ☐ 8048 ☐ 8085 ☐ Other _____ (use additional sheets if necessary)

Program Title

MEMORY TEST PROGRAM/KYBD ENTRY OF START AND END VALUES

Function

Provides extensive bit pattern tests to RAM memory located in any memory space higher than 300H.

Required Hardware

Intellec equipment with kybd and printer

Required Software

MDS-800 Monitor

Input Parameters

Start address as 4 hex digits and end address as 4 hex digits from console.

Output Results

Printout of all error lcoations along with expected and found values.

Registers Modified: ALL	Programmer: Floyd L. Nordin
RAM Required: About 3/4 K bytes	Company: Nordin Enterprises
ROM Required: Monitor Subroutines	Address: P.O. Box 1277
Maximum Subroutine Nesting Level: 12	City: Cupertino
Assembler/Compiler Used: 8080 MDS Macro Assembler V. 1.0	State: CA 95014

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB46

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000

(use additional sheets if necessary)

Program Title

Basic CPU state vector maintenance in a multiprogrammed environment.

Function

These three subroutines can be used by an operating system to maintain CPU state vectors when transferring control from one program to another. The subroutines use RAM locations LPnx to store the x state vectors of program n. Round robin program sequencing is used.

Required Hardware

None

Required Software

None

Input Parameters

None

Output Results

Multiprogrammed state vector maintenance .

Registers Modified: All	**Assembler/Compiler Used:** MicroPac # 80/A Micro Assembler
RAM Required: 30 bytes	**Programmer:** Stanley J. Kaczynski
ROM Required: 83_x bytes	**Company:** Varian/Extrion
Maximum Subroutine Nesting Level: Unlimited	**Address:** P.O. Box 1226/Blackburn Ind. Park Gloucester, MA 01930

```
                        ;REF. NO. AB46
                        ;PROGRAM TITLE BASIC CPU STATE VECTOR MAINTENANCE
                        ;
                        ;
                        ;
0000 E3         LC12:   XTHL
0001 228400             SHLD    LP1P
0004 E1                 POP     H
0005 228600             SHLD    LP1H
0008 EB                 XCHG
0009 228800             SHLD    LP1D
000C C5                 PUSH    B
000D E1                 POP     H
000E 228A00             SHLD    LP1B
0011 F5                 PUSH    PSW
0012 E1                 POP     H
0013 228C00             SHLD    LP1A
0016 2A9600             LHLD    LP2A
0019 E5                 PUSH    H
001A F1                 POP     PSW
 01B 2A9400             LHLD    LP2B
001E E5                 PUSH    H
001F C1                 POP     B
0020 2A9200             LHLD    LP2D
0023 EB                 XCHG
0024 2A8E00             LHLD    LP2P
0027 E5                 PUSH    H
0028 2A9000             LHLD    LP2H
002B C9                 RET
002C E3         LC23:   XTHL
002D 228E00             SHLD    LP2P
0030 E1                 POP     H
0031 229000             SHLD    LP2H
0034 EB                 XCHG
0035 229200             SHLD    LP2D
0038 C5                 PUSH    B
0039 E1                 POP     H
003A 229400             SHLD    LP2B
003D F5                 PUSH    PSW
003E E1                 POP     H
003F 229600             SHLD    LP2A
0042 2AA000             LHLD    LP3A
0045 E5                 PUSH    H
0046 F1                 POP     PSW
0047 2A9E00             LHLD    LP3B
 04A E5                 PUSH    H
 04B C1                 POP     B
004C 2A9C00             LHLD    LP3D
004F EB                 XCHG
```

```
0050 2A9800            LHLD    LP3P
0053 E5                PUSH    H
0054 2A9A00            LHLD    LP3H
0057 C9                RET
0058 E3          LC31: XTHL
0059 229800            SHLD    LP3P
005C E1                POP     H
005D 229A00            SHLD    LP3H
0060 EB                XCHG
0061 229C00            SHLD    LP3D
0064 C5                PUSH    B
0065 E1                POP     H
0066 229E00            SHLD    LP3B
0069 F5                PUSH    PSW
006A E1                POP     H
006B 22A000            SHLD    LP3A
006E 2A8C00            LHLD    LP1A
0071 E5                PUSH    H
0072 F1                POP     PSW
0073 2A8A00            LHLD    LP1B
0076 E5                PUSH    H
 077 C1                POP     B
0078 2A8800            LHLD    LP1D
007B EB                XCHG
007C 2A8400            LHLD    LP1P
007F E5                PUSH    H
0080 2A8600            LHLD    LP1H
0083 C9                RET
0084 001E      LP1P:   DW      17000Q
0086 021E      LP1H:   DW      17002Q
0088 041E      LP1D:   DW      17004Q
008A 061E      LP1B:   DW      17006Q
008C 081E      LP1A:   DW      17010Q
008E 0A1E      LP2P:   DW      17012Q
0090 0C1E      LP2H:   DW      17014Q
0092 0E1E      LP2D:   DW      17016Q
0094 101E      LP2B:   DW      17020Q
0096 121E      LP2A:   DW      17022Q
0098 141E      LP3P:   DW      17024Q
009A 161E      LP3H:   DW      17026Q
009C 181E      LP3D:   DW      17030Q
009E 1A1E      LP3B:   DW      17032Q
00A0 1C1E      LP3A:   DW      17034Q
0000           END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AC14

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 (use additional sheets if necessary)

Program Title

MP8208 A/D Converter Routine

Function

Program functions as an interface between user's PL/M program and the Burr-Brown MP8208 A/D converter on an Intellec 8/MOD 80.

Required Hardware

Burr-Brown MP8208 A/D converter Intellec 8/MOD 80.

Required Software

PL/M 80 cross-compiler

Input Parameters

A/D Channel number

Output Results

Address-type variable containing 12 bit value from selected A/D channel.

Registers Modified: A,B,H,L	**Assembler/Compiler Used:** PL/M 80
RAM Required: 1 byte of RAM used for storage, 2 bytes for stack	**Programmer:** Jeff Stewart
ROM Required: Program occupies 17 bytes of ROM or RAM.	**Company:** Parke, Davis
Maximum Subroutine Nesting Level: 1	**Address:** 2800 Plymouth Road Ann Arbor, Mich. 48106

```
00001  1
00002  1   /*REF. NO. AC14 */
00003  1   /*PROGRAM TITLE MP8208 A/D CONVERTER ROUTINE */
0  04  1        DECLARE ORG LITERALLY '400H';
00005  1        DECLARE ATODASMM LITERALLY '406H';
00006  1   ORG:
00007  1        DECLARE ATODASM DATA (26H,0FFH,79H,87H,6FH,7EH,2CH,46H,0C9H);
00008  1        DECLARE TEMP BYTE, VALUE ADDRESS;
00009  1   ATODREAD: PROCEDURE(NUMBER) ADDRESS;
00010  2        DECLARE NUMBER BYTE;
00011  2        GO TO ATODASMM;
00012  2        END ATODREAD;
00013  1   START:
00014  1        TEMP=INPUT(0);
00015  1        VALUE=ATODREAD(TEMP AND 0FH);
00016  1        IF ((TEMP AND 80H)=0) THEN
00017  1        DO;
00018  1             OUTPUT(0FFH)=LOW(VALUE);
00019  2        END;
00020  1        ELSE
00021  1        DO;
00022  1             OUTPUT(0FFH)=HIGH(VALUE);
00023  2        END;
00024  1        GO TO START;
00025  1   EOF;
NO PROGRAM ERRORS
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AC15

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 (use additional sheets if necessary)

Program Title

"PAGE"

Function

To provide a page break for a Tektronic #4010 graphics terminal.

Required Hardware

The required hardware is an Intel MDS 800 and a Tektronic #4010 CRT

Required Software

The required software is MDS Monitor Ver 1.1. The following changes are needed: Insert a CALL to EE00H in FD59 & FD5B in the Monitor. These instructions are to be replaced by "CD,00,EE,00" IN ORDER

Input Parameters

When the end of a page is reached depressing the letter "E" on the keyboard will erase the page and allow output to continue. Substituting 00 at F000 will disable this routine.

Output Results

Not Applicable

Registers Modified:	**Assembler/Compiler Used:** ISIS 8080 Macro Assy. V1.0
RAM Required: 6 locations	**Programmer:** Steve Weisbrod
ROM Required: 179 locations	**Company:** Medtronic, Inc.
Maximum Subroutine Nesting Level: 2	**Address:** 6120 Earle Brown Dr. Mpls. MN 55430

```
                ;REF. NO. AC15
                ;PROGRAM TITLE PAGE BREAK FOR TEKTRONIX 4010 I/O GRAPHICS TERMIN
                ;
                ;
                ;
                ;           "PAGE"
                ;
                ;THE FUNCTION OF THIS PROGRAM IS TO
                ;PROVIDE A PAGE BREAK FOR A TEKTRONIC #4010
                ;GRAPHICS TERMNAL.
                ;
                ;THE REQUIRED HARDWARE IS A INTEL MDS 800
                ;AND A TEKTRONIC #4010 CRT
                ;
                ;THE REQUIRED SOFTWARE IS MDS MONITOR
                ;VER 1.1 . THE FOLLOWING CHANGES ARE
                ;NEEDED: INSERT A CALL TO EE00H IN
                ;PLACE OF THE TWO INSTRUCTIONS AT
                ;FD59 & FD5B IN THE MONITOR
                ;THESE INSTRUCTIONS ARE TO BE REPLACED
                ;BY "CD,00,EE,00" IN ORDER
                ;
                ;WHEN THE END OF A PAGE IS REACHED DEPRESSING
                ;THE LETTER "E" ON THE KEYBOARD WILL
                ;ERASE THE PAGE AND ALLOW OUTPUT TO
                ;CONTINUE. SUBSTITUTING 00 AT
                ;F000 WILL DISABLE THIS ROUTINE
                ;*****************************************
EE00            ORG 0EE00H
F809            CO EQU 0F809H
F803            CI EQU 0F803H
EE00 F5         PAGE: PUSH PSW
EE01 D5         PUSH D
EE02 E5         PUSH H
EE03 3A02F0      LDA 0F002H ;CHECK TO SEE IF THIS IS THE
EE06 FE53       CPI 53H     ;FIRST TIME THROUGH.
EE08 C21EEE     JNZ SETUP
EE0B 3A03F0     LDA 0F003H
EE0E FE50       CPI 50H
EE10 C21EEE     JNZ SETUP
EE13 3A04F0     LDA 0F004H
EE16 FE7C       CPI 7CH
EE18 C21EEE     JNZ SETUP
EE1B CA35EE     JZ TEST1
EE1E 2102F0     SETUP: LXI H, 0F002H;SETUP FIRST TIME
EE21 3653       MVI M,53H           ;THROUGH.
EE23 2103F0     LXI H,0F003H
EE26 3650       MVI M,50H
EE28 2104F0     LXI H,0F004H
```

```
EE2B 367C     MVI M,7CH
EE2D 2100F0   LXI H,0F000H
EE30 3601     MVI M,01H
EE32 23       INX H
EE33 3600     MVI M,00H
EE35 2100F0   TEST1: LXI H,0F000H;TEST FOR GRAPHICS OR DISABLE
EE38 7E       MOV A, M
EE39 FE01     CPI 01H
EE3B CA42EE   JZ TEST2
EE3E C5       PUSH B
EE3F C37DEE   JMP DONE
EE42 79       TEST2: MOV A,C ;TEST FOR LF
EE43 E67F     ANI 7FH
EE45 FE0A     CPI 0AH
EE47 C5       PUSH B
EE48 C253EE   JNZ INPUT
EE4B 2101F0   LXI H, 0F001H
EE4E 3601     MVI M,01H
EE50 C37DEE   JMP DONE
EE53 2101F0   INPUT: LXI H, 0F001H
EE56 7E       MOV A,M
 E57 FE00     CPI 00H
EE59 CA7DEE   JZ DONE
EE5C 3600     MVI M,00H
EE5E 1E06     MVI E,06H
EE60 1D       LOOP4:DCR E
EE61 CA6DEE   JZ BREAK
EE64 06FF     MVI B, 0FFH
EE66 05       LOOP5: DCR B
EE67 C266EE   JNZ LOOP5
EE6A C360EE   JMP LOOP4
EE6D DBF7     BREAK:IN 0F7H ;ANYTHING ON INPUT PORT?
EE6F E602     ANI 02H
EE71 CA7DEE   JZ DONE
EE74 DBF6     IN 0F6H ;IS IT A BREAK?
EE76 E67F     ANI 7FH
EE78 FE60     CPI 60H
EE7A CC86EE   CZ WAITE
EE7D C1       DONE: POP B
EE7E E1       POP H
EE7F D1       POP D
EE80 F1       POP PSW
EE81 DBF7     IN 0F7H ;INSTR'N FM MON
EE83 E601     ANI 01H
EE85 C9       RET
EE86 CD03F8   WAITE: CALL CI ;"E" DEPRESSED?
 E89 E67F     ANI 7FH
_E8B FE45     CPI 'E'
EE8D C286EE   JNZ WAITE
EE90 C5       ERASE: PUSH B ;SAVE CHAR
```

```
EE91 0E1B       MVI C,1BH ;ERASE CRT
EE93 CD09F8     CALL CO
EE96 0E0C       MVI C,0CH ;RESET CRT
EE98 CD09F8     CALL CO
EE9B 1602       DELAY: MVI D,02H ;TIME DELAY FOR
                ;CRT ERASE RECOVERY
EE9D 7A         LOOP1: MOV A,D
EE9E FE00       CPI 00H
EEA0 CAB3EE     JZ RESTR
EEA3 15         DCR D
EEA4 1EFF       MVI E,0FFH
EEA6 1D         LOOP2: DCR E
EEA7 CA9DEE     JZ LOOP1
EEAA 06FF       MVI B,0FFH
EEAC 05         LOOP3: DCR B
EEAD C2ACEE     JNZ LOOP3
EEB0 C3A6EE     JMP LOOP2
EEB3 C1         RESTR: POP B ;RESTORE
                ;CHAR FOR NEW PAGE
EEB4 C9         RET
0000            END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

intel

Ref. No. AC16

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 (use additional sheets if necessary)

Program Title

CRTBZ

Function

Transmits an ENQ (E^C) from the computer to CRT terminal and waits til an ACK (F^C) is received. Permits hand-shaking between CRT & computer for timing synchronization.

Required Hardware

MDS (Intellec), 2640 CRT terminal (Hewlett Packard) RS232C cable

Required Software

I/O subroutines in Intellec MDS Monitor CI and CO

Input Parameters

None

Output Results

Permits smooth interfacing of 2640 CRT with MDS otherwise MDS dumps information at a faster rate (9600 baud) than the CRT can take (2400 baud maximum), consequently CRT will skip or delete some information.

NOTE: CRTBZ is ordered as one program with GET. Ref. No. AC16 refers to both routines.

Registers Modified: A, C and FLAGS	**Assembler/Compiler Used:** Intellec MDS Assembler Version 1.0
RAM Required: 14 bytes	**Programmer:** Ahmed Muneeruddin
ROM Required: None	**Company:** Management Business Machines
Maximum Subroutine Nesting Level:	**Address:** 5867 Broadway Denver, Colorado 80216

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AC16

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 (use additional sheets if necessary)

Program Title

GET

Function

Read characters from CRT (2640 HP) in block mode strapped for line & format mode on; store the characters at a specified place. The number of unprotected fields to be read is selected by the user (i.e. the length of the block).

Required Hardware

Intellec MDS system, 2640 CRT terminal (HP)
RS232C cable

Required Software

MDS I/O routines: CI,CO
CRT Control Routines: CRTBZ, ENABL, TRIGR, RESET, CRLF,PRINT

Input Parameters

B must contain number of unprotected fields to be read.

H&L should contain starting address in RAM where information can be read.

Output Results

Information read from CRT memory is available in specified area of RAM with end of block identified with a record separator rs. Each unprotected field is separated with a carriage return. At the end of input process, a message appears for the CRT operator to turn off the block mode key on the CRT.

NOTE: GET is ordered as one program with CRTBZ. Ref. No. AC16 refers to both routines.

Registers Modified: A,C,B,H&L, & FLAGS	**Assembler/Compiler Used:** MDS 8 Macro Assembler Version 1.0
RAM Required: 4C bytes + storage for other subroutines	**Programmer:** Ahmed Muneeruddin
ROM Required: NONE	**Company:** Management Business Machines
Maximum Subroutine Nesting Level:	**Address:** 5867 Broadway Denver, Colorado 80216

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB48

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 (use additional sheets if necessary)

Program Title

Disasembler

Function

This program transforms machine codes in memory to a listing of:
1. Addresses
2. Machine codes (1,2 or 3 bytes) depending on the type of instruction.
4. Addresses when applicable (JMP and Call)

Required Hardware

TTY on port 0 and 1

Machine line INTELLEC 8/MOD 8

MONITOR VER 3.0 (on PROMS)

Required Software

Intellec 8 MOD/8 Text Editor VER4.0
Macro Assembler Ver 2.0
Monitor Ver 3.0

Input Parameters

Start of this program:
G037ED
The TTY types DISASEMBLER and expects two HEX parameters of the memory dump to be disasembled, typed on the TTY.

Output Results

Registers Modified: A,B,C,D,E,H,L	**Assembler/Compiler Used:** Intellec 8 Macro Assembler 2.0 Monitor Ver 3-0
RAM Required: 255 Bytes	**Programmer:** Manuel Puigbo
ROM Required: MONITOR VER3-0 1024 BYT	**Company:** Elecma
Maximum Subroutine Nesting Level: 2	**Address:** Ricardo Calvo, 13 Barcelona (6) Spain

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB49

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000

(use additional sheets if necessary)

Program Title

ERLIST

Function

Will search through a file assumed to be all 8080 macro assembler output list file and will copy all lines containing errors to the console.

Required Hardware

MDS-800 and MDS-DOS with 32K RAM

Required Software

ISIS (NOTE: will not run under V1.0)

Input Parameters

File Name
(ERLIST FILENAME.EXT)

Output Results

Lines which contain an error code to the console

Registers Modified: All	Assembler/Compiler Used: ASM80 - V1.1
RAM Required: 32K MDS	Programmer: Barry Yarkoni
ROM Required: --	Company:
Maximum Subroutine Nesting Level: 1	Address: 57 West Timonium Road Timonium, Md. 21093

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. AB50

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000

(use additional sheets if necessary)

Program Title

EXAMIN

Function

Copies "N" lines at a time from a specified file to the MDS console. Hitting the space bar will type the next 10 lines. An "E" will terminate the program at any time.

Required Hardware

MDS-800, MDS-DOS, console, and 32K RAM

Required Software

32K ISIS, any version
MDS monitor, V1.2 or greater

Input Parameters

FILE NAME
EXAMIN FILENAME.EXT (CR): program will request console input

Output Results

Directly to console, 10 lines at a time

Registers Modified:	Assembler/Compiler Used:
All	ASM80 - V1.1
RAM Required:	**Programmer:**
32K MDS	Barry Yarkoni
ROM Required:	**Company:**
--	
Maximum Subroutine Nesting Level:	**Address:** 57 West Timonium Road Timonium, Md. 21093
1	

98-034C

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

intel

Ref. No. AB51

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

Ice-80 Disassembler

Function

A subroutine to be used as a ''MDS call'' in Ice-80 command language.
Translates control block information to assembly statements that are output to selected list device.

Required Hardware

Intellec MDS and Ice-80.

Required Software

Ice-80 software driver & RAM based version 2.Ø or 3.Ø or disk based version 1.Ø.

Input Parameters

Information stored in control block by Ice-80 when returning form emulation mode.

Output Results

Assembly statements are output to selected list device as the instructions are executed by the Ice-80 CPU.

Registers Modified: All	**Assembler/Compiler Used:** MDS Assembler
RAM Required: 1121 bytes	**Programmer:** Ove Andersson
ROM Required:	**Company:** Intel Scandinavia A/S
Maximum Subroutine Nesting Level:	**Address:** Lyngbyvej 32F,II 2100 Copenhagen Ø. Denmark

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Dec. 1976

Ref. AB52

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

DELETE COMMENTS

Function

Reads 8080 source tape from assigned reader (uses monitor RI routine) and deletes all comments. A reduced source tape (instructions only) is punched on the assigned punch device (uses monitor P O routine; can be co-resindent with 8080 assembler, etc.)

Required Hardware

MDS 800 with TTY assigned as punch and reader, is the minimum configuration. High speed punch and reader may be used if available.

Required Software

MDS 800 System Monitor

Input Parameters

Program calls RI (ØF8Ø6M)
Requires character read to be in REG C
Requires carry Bit set if reader has timed out.
(This is as per the monitor routine.)

Output Results

Program calls P O (ØF8ØCH)
Character to be punched is in REG C
At end of tape (reader times out) program returns control to the monitor.

Registers Modified: A,B,C,FLGS	**Assembler/Compiler Used:** 8080 MDS Assembler V.1.Ø
RAM Required: 160 Bytes	**Programmer:** I.E. Powers
ROM Required:	**Company:** Plessey Radar
Maximum Subroutine Nesting Level:	**Address:** Cheapside, Liverpool L2 2EA (England)

```
                        ;REF. NO. AB52
                        ;PROGRAM TITLE DELETE COMMENTS
                        ;
                        ;
                        ;
                        ;
                        ;  DELETE COMMENTS PROGRAM I E POWERS              V 1.1
                        ;
                        ;PROGRAM READS 8080 SOURCE TAPE FROM ASSIGNED READER
                        ;ALL COMMENTS PREFIXED WITH  ;  ARE DELETED AND ALL
                        ;RUBOUT AND NULL CHARACTERS ARE IGNORED
                        ;THE REMAINING CHARACTERS ARE OUTPUT ON THE ASSIGNED
                        ;PUNCH DEVICE
                        ;LINES BEGINNING WITH ;  ARE DELETED COMPLETELEY
                        ;COMMENTS FOLLOWING INSTUCTIONS ARE DELETED WITHOUT
                        ;AFECTING THE FOLLOWING CR LF SEQUENCE
                        ;
                        ;OPERATING PROCEDURE:-
                        ;LOAD ME BY TYPING R0 ON THE CONSOLE DEVICE
                        ;PLACE THE TAPE TO BE PROCESSED IN THE READER AND
                        ;TYPE G3000 ON THE CONSOLE
                        ;AFTER PUNCHING THE REDUCED TAPE I RE-ENTER THE MONITOR
                        ;
                        ;TEST PROCEDURE:-
                        ;ASSEMBLE TEST TAPE TO OBTAIN AN OBJECT TAPE
                        ;USE MY OBJECT TAPE TO PROCESS TEST SOURCE TAPE
                        ;ASSEMBLE THE REDUCED SOURCE TAPE
                        ;BOTH OBJECT TAPES SHOULD CORRESPOND EXACTLY
                        ;
3000                            ORG     3000H
3000 AF                 BEGIN:  XRA     A       ;CLEAR ACC
3001 329E30                     STA     IGLIN   ;CLEAR IGNORE LINE FLAG
3004 329F30                     STA     IGTXT   ;CLEAR IGNORE TEXT FLAG
3007 32A030                     STA     NEWLI   ;CLEAR NEWLINE FLAG
300A CD4B30                     CALL    PUL     ;PUNCH LEADER
300D CD06F8             READ:   CALL    0F806H  ;READ CHAR FROM RI
3010 DA9A30                     JC      EOJ     ;IF READER TIMES OUT - GOTO END OF JOB
3013 0601                       MVI     B,1     ;SET OUTPUT CHARACTER COUNT TO 1
3015 4F                         MOV     C,A     ;MOVE INPUT CHAR TO A
3016 E67F                       ANI     7FH     ;IGNORE PARITY BIT
3018 FE7F                       CPI     7FH     ;CHECK FOR RUBOUT
301A CA0D30                     JZ      READ    ;IGNORE RUBOUT -GET NEXT CHAR
301D FE00                       CPI     0       ;CHECK FOR NULL
301F CA0D30                     JZ      READ    ;IGNORE NULL
3022 FE0D                       CPI     0DH     ;CHECK FOR CR
3024 CA5730                     JZ      CRET    ;CR DETECTED
3027 FE0A                       CPI     0AH     ;CHECK FOR LF
3029 CA6630                     JZ      LINF    ;LF DETECTED
302C FE3B                       CPI     3BH     ;CHECK FOR ;
```

```
302E CA7A30             JZ      SEMCO   ; ; DETECTED
                    ;ASSUME CHAR OTHER THAN CR LF OR ;
3031 AF                 XRA     A
3032 32A030             STA     NEWLI   ;CLEAR NEWLINE FLAG
3035 3A9E30             LDA     IGLIN   ;CHECK IGNORE WHOLE LINE FLAG
3038 FE00               CPI     0
303A C20D30             JNZ     READ    ;FLAG SET -IGNORE CHAR
303D 3A9F30             LDA     IGTXT   ;CHECK IGNORE TEXT FLAG
3040 FE00               CPI     0
3042 C20D30             JNZ     READ    ;FLAG SET -IGNORE CHAR
                    ;NO FLAGS SET -PRINT CHAR ON PUNCH
3045 CD4F30         PRI:    CALL    PO
3048 C30D30             JMP     READ    ;GET NEXT CHAR
                    ;PUNCH LEADER ROUTINE
304B 0664           PUL:    MVI     B,100   ;SET LENGTH OF LEADER COUNT
304D 0E00               MVI     C,0     ;SET A NULL CHAR FOR OUTPUT
                    ;PUNCH OUTPUT ROUTINE
304F CD0CF8         PO:     CALL    0F80CH  ;CALL PUNCH OUTPUT IN MONITOR
3052 05                 DCR     B       ;DECREMENT CHAR COUNT
3053 C24F30             JNZ     PO      ;JUMP BACK IF MORE TO PUNCH
3056 C9                 RET
 057 3A9E30         CRET:   LDA     IGLIN   ;CHECK IGNORE WHOLE LINE FLAG
305A FE00               CPI     0
305C C20D30             JNZ     READ    ;FLAG SET -IGNORE CR
305F AF                 XRA     A       ;FLAG NOT SET -ZERO A
3060 329F30             STA     IGTXT   ;CLEAR THE IGNORE TEXT FLAG
3063 C34530             JMP     PRI     ;OUTPUT THE CHAR
3066 3E01           LINF:   MVI     A,1     ;WE ARE STARTING A NEW LINE
3068 32A030             STA     NEWLI   ;SET NEWLINE FLAG
306B 3A9E30             LDA     IGLIN   ;CHECK IGNORE WHOLE LINE FLAG
306E FE00               CPI     0
3070 CA4530             JZ      PRI     ;FLAG NOT SET -OUTPUT LF
3073 AF                 XRA     A       ;FLAG SET -IGNORE LF
3074 329E30             STA     IGLIN   ;RESET THE FLAG
3077 C30D30             JMP     READ    ;GET NEXT CHAR
307A 3A9E30         SEMCO:  LDA     IGLIN   ;CHECK IGNORE WHOLE LINE FLAG
307D FE00               CPI     0
307F C20D30             JNZ     READ    ;IGNORE COMMENTS WITHIN COMMENTS
3082 3AA030             LDA     NEWLI   ;CHECK NEWLINE FLAG
3085 FE00               CPI     0
3087 CA9230             JZ      MID     ;NOT SET -MID LINE COMMENT
308A 3E01               MVI     A,1     ;FLAG SET WHOLE LINE IS COMMENT
308C 329E30             STA     IGLIN   ;SET IGNORE WHOLE LINE FLAG
308F C30D30             JMP     READ    ;IGNORE ; AND GET NEXT CHAR
3092 3E01           MID:    MVI     A,1     ;HANDLE MID LINE COMMENT
3094 329F30             STA     IGTXT   ;SET IGNORE TEXT ON THIS LINE FLAG
 097 C30D30             JMP     READ    ;GET NEXT CHAR
_09A CD4B30         EOJ:    CALL    PUL     ;END OF JOB -PUNCH TRAILER
309D C7                 RST     0       ;BACK TO MONITOR
309E 00             IGLIN:  DB      0       ;IGNORE WHOLE LINE FLAG
```

```
309F 00        IGTXT:  DB      0       ;IGNORE TEXT ON THIS LINE FLAG
30A0 00        NEWLI:  DB      0       ;NEW LINE FLAG
0000                   END
```

Ref. # AB53

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☐ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

TYPE

Function

Types a source file with the correct number of spaces inserted for each tab. Display can be "frozen" by typing any keyboard character, and unfrozen by a similar action. The program is terminated when a C is typed.

Required Hardware

MDS with Diskette

Required Software

ISIS-I

Input Parameters

File Name

Output Results

File Listing

Registers Modified:	Programmer: Norman H. Azadian
RAM Required:	Company: System Development Corp.
ROM Required:	Address: 2500 Colorado Avenue
Maximum Subroutine Nesting Level:	City: Santa Monica
Assembler/Compiler Used: Intellec 8080 Macro Assembler V1.0	State: California 90406

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Dec. 1976

Ref. AB54

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 (use additional sheets if necessary)

Program Title

SDK-80 keyboard monitor

Function

basic operating system for the SDK-80 microcomputer kit. allows loading program and execution of user programs as well as breakpoint and register examination and modification

Required Hardware

1. Intel SDK-80 microcomputer with I/O port installed at locations 0F4h to 0F7h.
2. Keyboard and keyboard control circuit per schematic

Required Software

none additional

Input Parameters

keystrokes

Output Results

implementation of functions requested

Registers Modified: all	**Assembler/Compiler Used:** MDS Isis-ASM80 V1.0
RAM Required: 19 decimal locations	**Programmer:** J.F. Jankura
ROM Required: 24A hexadecimal locations	**Company:** General Electric
Maximum Subroutine Nesting Level: 5 levels	**Address:** Nela Park, Cleveland, Oh.

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Dec. 1976
Ref. AB55

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 (use additional sheets if necessary)

Program Title

Disc Dump Routine for ICOM FDOS-II/MOD 80 floppy disc operating system.

Function

This routine provides a formatted dump of any portion of the floppy disc, in decoded ASCII, hexadecimal, or both. The user specifies the number of sectors to be dumped and the desired format.

Required Hardware

Intellec 8/MOD 80, ICOM floppy disc drive.

Required Software

Intellec monitor, ICOM FODS-11/mod 80 resident module.

Input Parameters

Unit no., first sector and track to be dumped, no. of sectors to be dumped, and format (alpha, hex, or both).

Output Results

Formatted dump of disc. DD-marked sectors can not be dumped, so a message is printed to this effect. At the end of the dump, the user may rerun the program, or go to either the Intellec or FDOS monitor.

Registers Modified:	**Assembler/Compiler Used:** PL/M vers 3.2
RAM Required: 700 Bytes hex	**Programmer:** L.P.M. Payzant
ROM Required:	**Company:** Nova Scotia Technical College
Maximum Subroutine Nesting Level:	**Address:** P. O. Box 1000, Halifax, Nova Scotia, Canada.

98-034C

```
/* REF. NO. AB55 */
/* PROGRAM TITLE DUMP ROUTINE (PL/M SOURCE) */


100H: /*     D I S C   D U M P   P R O G R A M        */
/*          P. PAYZANT, N. S. T. C., AUGUST, 1976            */
DECLARE (UNIT, TRACK, SECTOR, CODE, PRINT, J, K, CHAR) BYTE;
DECLARE (ACTTRACK, ACTSECT, EXPSECT, EXPTRACK) BYTE;
DECLARE (NSECT, I) ADDRESS;
DECLARE BUF (128) BYTE;
DECLARE TRUE LITERALLY '0FFH', FALSE LITERALLY '0';

CIC: PROCEDURE BYTE;     /*   READ A CHARACTER  */
     GO TO 3803H;
     END CIC;

COC:      PROCEDURE (CHAR);  /* PRINT A CHARACTER */
          DECLARE CHAR BYTE;
          GO TO 3809H;
          END COC;

SPACE:    PROCEDURE(NSPACES);    /* PRINT N SPACES */
          DECLARE (N, NSPACES) BYTE;
          DO N=1 TO NSPACES;
           CALL COC(' ');
           END;
          END SPACE;

RDISC:    PROCEDURE BYTE;  /* READ A BYTE FROM THE DISC  */
          GO TO 3105H;
          END RDISC;

PRINTN:   PROCEDURE(NUMBER);    /* PRINT A BYTE IN HEX  */
          DECLARE (NUMBER, CHAR) BYTE;
          CHAR=SHR(NUMBER, 4);
          IF CHAR>=10 THEN CHAR=CHAR+'A'-10;
                      ELSE CHAR=CHAR+'0';
          CALL COC(CHAR);
          CHAR=NUMBER AND 0FH;
          IF CHAR >= 10 THEN CHAR=CHAR+'A'-10;
                        ELSE CHAR=CHAR+'0';
          CALL COC(CHAR);
          END PRINTN;

PRINTA:   PROCEDURE(STRINGADDR, CRLFFLAG);  /* PRINT A STRING */
          DECLARE STRINGADDR ADDRESS, (J, CRLFFLAG) BYTE;
          DECLARE (STRING BASED STRINGADDR) (70) BYTE;
          IF CRLFFLAG=1 THEN DO;
           CALL COC(0DH); /* CR */
           CALL COC(0AH); /* LF */
           END;
          J=0;
          DO WHILE STRING(J)<> '$';
           CALL COC(STRING(J));
           J=J+1;
           END;
          END PRINTA;

READHEX:  PROCEDURE ADDRESS;    /* READ A HEX NUMBER TYPED AT THE CONSOLE */
          DECLARE VAL ADDRESS, CHAR BYTE;
          VAL=0;
          DO WHILE TRUE;
```

```
          CHAR=CIC;
          CALL COC(CHAR);
          IF (CHAR AND 7FH)=0DH THEN DO;
           CALL COC(0AH);
           RETURN VAL;
           END;
          CHAR=(CHAR AND 7FH)-'0';
          IF CHAR>=0 AND CHAR <=9 THEN DO;
           VAL=(16*VAL)+CHAR;
           END;
          ELSE IF CHAR>=11H AND CHAR <=16H THEN DO;
           VAL=(16*VAL)+CHAR-7;
           END;
          ELSE CALL COC(07H); /* RING BELL FOR NON-HEX CHARACTER */
          END;
         END READHEX;

DECLARE MSG1 DATA ('DISC DUMP PROGRAM$'),MSG2 DATA ('UNIT NO.? (0-3) $'),
         MSG3 DATA ('STARTING TRACK? (00-4C) $'),MSG4 DATA('STARTING SECTOR? (1-
1A) $'), MSG5 DATA ('NO OF SECTORS? (1-7D2) $'), MSG6 DATA ('ALPHA(1), HEX(2), O
R BOTH(3)? $');
DECLARE MSG7 DATA('TRACK $'),MSG8 DATA (', SECTOR $'),MSG9 DATA ('BYTE$'),
        MSG10 DATA ('$'),MSG11 DATA ('ENTER A (AGAIN), D (DOS), OR M (MONITOR) $
'), MSG12 DATA (' *THIS SECTOR DD MARKED*$');

/* DIALOGUE WITH USER */

          CALL PRINTA(.MSG1,1);
P1%       CALL PRINTA(.MSG2,1);
          IF (UNIT%=LOW(READHEX))<0 OR UNIT >3 THEN GO TO P1;
P2%       CALL PRINTA(.MSG3,1);
          IF(TRACK%=LOW(READHEX))<0 OR TRACK>4CH THEN GO TO P2;
P3%       CALL PRINTA(.MSG4,1);
          IF(SECTOR%=LOW(READHEX))<1 OR SECTOR>1AH THEN GO TO P3;
P4%       CALL PRINTA(.MSG5,1);
          IF(NSECT%=READHEX)<1 OR NSECT>7D2H THEN GO TO P4;
P5%       CALL PRINTA(.MSG6,1);
          IF(CODE%=LOW(READHEX))<1 OR CODE>3 THEN GO TO P5;

/* INITIALIZE FILE POINTER TABLE */

DECLARE TABLELOC ADDRESS, (TABLE BASED TABLELOC) (5) BYTE;
          TABLELOC=5H;  /*  ADDRESS OF INPUT FILE TABLE  */
          NSECT=NSECT+1;
          TABLE(0)=LOW(NSECT);
          TABLE(1)=HIGH(NSECT);
          TABLE(2)=TRACK;
          TABLE(3)=(SECTOR-1) OR ROL(UNIT,6);
          TABLE(4)=0;

/*  MAIN DUMP LOOP   */

          EXPTRACK=TRACK;
          EXPSECT=SECTOR;
          NSECT=NSECT-1;
          DO I=1 TO NSECT;

/* READ 1 SECTOR (128 BYTES) INTO RAM.   */

           DO J=0 TO 127;
            BUF(J)=RDISC;
            END;
```

```
/* GET  SECTOR AND TRACK JUST READ FROM FILE POINTER TABLE, AND CHECK
THAT IT MATCHES THE EXPECTED TRACK AND SECTOR.  IF NOT, ISSUE DD MESSAGE, INCRE-
MENT EXPECTED SECTOR, AND CHECK AGAIN UNTIL EXPECTED AND ACTUAL MATCH.  */

            CALL PRINTA(.MSG7,1);
            CALL PRINTN(EXPTRACK);
            CALL PRINTA(.MSG8,0);
            CALL PRINTN(EXPSECT);
            ACTSECT=TABLE(3)AND 1FH;
            ACTTRACK=TABLE(2);
            DO WHILE ACTSECT<>EXPSECT OR ACTTRACK<>EXPTRACK;
             CALL PRINTA(.MSG12,0);
             EXPSECT=EXPSECT+1;
             IF EXPSECT=1BH THEN DO;
              EXPSECT=1;
              EXPTRACK=EXPTRACK+1;
              END;
             CALL PRINTA(.MSG7,1);
             CALL PRINTN(EXPTRACK);
             CALL PRINTA(.MSG8,0);
             CALL PRINTN(EXPSECT);
             END;
            EXPSECT=EXPSECT+1;
           IF EXPSECT=1BH THEN DO;
             EXPSECT=1;
             EXPTRACK=EXPTRACK+1;
             END;
            CALL PRINTA(.MSG9,1);

/* PRINT THE SECTOR IN LINES OF 10H BYTES  */

           DO J=0 TO 70H BY 10H;
            CALL PRINTA(.MSG10,1);
            CALL SPACE(1);
            CALL PRINTN(J);
            CALL SPACE(1);
            IF(CODE AND 01H)=1 THEN DO;  /* ALPHA DUMP */
             PRINT=FALSE;
             DO K=0 TO 15;  /* SCAN FOR PRINTABLE CHARACTERS  */
              CHAR=BUF(J+K);
              IF CHAR>20H AND CHAR<7EH THEN PRINT=TRUE;
              END;
             IF PRINT THEN DO K=0 TO 15;
              CALL SPACE(2);
              CHAR=BUF(J+K) AND 7FH;
              IF CHAR<20H OR CHAR>7EH THEN CHAR=' ';
              CALL COC(CHAR);
              END;
             END;
            IF(CODE AND 02H)=2 THEN DO;  /* HEX DUMP  */
             IF (CODE AND 01H)=1 THEN DO;
              CALL PRINTA(.MSG10,1);
              CALL SPACE(4);
              END;
             DO K=0 TO 15;
              CALL SPACE(1);
              CALL PRINTN(BUF(J+K));
              END;
             END;
           END;
```

```
          END;

P6%       CALL PRINTA(.MSG11,1);
          CALL COC(CHAR%=CIC);
          CHAR=CHAR AND 7FH;
          IF CHAR='A' THEN GO TO P1;
          ELSE IF CHAR='M' THEN GO TO 3800H;
          ELSE IF CHAR='D' THEN GO TO 3000H;
          ELSE GO TO P6;
EOF
 COPY COMPLETE.
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Dec. 1976
Ref. AB56

☐ 4004 ☐ 4040 ☐ 8008 ☐ 8080 ☐ 3000

(use additional sheets if necessary)

Program Title

LIST/PRINT/TYPE "LIST, SRC" on Diskette.

Function

LIST a file on the line printer allowing space for tabs.

Required Hardware

Intellec MDS and appropriate output device.

Required Software

ISIS diskette Operating Systems.

Input Parameters

Change output file name (symbol "LFILE") to
:TO: for TTY (PRINT PROG)
:CO: for CRT/TTY (LIST PROG)
:VO: for CRT (LIST PROG)
:LP: for Line Printer (LIST PROG)

Output Results

Tab characters will be acknowledged.

Registers Modified:	Assembler/Compiler Used: ASM80
RAM Required: 32K	Programmer: Brian L. Halla
ROM Required:	Company: Intel Corp.
Maximum Subroutine Nesting Level:	Address: 3065 Bowers Ave. Santa Clara, Ca. 95051

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Dec. 1976

Ref. AC18

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

Interfacing the MDS and HP2644A

Function

Acts as an interface between MDS and HP2644A mini data station. Lets user use tape for both reading and writing data.

Required Hardware

MDS system, HP2644A mini data station optional floppy disc system.

Required Software

Input Parameters

For writing data of tape: C Reg should contain ASCII character.
For reading data from tape: A Reg contains tha ASCII character.
For using the console as a list device: A Reg should contain ASCII character.

Output Results

When reading data from tape, the program returns the ASCII character in the 'A' registers. When it detects end of file, it returns two null characters. When writing data to tape, the program writes out the character in the 'C' register to tape.

Registers Modified: A, PSW	**Assembler/Compiler Used:** ASM80 Assembler
RAM Required: 240H Bytes	**Programmer:** A. Aggarwal
ROM Required: 0	**Company:** BNR, Inc.
Maximum Subroutine Nesting Level: 2	**Address:** 3174 Porter Dr., Palo Alto, CA.

98-034B

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Dec. 1976
Ref. AC19

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 (use additional sheets if necessary)

Program Title

Video Driver

Function

To drive a video system of 16 lines by 64 characters per line.

Required Hardware

1K video ram, locations OFC00H to OFFFFH, as part of 8080 bus. Each memory location represents 1 character. Each successive block of 64 locations represents one line.

Required Software

None

Input Parameters

A register has 7 bit ASCII character.

FF (form feed) character sets all of 1K block to spaces (020H) and sets the character pointer to the start of the memory.

LF (line feed) character adds 64 modulo 1024 to the character pointer and then blanks line.

CR (carriage return) character sets memory locations from the character pointer to the end of the current line to blanks and sets the character pointer to the beginning of the current line.

Output Results

Registers Modified: None	**Assembler/Compiler Used:** Intellec 8 Mod 80 Ver 3.0
RAM Required: 1K Video RAM at OFC00H 2 bytes for character pointer	**Programmer:** S. Graf
ROM Required: 96 bytes	**Company:** B.C. Telephone Company
Maximum Subroutine Nesting Level: 1	**Address:** 768 Seymour STR. (4th WCT) Vancouver, B.C., Canada V6B 3K9

```
                        ;
                        ;REF. NO. AC19
                        ;PROGRAM TITLE VIDEO DRIVER
                        ;
                        ;
                        ;
                        ;
                        ;

                        ;       RAM SPACE
                        ;
0000 00FC               VIDCP:  DW      0FC00H  ;2 BYTES REQUIRED
                        ;
                        ;
                        ;       EQUATES
                        ;
000C                    FF      EQU     0CH     ;FORM FEED
000A                    LF      EQU     0AH     ;LINE FEED
000D                    CR      EQU     0DH     ;CARRIAGE RETURN
0020                    SPACE   EQU     020H    ;SPACE
                        ;
                        ;
                        ;
                        ;       PROGRAM SPACE
                        ;
                        ;
                        ;       ENTRY POINT
                        ;
                        WRITE:
0002 F5                         PUSH    PSW     ;SAVE
0003 E5                         PUSH    H
0004 2A0000                     LHLD    VIDCP   ;GET CHARACTER POINTER
0007 FE0C                       CPI     FF      ;CHECK FOR FORM FEED
0009 CA1E00                     JZ      VIDFF   ;JUMP IF FORM FEED
000C FE0D                       CPI     CR      ;CHECK FOR CARRIAGE RETURN
000E CA2F00                     JZ      VIDCR   ;JUMP IF CARRIAGE RETURN
0011 FE0A                       CPI     LF      ;CHECK FOR LINE FEED
0013 CA3B00                     JZ      VIDLF   ;JUMP IF LINE FEED
0016 77                         MOV     M,A     ;PUT CHARACTER IN VIDEO MEMORY
0017 23                         INX     H       ;BUMP POINTER
                        ;
                        ;
                        ;       COMMON EXIT POINT
                        ;
                        VIDRT:
0018 220000                     SHLD    VIDCP   ;SAVE CHARACTER POINTER
001B E1                         POP     H       ;RESTORE
001C F1                         POP     PSW     ;RESTORE
001D C9                         RET             ;EXIT
                        ;
```

```
                        ;
                        ;       FORM FEED SERVICE
                        ;
                        VIDFF:
001E 2100FC                     LXI     H,0FC00H;SET STARTOF VIDEO MEMORY
0021 220000                     SHLD    VIDCP   ;SET NEW START POSITION
                        VIDFC:
0024 3620                       MVI     M,SPACE ;PUT SPACE IN VIDEO MEMORY
0026 23                         INX     H       ;BUMP POINTER
0027 AF                         XRA     A       ;SET A TO 0
0028 BC                         CMP     H       ;CHECK IF H IS 0
0029 C22400                     JNZ     VIDFC   ;JUMP IF NOT
002C E1                         POP     H       ;DONE - RESTORE
002D F1                         POP     PSW     ;RESTORE
002E C9                         RET             ;EXIT
                        ;
                        ;
                        ;       CARRIAGE RETURN SERVICE
                        ;
                        VIDCR:
002F E5                         PUSH    H       ;SAVE PRESENT CARRACTER POSITION
0030 CD5800                     CALL    VBLNK   ;BLANK FILL LINE
0033 E1                         POP     H       ;RETORE CHARACTER POSITION
0034 7D                         MOV     A,L     ;GET LOW BYTE CHARACTER POINTER
0035 E6C0                       ANI     0C0H    ;SET BACK TO START OF LINE
0037 6F                         MOV     L,A     ;PUT BACK
0038 C31800                     JMP     VIDRT   ;USE COMMON EXIT
                        ;
                        ;
                        ;       LINE FEED SERVICE
                        ;
                        VIDLF:
003B 7D                         MOV     A,L     ;GET LOW BYTE CHARACTER POSITION
003C C640                       ADI     040H    ;INCREMENT LINE COUNT
003E 6F                         MOV     L,A     ;PUT BACK
003F 7C                         MOV     A,H     ;GET HIGH BYTE
0040 CE00                       ACI     0       ;INCREMENT IF NECESSARY
0042 67                         MOV     H,A     ;PUT BACK
0043 FA4C00                     JM      VLFBL   ;BLANK LINE IF NO WRAP AROUND
0046 26FC                       MVI     H,0FCH  ;SET TO START IF OVER END
0048 7D                         MOV     A,L     ;GET LOW ORDER LINE BITS
0049 E6C0                       ANI     0C0H    ;CLEAR THEM
004B 6F                         MOV     L,A     ;PUT BACK
                        VLFBL:
004C E5                         PUSH    H       ;SAVE CHARACTER POSITION
004D 7D                         MOV     A,L     ;GET POSITION IN LINE
004E E6C0                       ANI     0C0H    ;SET TO START
0050 6F                         MOV     L,A
0051 CD5800                     CALL    VBLNK   ;BLANK LINE
0054 E1                         POP     H       ;RESTORE CHARACTER POSITION
```

```
0055 C31800                 JMP     VIDRT   ;USE COMMON EXIT
                            ;
                            ;
                            ;       BLANK TO END OF LINE
                            ;
                            VBLNK:
0058 3620                   MVI     M,SPACE ;PUT IN SPACE
005A 2C                     INR     L       ;BUMP COUNTER
005B 7D                     MOV     A,L     ;GET POSITION IN LINE
005C E63F                   ANI     03FH    ;ONLY
005E C25800                 JNZ     VBLNK   ;JUMP IF NOT AT END OF LINE
0061 C9                     RET             ;DONE
                            ;
                            ;
                            ;
0000                        END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

intel®

Dec. 1976

Ref. AC20

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

BASIC DIGITAL PANEL METER CALL

Function

Reads BCD data from a 4 digit plus sign panel meter and sets a BASIC variable X equal to the data. Meter initiation is done in hardware.

Required Hardware

8Ø8Ø system with panel meter I/O interface.

Required Software

8Ø8Ø BASIC, floating point package, and octal debugging program.

Input Parameters

BASIC call statement protocol:

CALL (1,X)

X Should Be Previously Defined.

Output Results

The variable X equals the meter reading integer value.

I/O port numbers: 6 reads MS 8 bits, 7 reads LS 8 bits, 1 reads sign bit: D_7.

Registers Modified: A, B, C, D, E, H, L	**Assembler/Compiler Used:** 8080 macro assem. ver. 2.2
RAM Required:	**Programmer:** C. L. Pomernacki
ROM Required: 103_{10} locations	**Company:** LLL
Maximum Subroutine Nesting Level:	**Address:** Livermore, California 94550

```
                        ;REF. NO. AC20
                        ;PROGRAM TITLE BASIC DIGITAL PANEL METER CALL
                        ;
                        ;
                        ;
                        ; CALL ROUTINES

0000 30AA               FWAM:   DW 0AA30H               ;DEFINE FWAM POINTER

                        ;ENTRIES TO SUBTABLE

0002 01                 SUBS:   DB 1
0003 0600                       DW BCDBIN

0005 FF                         DB 377Q                 ;END OF TABLE

                        ; SUB #1

0006 DB06               BCDBIN: IN 6                    ;MS8 TO A
0008 67                         MOV H,A                 ;A TO H
0009 DB07                       IN 7                    ;LS8 TO A
000B 6F                         MOV L,A                 ;A TO L
000C 1E00                       MVI E,0                 ;CLEAR E REG.
000E 7D                         MOV A,L                 ;L TO A
000F CD2500                     CALL MSKMS
0012 D5                         PUSH D                  ;D,E TO STACK
0013 7D                         MOV A,L                 ;L TO A AGAIN
0014 CD2900                     CALL MSKLS
0017 D5                         PUSH D                  ;D,E TO STACK
0018 7C                         MOV A,H                 ;H TO A
0019 CD2500                     CALL MSKMS
001C D5                         PUSH D                  ;D,E TO STACK
001D 7C                         MOV A,H                 ;H TO A AGAIN
001E CD2900                     CALL MSKLS
0021 D5                         PUSH D                  ;D,E TO STACK
0022 C33100                     JMP BBIN

0025 E60F               MSKMS:  ANI 17Q                 ;MASKOUT MS4
0027 57                         MOV D,A                 ;A TO D
0028 C9                         RET

0029 E6F0               MSKLS:  ANI 360Q                ;MASKOUT LS4
002B 0F                         RRC                     ;SHIFT
002C 0F                         RRC
002D 0F                         RRC
002E 0F                         RRC                     ;4 RIGHT
002F 57                         MOV D,A                 ;A TO D
0030 C9                         RET
```

```
0031 0E04       BBIN:   MVI C,4             ;4 IS # OF BCD DIGITS
0033 210000             LXI H,0             ;INITIALIZE H,L
0036 AF                 XRA A               ;AND A TO ZERO

                ;MULT EACH DIGIT BY 10

0037 E5         LOOP:   PUSH H              ;SAVE LS 16 BITS ON STK.
0038 47                 MOV B,A             ;SAVE A IN B REG.
0039 29                 DAD H               ;MULT
003A 8F                 ADC A
003B 29                 DAD H
003C 8F                 ADC A               ;BY 4
003D D1                 POP D               ;RESTORE 16 BITS TO D,E
003E 19                 DAD D               ;ADD 1
003F 88                 ADC B               ;TO MAKE 5
0040 29                 DAD H               ;MULT BY 2
0041 8F                 ADC A               ;TO MAKE 10
0042 47                 MOV B,A             ;SAVE A IN B
0043 D1                 POP D               ;GET NEXT DIGIT
0044 7A                 MOV A,D             ;INTO A REG.
0045 85                 ADD L               ;ADD IN NEW DIGIT
0046 6F                 MOV L,A
0047 7C                 MOV A,H
0048 CE00               ACI 0               ;ADD IN CARY
004A 67                 MOV H,A
004B 78                 MOV A,B
004C CE00               ACI 0               ;ADD IN CARRY
004E 0D                 DCR C               ;BUMP DOWN COUNTER
004F C23700             JNZ LOOP            ;DONE%

                ;NOTE; A,H,L CONTAIN FIXED PT. #
                ;DIGITS ARE RIGHT JUSTIFIED MS IN TOP OF STACK

0052 11307C             LXI D,7C30H         ;GET REG. ADD.
0055 EB                 XCHG                ;D,E NAS # H,L HAS ADD.
0056 77                 MOV M,A             ;MS TO FREG1
0057 23                 INX H
0058 72                 MOV M,D             ;S TO FREG1+1
0059 23                 INX H
005A 73                 MOV M,E             ;LS TO FREG1+2
005B 23                 INX H
005C DB01               IN 1                ;SIGN TO A
005E 77                 MOV M,A             ;SIGN TO FREG1+3
005F 2B                 DCX H
0060 2B                 DCX H
0061 2B                 DCX H               ;RESET POINTER
0062 CD0FCA             CALL 0CA0FH          ;FLOAT #
0065 D1                 POP D               ;GET VAR. ADD.
0066 EB                 XCHG                ;H,L HAS VAR. ADD. ; D,E HAS FREG1
```

```
0067 CD17B6             CALL 0B617H          ;SET VAR. =FREG1
006A C9                 RET                  ;DONE
0000                    END
```

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Dec. 1976
Ref. AC21

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 (use additional sheets if necessary)

Program Title

APL Graphic Display on a 5 x 7 Dot Matrix

Function

The program decodes a standard APL keyboard and outputs the pressed key to a 5 x 7 dot matrix display.

Required Hardware

Required Software

None

Input Parameters

Input port 80 is 7-bit code from an APL keyboard

Output Results

Output ports 0 through 4 drive the five columns on a MAN 2A dot matrix display. I/O W latches the output data, which is input into the seven rows of the MAN 2A dot matrix display.

Program offered on diskette only.

Registers Modified: A, B, C, H, and L	**Assembler/Compiler Used:** 8080 Assembler, Ver. 2.4
RAM Required: 1 byte	**Programmer:** Helene Young Myers
ROM Required: 610 bytes	**Company:**
Maximum Subroutine Nesting Level: 1	**Address:** 4313 Judith Street Rockville, Maryland 20853

98-0348

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Dec. 1976
Ref. AC22

☐ 4004 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

$BLPT

Function

Buffered line printer driver for the Centronix 101A Line Printer, utilizing the Intellec 8-Mod 80

Required Hardware

Intellec 8 Mod 80

Centronix 101A Line Printer

Required Software

Intellec 8 Mod 80 Exec

Input Parameters

```
AL = 1
...
(CHAR IN C)
CALL LO
```

Output Results

The assembled line is printed on the line printer when a carriage return character is received.

Registers Modified: none	Maximum Subroutine Nesting Level: 1
RAM Required: 86 bytes (including CALL LO)	Assembler/Compiler Used: Intellec 8 Macro Assembler
ROM Required: 256 bytes	Programmer: George Woodley
	Company: Woodley Associates 604 Indian Home Road Danville, CA 94526

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Dec. 1976

Ref. AE1

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

LIST

Function

THIS ROUTINE IS TO BE USED AS AN MDSCALL IN CONJUNCTION WITH THE ICE80 SOFTWARE DRIVER. WHEN CALLED, IT WILL RETRIEVE THE PC FROM THE ICE80 CONTROL BLOCK, MATCH IT WITH AN ADDRESS IN THE USER'S LIST FILE GENERATED BY ASM80, AND PRINT THE CORRESPONDING SOURCE CODE ON THE CONSOLE. IN THE STEP MODE, THE ENTIRE PROGRAM MAY BE LISTED.

Required Hardware

MDS-800 MDS-DOS

ICE-80 CONSOLE

Required Software

ISIS MDS MONITOR V1.2

ICE-80 ASM80

Input Parameters

THE PROGRAM WILL ASK FOR THE USER'S LIST FILE FILENAME ON THE FIRST SUBROUTINE CALL.

Output Results

A LINE OF SOURCE CODE, STARTING WITH THE LABEL FIELD AND EXTENDING THROUGH THE COMMENT FIELD, IS LISTED ON THE CONSOLE. ISIS ERRORS CAUSE AN ERROR MESSAGE TO BE DISPLAYED AND HALT THE PROCESSOR. A LARGE BLOCK OF COMMENTS OR PSEUDO-OPS IN THE MIDDLE OF THE LIST FILE MAY CAUSE THE ROUTINE TO HANG UP IN A LOOP.

Registers Modified: ALL	**Assembler/Compiler Used:** ISIS 8080 V1.1
RAM Required: 435 BYTES	**Programmer:** BERNARD J. VERREAU
ROM Required:	**Company:** NATIONAL CASH REGISTER
Maximum Subroutine Nesting Level: 3	**Address:** P.O. BOX 607, MILLSBORO DEL. 19963

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Dec. 1976
Ref. AE2

☐ 4004 ☐ 4040 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program Title

Trace Routine

Function

This routine is a debugging aid, which when appended to a program allows the user to run the program normally, or to single step the program. In single step mode, he can display breakpoint, jump to the monitor, ISIS, or proceed, by typing in a character from the system console. The list of commands is easily expanded.

Required Hardware

MDS

Required Software

MDS System Monitor V 2.0
MAC 80 for assembly

Input Parameters

"DEBUG" is one byte which is set to one if the user wishes to run his program in trace mode; otherwise set to zero.

"SINGLESTEP" is one byte which, when not on will merely print user's message on the console device. If set to one, the program will single step and wait for the user to type a character on the console device, execute the corresponding command before proceeding.

The user must also supply the text message desired.

Output Results

The program will print out the desired text, singlestep if desired, and respond to user commands by displaying the breakpoint, or jumping to the system monitor, or jumping to ISIS, or simply proceeding. The possible commands can be expanded.

Registers Modified: None - all registers restored	**Assembler/Compiler Used:** MAC 80
RAM Required: 115 bytes	**Programmer:** Ed Klingman
ROM Required: None	**Company:** Cybernetic Micro Systems
Maximum Subroutine Nesting Level: Three (counting monitor calls)	**Address:** 2460 Embarcadero Way Palo Alto, CA. 94303

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Dec. 1976
Ref. AE3

☐ 4004 ☐ 4040 ☐ 8008 ☑ 8080

(use additional sheets if necessary)

Program Title

8080 Symbol Table Dump

Function

This program prints the name and location of all symbols (including macro's), after the first or second pass of the assembler. The program can be assembled in PROM if desired.

Required Hardware

Intellec 8/MOD 80 and ASR 33 TTY

Required Software

Monitor V 3.Ø
Macro Assembler V 4.1

Input Parameters

None

Output Results

Test tape included to verify macro capability. This program is based on the (8008) "Symbol Table List Routine" (Ref. AB30) by Robert Uleski.

Registers Modified: All	**Assembler/Compiler Used:** Macro Assembler V 4.1
RAM Required: 161 bytes	**Programmer:** Earl G. Day
ROM Required: N/A	**Company:** DIT-MCO International Corp.
Maximum Subroutine Nesting Level: N/A	**Address:** 5612 Brighton Terr Kansas City, MO. 64130

```
                        ;REF NO.  AE3
                        ;PROGRAM TITLE 8080 SYMBOL TABLE DUMP
                        ;
                        ;
                        ;
2A00                            ORG     2A00H
3CCD                    CRLF    EQU     3CCDH
3C5D                    TTY     EQU     3C5DH
1C8E                    SYMBL   EQU     1C8EH
386B                    START   EQU     386BH
3CC5                    CONV    EQU     3CC5H
2A00 CDCD3C                     CALL    CRLF
2A03 0616                       MVI     B,22
2A05 CD832A                     CALL    BLKO    ;PRINT THE NUMBER OF BLANKS
                        ;    IN REGISTER B
2A08 21732A                     LXI     H,MSG   ;PRINT "SYMBOL TABLE"
2A0B 060E                       MVI     B,14
2A0D 4E                 SYM0:   MOV     C,M
2A0E 23                         INX     H
2A0F CD5D3C                     CALL    TTY
 A12 05                         DCR     B
2A13 C20D2A                     JNZ     SYM0
2A16 118E1C                     LXI     D,SYMBL ;POINT TO BEGINNING OF
                        ;    SYMBOL TABLE
2A19 CD2B2A             LOOP:   CALL    LOOP0   ;PRINT FOUR SYMBOLS PER LINE
2A1C CD2B2A                     CALL    LOOP0
2A1F CD2B2A                     CALL    LOOP0
2A22 CD2B2A                     CALL    LOOP0
2A25 CDCD3C                     CALL    CRLF
2A28 C3192A                     JMP     LOOP
2A2B 62                 LOOP0:  MOV     H,D     ;PRINT A SYMBOL AND ADDRESS SR
2A2C 6B                         MOV     L,E
2A2D 7E                         MOV     A,M
2A2E 0608                       MVI     B,8     ;SET TRIES COUNTER.
2A30 CD8D2A                     CALL    ALFA1   ;FIND ALPHA CHAR.
2A33 6B                         MOV     L,E
2A34 0605                       MVI     B,5
2A36 4E                 LOOP1:  MOV     C,M     ;LOAD & PRINT LABEL
2A37 23                         INX     H
2A38 CD5D3C                     CALL    TTY
2A3B 05                         DCR     B
2A3C C2362A                     JNZ     LOOP1
2A3F 0E20                       MVI     C,20H   ;PRINT TWO SPACES
2A41 23                         INX     H
2A42 CD5D3C                     CALL    TTY
 945 0E20                       MVI     C,20H
<A47 CD5D3C                     CALL    TTY
2A4A 23                         INX     H
2A4B CD592A                     CALL    BYTE    ;PRINT MSB OF ADDRESS
```

```
2A4E 2B                 DCX     H
2A4F CD592A             CALL    BYTE    ;PRINT LSB OF ADDRESS
2A52 CD6C2A             CALL    NEX
2A55 CD812A             CALL    BLANK   ;PRINT 6 SPACES
2A58 C9                 RET
2A59 7E         BYTE:   MOV     A,M     ;PRINT A BYTE AS
                ;   TWO ASCII CHARACTERS
2A5A 0F                 RRC
2A5B 0F                 RRC
2A5C 0F                 RRC
2A5D 0F                 RRC
2A5E E60F               ANI     0FH
2A60 CD662A             CALL    HXD
2A63 7E                 MOV     A,M
2A64 E60F               ANI     0FH
2A66 CDC53C     HXD:    CALL    CONV    ;CALLS MONITOR CONVERT SB-RT.
2A69 C35D3C             JMP     TTY
2A6C 7B         NEX:    MOV     A,E     ;SET D&E TO NEXT SYMBOL
2A6D C608               ADI     8
2A6F 5F                 MOV     E,A
2A70 D0                 RNC
 A71 14                 INR     D
2A72 C9                 RET
2A73 53594D42 MSG:      DB      'SYMBOL TABLE',0DH,0AH
2A77 4F4C2054
2A7B 41424C45
2A7F 0D0A
2A81 0606       BLANK:  MVI     B,6     ;PRINT 6 BLANKS
2A83 0E20       BLK0:   MVI     C,20H   ;PRINT THE NUMBER OF BLANKS
                ;   IN REG B.
2A85 CD5D3C             CALL    TTY
2A88 05                 DCR     B
2A89 C8                 RZ
2A8A C3832A             JMP     BLK0
2A8D 7E         ALFA1:  MOV     A,M
2A8E FE5A               CPI     5AH     ;COMPARE TO 'Z'
2A90 D2992A             JNC     NOALFA  ;JMP IF ACC>'Z'
2A93 FE41               CPI     41H     ;COMPARE TO 'A'
2A95 DA992A             JC      NOALFA  ;JMP IF ACC<'A'
2A98 C9                 RET
2A99 23         NOALFA: INX     H
2A9A 13                 INX     D
2A9B 05                 DCR     B
2A9C C28D2A             JNZ     ALFA1
2A9F C36B38             JMP     START
0000                    END
```

Ref. # AB57

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 ☐ Other_____ (use additional sheets if necessary)

Program Title

STATEMENT-COUNTER

Function

All Assembler statements in a 8080 source file are counted.

Required Hardware

Intellec MDS 800

Required Software

MDS Monitor

Input Parameters

The source file to be counted.

Output Results

Output message on the TTY.

Registers Modified: All	Programmer: J. Thommen
RAM Required: 640 bytes	Company: MOSTEC, LIESTAL/SWITZERLAND
ROM Required: --	Address: Fraumattstr. 11
Maximum Subroutine Nesting Level: 1	City: CH-4410 Liestal
Assembler/Compiler Used: 8080 MDS MACRO ASS. Vers. 1.0	State: Switzerland

Ref. # AB58

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 ☐ Other _____ (use additional sheets if necessary)

Program Title

9600

Function

Re-initialize CRT UART for 9600 Baud

Required Hardware

Intellec MDS development system

Required Software

None

Input Parameters

Output Results

Registers Modified:	Programmer: Evan Schaffer
RAM Required:	Company: Interface Consultants
ROM Required:	Address: P. O. Box 952
Maximum Subroutine Nesting Level:	City: Santa Cruz
Assembler/Compiler Used: ISIS 8080 MACRO V1.1	State: California 95060

```
                        ;REF. NO. AB58
                        ;PROGRAM TITLE 9600 RE-INITIALIZE CRT UART FOR 9600 BAUD
                        ;
                        ;
                        ;
                        ;
                        ;       RE-INITIALIZE CRT TO 9600 BAUD
                        ;
                        ;
                        ;
0001                    MDS             SET     1
0000                    INTELLEC        SET     NOT MDS AND 1
                        ;
                        ;
3100                            ORG     3100H
                        START:
3100 3E40                       MVI     A,40H
3102 D3F7                       OUT     CRTC    ; SEND RESET COMMAND
3104 3E4E                       MVI     A,04EH
3106 D3F7                       OUT     CRTC    ; SEND 9600 BAUD COMMAND
3108 3E27                       MVI     A,27H
310A D3F7                       OUT     CRTC    ; ENABLE UART
                        ;
                        IF      INTELLEC
                                JMP     $
                        ENDIF
                        IF      MDS
310C 0E09                       MVI     C,EXIT
310E 111431                     LXI     D,EBLK
3111 CD4000                     CALL    ISIS
                        ;
                        ;
                        EBLK:
3114 1631                       DW      ESTAT
                        ESTAT:
3116                            DS      2
                        ENDIF
                        ;
                        ;
00F7                    CRTC    EQU     0F7H
00F5                    TTC     EQU     0F5H
0009                    EXIT    EQU     9
0040                    ISIS    EQU     64
3100                            END     START
```

Ref. # AE4

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 ☐ Other_____ (use additional sheets if necessary)

Program Title

SNAP DUMP 8080

Function

To provide register and memory dumps for software debug.

Required Hardware

MDS-800 System

Required Software

MDS Monitor subroutine for list output.

Input Parameters

1) Calling Address
2) Options Flags
3) Start Memory Dump Address
4) End Memory Dump Address

Output Results

Printed Snap Message with Requested Data Displayed.

Registers Modified: NONE, all are saved/restored	**Programmer:** S. G. Thompson
RAM Required: 17 bytes	**Company:** Harris Controls
ROM Required: 554 bytes	**Address:** P. O. Box 430
Maximum Subroutine Nesting Level: 3	**City:** Melbourne
Assembler/Compiler Used: Microtek Version 3.1	**State:** Florida 32901

USERS MANUAL FOR
SNAP DUMP 8080

by: Steven Thompson

Calling Sequence:

```
CALL SNAP
DB  FLAGS+ID
DW  START
DW  END
```

Calling Parameters:

Flags/ID Byte:

10NNNNNN	=	No Memory Dumped
01NNNNNN	=	No Register Dumped
11NNNNNN	=	No Register or Memory Dumped
00NNNNNN	=	Both Register and Memory Dumped

N = ID# used to identify dump

NOTE: The flag option bits may be specified in any combination

Start Adr./End Adr. Define the start and end of the memory area to be dumped.

Character Output Subroutine Assignment

The program presently uses the MDS monitors character output subroutine with the reverse break (control "b"). This allows assignment of the output and interruption of the printing by the operator. The equate to L0 may be used to change the assignment.

SAMPLE OUTPUT

```
SNAP AT: AAAA ID=BB
REGISTERS: A=QQ B=RR C=SS D=TT E=UU H=VV L=WW CC=XX=SZ0A0P1C SP=YYYY
CCCC DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD *EEEEEEEEEEEEEEEE*
```

WHERE:
AAAA = Address of Snap In Hex
BB = Snap Identification Number in Decimal
QQ = A Register Contents In Hex
RR = B Register Contents In Hex
SS = C Register Contents In Hex
TT = D Register Contents In Hex
UU = E Register Contents In Hex
VV = H Register Contents In Hex
WW = L Register Contents In Hex
XX = Condition Code Register Contents In Hex
YYYY = Stack Pointer Register Contents In Hex
CCCC = Hex Address of 16 Words of Storage Dumped
DD = Hex Memory Data
E = ASCII Interpretation of Memory Data if (A-Z, 0-9, blank) else a Period.

Ref. # AB59

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 ☐ 8080 ☐ 3000 ☐ Other_____ (use additional sheets if necessary)

Program Title

RTM (real time monitor).

Function

RTM is a small real time operating system which is meant to supervise periodic execution of RT-programs in a fixed priority manner.

It has also a few standard output routines which must be used.

Required Hardware

The monitor itself require the INTELLEC 8080 SYSTEM
In addition comes a real time clock.

Required Software

MCS-80 system monitor CO (console output) is used, but can of course easely be replaced.

Input Parameters

Each RT-program must have a description table located from 155H.

Output Results

Depends on user programs.

Note: To start up RTM supply RST4 via console, start in address F6 and turn on the real time clock.

Registers Modified: All Register Saved	Programmer: Odd Stormo
RAM Required: 155 HEX CELL	Company: ØSTFOLD TEKNISKE SKOLE
ROM Required:	Address: 1712 Valaskjold
Maximum Subroutine Nesting Level: 5 until start of user program	City:
Assembler/Compiler Used: 8080 MACRO ASSEMBLER, VER.3.0	State: NORWAY

Ref. # AB60

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

FORMAT INTEL DATA

Function

Procedure FMINDA is part of a bootloader which loads object code from from a cassette tape into the Intellec MDS memory. The object code, from either the MAC80 Assembler or PL/M Compiler (or both), is written onto cassette tape using a T.I. Silent 700. Procedure FMINDA takes each 90 word record containing the object code in ASCII bytes, reformats the data and stores the object code in the designated location in RAM.

Required Hardware

Required Software

Input Parameters

Output Results

Registers Modified: ALL	**Assembler/Compiler Used:** PL/M COMPILER
RAM Required: 380 BYTES	**Programmer:** VICTOR H. SAUCEDO
ROM Required:	**Company:** Honeywell Information Systems
Maximum Subroutine Nesting Level:	**Address:** P.O. Box 6000 Phoenix, Arizona 85005

Ref. #AB61

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004 ☐4040 ☐8008 ☐8080 ☐3000 ☐Other_____ (use additional sheets if necessary)

Program Title

LIST VER 3.0

Function

To display/print ISIS files or MDS Monitor listings on the console in pages with optional line-numbers, optional assembly listing error only display, and for ISIS files, optional listing between given line-numbers only. LIST includes all tabs.

UPDATED VERSION 2/28/77

Required Hardware

MDS, Console

Required Software

Input Parameters

MDS Monitor

ISIS (Optional)

Monitor users set D = 0 for no line numbers, else numbers.

E = 0 for normal listing, else error only.

then go to start address.

Output Results

ISIS users give a LIST as a console command. See the listing for details of syntax and options. e.g.

- LIST FILE NE 143, 227 lists only lines in error between 143 and 227 inclusive, with each such line numbered.

MDS Monitor - Sets up the list assignment. All subsequent list output goes through LIST.

ISIS - Displays the required file.

After each page key no. of lines wanted (0 - 9) or escape to ISIS.

Registers Modified:	Programmer:
RAM Required:	Company:
ROM Required:	Address:
Maximum Subroutine Nesting Level:	City:
Assembler/Compiler Used:	State:

Ref. # AA12

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004 ☐4040 ☐8008 ☒8080 ☐3000 ☐Other_____ (use additional sheets if necessary)

Program Title

8080 CPU Exerciser Routine

Function

Designed as an on-line, periodic, exercising program. Executes almost all instruction in the 8080 to ensure proper functioning. A set of instructions is executed to which a predefined result is expected. If result does not match, program halts. See attached sheets.

Required Hardware

N/A

Required Software

None

Input Parameters

None

Output Results

N/A

Registers Modified: All including SP	Programmer: W. Iwamoto/R. Lonchar
RAM Required: 3 Bytes	Company: North Electric Company
ROM Required: 376 Bytes	Address: P. O. Box 20345
Maximum Subroutine Nesting Level: 2	City: Columbus
Assembler/Compiler Used: ISIS 8080 Macro Assembler	State: Ohio 43220

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

Requirements and Notes:

A. The 3 Bytes at Locations 40FCH-40FEH are to be RAM locations.

B. Program is address dependent, occuring from location 4102H. The CPI instruction 4107H will have to be changed if RAM addresses do not correspond to the ones in this assembly. Adjustment may also have to be made to the ORI instruction at location 4133H. Accumulator should contain 0C3H after the operation.

C. Numbers at locations 401FH-4024H are random and may be changed. This is also true for the LVI instruction at location 4092H. This instruction requires a RAM address.

D. Instruction at location 4179H should be changed to a JMP or an RET instruction.

No results are generated and no input parameters are needed. Program either passes or halts.

Ref. # AB62

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004 ☐4040 ☐8008 ☒8080 ☐3000 ☐Other_____ (use additional sheets if necessary)

Program Title

BTP (Binary Tape Programme)

Function

To read, check or punch binary paper tape directly to and from MDS memory using a rubout character as the tape index. Specified blocks or memory can be inverted (complemented). All periferials are assigned by the resident monitor.

Required Hardware

MDS 800
Console Device
Listing Device
Paper Tape Reader
Paper Tape Punch

Required Software

MDS Monitor

Input Parameters

Initial command R,V,P,I,M or D then MDS memory addresses specified in hexidecimal followed by carriage return.

M and D do not require addresses or CR.
The correct format for the other commands is shown in the listing.

Output Results

Tape will be punched when using P command with blank leader and trailer.
Data is immediately preceded by a rubout character and the data from memory is addressed along sequential locations on the tape.
All commands except P, make software changes only.
An error signal will result if:

1. The memory doesn't write correctly.
2. The memory addresses are not specified correctly.
3. The tape doesn't agree with memory when verifying.

Registers Modified: All	Programmer: Jim Arkell
RAM Required: 0.5K for 48H (37B0 to 37FF) Programme	Company: RAPID RECALL LIMITED
ROM Required: 0.5K when ROM resident	Address: 9 Betterton Street
Maximum Subroutine Nesting Level:	City: London
Assembler/Compiler Used: ISIS MACRO 8080 V1.0	State: WC2H 9BS, England

Ref. # AB63

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

intel

☐ 4004 ☐ 4040 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program Title

LERR (List Assembly Errors)

Function

Searches diskette list file for assembly error statements and prints all such statements on console device.

Required Hardware

MDS-80, (32K RAM), TTY, Diskette System

Required Software

ISIS V1.0, MDS Monitor CO routine

Input Parameters

ISIS 8080 Macro Assembler Program
list file name, such as "FILE.LST".

Output Results

Listing on console device of all error containing program statements.

Registers Modified: All	**Assembler/Compiler Used:** ISIS 8080 Macro Assembler
RAM Required: See Listing	**Programmer:** M. Polad
ROM Required: See Listing	**Company:** Data Card Corporation
Maximum Subroutine Nesting Level: See Listing	**Address:** 7625 Parklawn Ave Minneapolis, MN. 55435

```
                        ;REF. NO. AB63
                        ;PROGRAM TITLE LERR


                        TITLE   'PROGRAM LERR                   9/16/76'

                        ;COMMAND: <LERR FILENAME>

                        ;PROGRAM SEARCHES THROUGH LIST FILE (OUTPUT FROM ASM80)
                        ;PRINTS ALL STATEMENTS CONTAINING ASSEMBLY ERRORS ON THE
                        ;DEVICE.

4000                    ORG     4000H

0000            OPEN    EQU     0
0003            READ    EQU     3
000C            ERROR   EQU     12
0040            ISIS    EQU     64
0009            EXIT    EQU     9
000D            CR      EQU     0DH
 00A            LF      EQU     0AH
F809            CO      EQU     0F809H

4000 314A41     BEGIN:  LXI     SP,STACK+10
4003 0E03               MVI     C,READ
4005 119B40             LXI     D,RBLK
4008 CD7240             CALL    SYS
400B 0E00               MVI     C,OPEN
400D 11A540             LXI     D,OBLK
4010 CD7240             CALL    SYS     ;OPEN LIST FILE ;READ FILENAME FROM :CI:
4013 CD5B40             CALL    RD      ;READ BLOCK FROM LIST FILE TO BUF
4016 CD5440     LOOP:   CALL    GNC     ;GET A CHAR FROM BUF
4019 79                 MOV     A,C
401A FE0A               CPI     LF
401C C21640             JNZ     LOOP    ;LOOP UNTIL CHAR= LF
401F CD5440     LOOP1:  CALL    GNC     ;GET NEXT CHAR (FIRST ON LINE)
4022 79                 MOV     A,C
4023 FE0D               CPI     0DH
4025 CA1640             JZ      LOOP    ;IF CHAR= CR OR SPACE, LOOK FOR NEXT LF
4028 FE20               CPI     ' '
402A CA1640             JZ      LOOP
402D 79         CPHD:   MOV     A,C     ;SAVE FIRST CHAR ON LINE
402E 32BD40             STA     TEMP
4031 CD5440             CALL    GNC     ;GET 2ND CHAR ON LINE
4034 79                 MOV     A,C
 035 FE20               CPI     ' '
+037 C21640             JNZ     LOOP    ;IF 2ND CHAR NOT SPACE, IGNORE (PAGE HDG
403A 3ABD40             LDA     TEMP    ;GET FIRST CHAR
403D 4F                 MOV     C,A
```

```
403E CD09F8          CALL    CO      ;OUTPUT FIRST CHAR
4041 4E              MOV     C,M     ;GET 2ND CHAR
4042 CD09F8  WRT:    CALL    CO      ;SEND CHARS TO :CO: UNTIL NEXT LF
4045 CD5440          CALL    GNC
4048 79              MOV     A,C
4049 FE0A            CPI     LF
404B C24240          JNZ     WRT
404E CD09F8          CALL    CO
4051 C31F40          JMP     LOOP1

4054 23      GNC:    INX     H       ;GET CHAR FROM BUF
4055 4E              MOV     C,M
4056 05              DCR     B
4057 CC5B40          CZ      RD      ;IF LAST CHAR IN BUF, READ 128 CHAR FROM
405A C9              RET

405B C5      RD:     PUSH    B
405C 0E03            MVI     C,READ
405E 11AF40          LXI     D,RBLK1
4061 CD7240          CALL    SYS
4064 2A3E41          LHLD    ACTUAL
 067 7C              MOV     A,H
4068 B5              ORA     L
4069 CA8540          JZ      DONE    ;IF ACTUAL = 0, DONE
406C C1              POP     B
406D 47              MOV     B,A
406E 21BD40          LXI     H,BUF-1
4071 C9              RET

4072 CD4000  SYS:    CALL    ISIS
4075 3ABB40          LDA     STATUS
4078 B7              ORA     A
4079 C27D40          JNZ     ERR
407C C9              RET

407D 0E0C    ERR:    MVI     C,ERROR
407F 11B940          LXI     D,EBLK
4082 CD4000          CALL    ISIS
4085 CD9040  DONE:   CALL    CRLF
4088 0E09            MVI     C,EXIT
408A 11B940          LXI     D,XBLK
408D CD4000          CALL    ISIS

4090 0E0D    CRLF:   MVI     C,CR
4092 CD09F8          CALL    CO
4095 0E0A            MVI     C,LF
 097 CD09F8          CALL    CO
+09A C9              RET

409B 0100    RBLK:   DW      1
```

```
409D BE40              DW      BUF
409F 8000              DW      128
40A1 3E41              DW      ACTUAL
40A3 BB40              DW      STATUS

40A5 AF40      OBLK:   DW      AFTN
40A7 BE40              DW      BUF
40A9 0100              DW      1
40AB 0000              DW      0
40AD BB40              DW      STATUS

               RBLK1:
40AF           AFTN:   DS      2
40B1 BE40              DW      BUF
40B3 8000              DW      128
40B5 3E41              DW      ACTUAL
40B7 BB40              DW      STATUS

               EBLK:
40B9 BB40      XBLK:   DW      STATUS

 3BB           STATUS: DS      2
40BD           TEMP:   DS      1
40BE           BUF:    DS      128
413E           ACTUAL: DS      2
4140           STACK:  DS      10
4000                   END     BEGIN
```

Ref. # AE5

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004 ☐4040 ☐8008 ☐8080 ☐3000 ☐Other_____ (use additional sheets if necessary)

Program Title

TRACE VER 7.0

Function

TRACE single-step through any user's program for a keyed number of instructions or until DI executed. It displays the user's registers flags, op-code mnemonic and data after each instruction. The monitor routines may be used at any time to control execution. The user's programme may be in RAM, ROM or both.

Required Hardware

MDS, Console

Required Software

MDS Monitor Ver 1

ISIS (Optional)

Input Parameters

MDS Monitor users load own program and TRACE and go to TRACE with a breakpoint at their start address. ISIS users may command - TRACE FILENAME where FILENAME is a binary file. Escape to ISIS and to the Monitor is provided with .G (C.R.) to restart TRACE.DI (0F3H) may be put at any number of breakpoints and CLEAR is a run to breakpoint without printing command.

Output Results

See example.

Note that interrupts below level 2 cannot be used.

CI cannot be used, CO works only on single shot

DI cannot be traced - it acts as a breakpoint

EI is not traced correctly.

Registers Modified: N/A	Programmer: C. J. Lusby Taylor
RAM Required: MDS 1021 bytes ISIS 1107 bytes	Company: Satchwell Control Sys. Ltd.
ROM Required: None	Address: P. O. Box 57
Maximum Subroutine Nesting Level: N/A. 4 bytes user's stock	City: Farnham Road
Assembler/Compiler Used: ISIS 8080 MACRO ASSEMBLER	State: SLOUGH, Berks.

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. # AB64

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080

(use additional sheets if necessary)

Program Title

ADCCP Remainder Routine

Function

Generates ADCCP remainder term R (X) as described in ADCCP document X3534/589 draft 5, pages 74-76. The ADCCP generator polynomial is $P(X) = X^{16} + X^{12} + X^5 + 1$. See attached note for methodology.

Required Hardware

Intel 8080 Microcomputer

Required Software

This routine can run as a subroutine to a main program. The main program enters the routine with each message byte, most significant byte first. With the initial byte, the remainder, which is in the B and C registers must be preset, which for ADCCP is a value of all ones. Of course, the B and C register values generated by the routine must be returned to the routine along with subsequent message bytes. PUSH, POP, RET, etc. instructions are added according to users design.

Input Parameters

A Register - Message data bytes
B Register - High order bits of R (X)
C Register - Low order bits of R (X)

Output Results

For a string of bytes to be transmitted the R (X) generated by this routine, upon being inverted, can become the final two bytes of the transmitted sequence, i.e. the Frame Check Sequence. For a string of bytes that has been received, B and C will contain 1DH and ØFH respectively, if the transmitted sequence has been received without error. The foregoing assumes that both B and C registers are preset to all ones.

Registers Modified:	Assembler/Compiler Used:
A, B, C, D, E	8080 Macro Assembler Version 1.0
RAM Required:	**Programmer:**
22H	George H. Mineah
ROM Required:	**Company:**
Ø	Aeronutronic Ford Corporation
Maximum Subroutine Nesting Level:	**Address:** 2880 E. Fountain Blvd. Colorado Springs, CO. 80910
Does not alter current maximun	

98-0348

```
                        ;REF. NO. AB64
                        ;PROGRAM TITLE ADCCP REMAINDER ROUTINE
                        ;
                        ;
                        ;
                        ;ADCCP REMAINDER ROUTINE
                        ;IN:
                        ;A=MESSAGE BYTE MOST SIGNIFICANT FIRST
                        ;B=HIGH ORDER BITS OF R(X) PRESET TO FFH FOR FIRST BYTE
                        ;C=LOW ORDER BITS OF R(X) PRESET TO FFH FOR FIRST BYTE
                        ;OUT:
                        ;B=HIGH ORDER BITS OF R(X)
                        ;C=LOW ORDER BITS OF R(X)
3D00                    ORG     3D00H
3D00 A8         ARR:    XRA     B
3D01 57                 MOV     D,A
3D02 0F                 RRC
3D03 0F                 RRC
3D04 0F                 RRC
3D05 0F                 RRC
3D06 E60F               ANI     0FH
3D08 AA                 XRA     D
3D09 5F                 MOV     E,A
3D0A 07                 RLC
3D0B 07                 RLC
3D0C 07                 RLC
3D0D 07                 RLC
3D0E E6F0               ANI     0F0H
3D10 A9                 XRA     C
3D11 47                 MOV     B,A
3D12 7B                 MOV     A,E
3D13 0F                 RRC
3D14 0F                 RRC
3D15 0F                 RRC
3D16 E61F               ANI     1FH
3D18 A8                 XRA     B
3D19 47                 MOV     B,A
3D1A 7B                 MOV     A,E
3D1B 0F                 RRC
3D1C 0F                 RRC
3D1D 0F                 RRC
3D1E E6E0               ANI     0E0H
3D20 AB                 XRA     E
3D21 4F                 MOV     C,A
3D22 76                 HLT
0000                    END
```

Ref. # AB65

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004 ☐4040 ☐8008 ☐8080 ☐3000 ☐Other_____ (use additional sheets if necessary)

Program Title

WIPE

Function

"Automates" the process of file deletion on ISIS diskettes. Only those files with attributes of W = Ø and F = Ø can be wiped.

Required Hardware

MDS with floppy disks

Required Software

ISIS

Input Parameters

File name of file that contains replica of normal directory of the diskette to be "wiped". If this name is for example TEMP from - DIR to TEMP$1 then:

1. WIPE TEMP allows one to select files to be deleted.
2. WIPE TEMP$A deletes all files that are not format or write protected.

Output Results

At console get displayed names of files up for deletion (if condition (1)). Type "Y" to query if you want to delete "N" if not.

Registers Modified: ALL	Programmer: Bob Glossman
RAM Required: 16K System Minimum	Company: Stanford University
ROM Required:	Address: c/o V. Grinich ERL 101
Maximum Subroutine Nesting Level:	City: Stanford
Assembler/Compiler Used: MDS MAC 80	State: California 94305

Ref. # AB66

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 ☐ Other_____ (use additional sheets if necessary)

Program Title

TABS

Function

Takes files that use control-I as tab character and expand tabs so that legible listings can be obtained.

Required Hardware

MDS with diskette system.

Required Software

ISIS

Input Parameters

Source file and (optionally) destination file, i.e.
-- TABS PROG 1
or
-- TABS PROG 1 to PROG 2

Output Results

If no destination file used (i.e. to etc. omited) create a new file PROG1.TBS with tabbing done. Otherwise the file PR062 will contain the tabbed result.

Registers Modified: --	Programmer: Bob Glossman
RAM Required: --	Company: Stanford University
ROM Required: --	Address: c/o V. Grinich IOIERL
Maximum Subroutine Nesting Level: --	City: Stanford
Assembler/Compiler Used: MAC80	State: California 94305

Ref. # AA13

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 XX 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

INTELLEC MICROCOMPUTER DEVELOPMENT SYSTEM DIAGNOSTIC CONFIDENCE TEST VERSION 1.1

Function

Exercise Intellec Microcomputer Development System to verify functionality.

Required Hardware

Intellec MDS 800

Required Software

REVISED TO VERSION 1.1 6/79

Input Parameters

None

Output Results

System functionality

DISKETTE AVAILABLE IN OBJECT CODE ONLY. LISTING IS IN SOURCE CODE. DISKETTE IS PRICED AT $35.00. PAPER TAPE IS NOT AVAILABLE.

CAUTION: BEFORE USING THIS PROGRAM, READ INSTRUCTIONS THOROUGHLY.

Registers Modified: ALL	Programmer:
RAM Required: 16 K	Company:
ROM Required: None	Address:
Maximum Subroutine Nesting Level: --	City:
Assembler/Compiler Used: --	State:

Ref. # AB67

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004 ☐4040 ☐8008 ☒8080 ☐3000 ☐Other_____ (use additional sheets if necessary)

Program Title
Intellec 8/Mod 80 Text Editor

Function
Text Editor

Required Hardware
Intellec 8/Mod 80 and TTY

Required Software
Mod 80 Monitor

Input Parameters

Output Results

Listing available from Insite for a prepaid $15.00 handling fee. Paper tape is not available except when ordering system. Program Diskette not offered.

Registers Modified: All	Programmer:
RAM Required: prox 4K bytes	Company:
ROM Required:	Address:
Maximum Subroutine Nesting Level:	City:
Assembler/Compiler Used: PL/M	State:

Ref. # AB68

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 ☐ Other_____ (use additional sheets if necessary)

Program Title

2708 Prom Programmer for Intellec 8/MOD80

Function

To program, read, and verify a 2708 type PROM of up to 1024 Bytes. This cannot be done using the Prom logic of the Intellec 8/MOD80 or Intellec MDS Monitor because of the special programming algorithm required by the 2708.

Required Hardware

Intellec 8 MOD80 or Intellec MDS and Universal Prom Programmer (UPP)

Required Software

System Monitor

Input Parameters

See Text of Program

Output Results

Listing available form Insite for a prepaid $15.00 handling fee. Paper tape is not available except when ordering system. Program Diskette is not offered.

Registers Modified: All	Programmer:
RAM Required: 1.5K	Company:
ROM Required:	Address:
Maximum Subroutine Nesting Level:	City:
Assembler/Compiler Used: 8080 Macro Assem. Ver. 2.3	State:

Ref. # AB69

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 ☐ Other_____ (use additional sheets if necessary)

Program Title

Intellec 8/MOD80 Monitor

Function

Controls Intellec 8/MOD80

Routines for

Required Hardware

Intellec 8/MOD80

Required Software

--

Input Parameters

See Listing

Output Results

See Listing

Listing available form Insite for a prepaid $15.00 handling fee. Paper tape is not available except when ordering system. Program Diskette is not offered.

Registers Modified: All	Programmer:
RAM Required: 2K	Company:
ROM Required:	Address:
Maximum Subroutine Nesting Level:	City:
Assembler/Compiler Used: 8080 Macro Assem. Ver. 2.2	State:

Ref. # AB70

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004 ☐4040 ☐8008 ☐8080 ☐3000 ☐Other_____ (use additional sheets if necessary)

Program Title

Intellec/MDS Monitor Version 2

Function

Intellec/MDS Interactuve Program for Handling Six I/O Devices, and Utility Routines for Display/Modify 8080 Memory and Registers.

Required Hardware

Intellec MDS

Required Software

Input Parameters

See Program Listing

Output Results

See Program Listing

Listing available form Insite for a prepaid $15.00 handling fee. Paper tape is not available except when ordering system. Program Diskette not offered.

Registers Modified: All	Programmer:
RAM Required: 2K bytes	Company:
ROM Required:	Address:
Maximum Subroutine Nesting Level:	City:
Assembler/Compiler Used: 8080 Macro Assembler Ver. 2.4	State:

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. # AB72

intel®

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 (use additional sheets if necessary)

Program Title

Utility Macros for 8080

Function

Save time in coding and debugging 8080 programs.

Required Hardware

Required Software

PDP-11 Cross Assembler for 8080 (e.g. Intel Library Ref no. C2)

Input Parameters

If other assemblers are used, modify syntax accordingly.

Output Results

Macro expansions

Registers Modified: n/a	**Assembler/Compiler Used:** PDP-11 Cross Assembler
RAM Required: n/a	**Programmer:** Alan C. Hui
ROM Required: n/a	**Company:** STANDARD ENGINEERING CORP.
Maximum Subroutine Nesting Level: n/a	**Address:** 44800 Industrial Drive, Fremont CA 94538

98-034B

Ref. # AB73

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 ☒ Other SBC 80/10 (use additional sheets if necessary)

Program Title

I/O Port Exerciser

Function

The program allows the user to output patterns to PPI ports and to read and print input port values under keyboard control.

Required Hardware

SBC 80/10, TTY or RSZ32-C compatable terminal

Required Software

EXERCISER PROM

Input Parameters

see write up

Output Results

see write up

Registers Modified: ALL	Programmer: Jeffrey W. Scott
RAM Required: 100 bytes	Company: Computer Applications
ROM Required: 1024 bytes	Address: 3030 Bridgoway
Maximum Subroutine Nesting Level:	City: Sausalito
Assembler/Compiler Used: MDS MACRO Assembler	State: California 94965

Ref. # AB75

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 ☐ Other_____ (use additional sheets if necessary)

Program Title

Fly Reader Driver

Function

1. Sets up user IO entry pt. for monitor
2. Generates read CMD for reader
3. Inputs ASCII character and sets carry if no char. for 15 μs

Required Hardware

Fly Reader 30 - Teleterminal Corp.
12 Cambridge Street
Burlington, Ma. 01803
(Plugs into TTY plug on MDS 800)

Required Software

Input Parameters

Output Results

Returns ASCII character in a register; sets carry if no character read in 15 μs.

Registers Modified: A	Programmer: William J. Casey
RAM Required: 025H	Company: Logic Sciences
ROM Required: 0	Address: 6420 Hillcroft, # 317
Maximum Subroutine Nesting Level: 0	City: Houston
Assembler/Compiler Used: 8080 MDS Assemb. V. 1.0	State: Texas 77081

```
                ;REF. NO. AB75
                ;PROGRAM TITLE FLY READER DRIVER
                ;
                ;
                ;

                ;                               "FLY READER DRIVER"

3FEF                    ORG 3FEFH
3FEF 0038               DW 3800H        ;READER I/O ENTRY POINT
3800                    ORG 3800H
00F9            PTR     EQU 0F9H
00FF            RTC     EQU 0FFH
3800 C5                 PUSH B
3801 060F               MVI B,0FH       ;INITIALIZE COUNTER
3803 3E08               MVI A,8         ;GENERATE READ COMMAND PULSE
3805 D3F9               OUT PTR
3807 3E0C               MVI A,0CH
3809 D3F9               OUT PTR
380B DBF9       STAT:   IN PTR          ;INPUT PTR STATUS
 380D E602              ANI 2
380F CA2138             JZ DATA
3812 DBFF               IN RTC          ;INPUT REAL TIME CLOCK STATUS
3814 DBFF               IN RTC
3816 CA0B38             JZ STAT
3819 05                 DCR B           ;DECREMENT COUNTER
381A C20B38             JNZ STAT
381D AF                 XRA A
381E 37                 STC             ;SET CARRY FOR NO DATA IN 15MS FROM READ
381F C1                 POP B
3820 C9                 RET
3821 DBF8       DATA:   IN 0F8H         ;INPUT DATA
3823 2F                 CMA
3824 C1                 POP B
3825 C9                 RET
0000                    END
```

Ref. # AB76

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 XX 8080 ☐ 3000 ☐ Other_____ (use additional sheets if necessary)

Program Title

NON-ENCODED KEYBOARD SUBROUTINE HANDLER

Function

The subroutine functions as a handler for two independent, 16-button, non-encoded keyboards. The handler tests for any key closure, provides contact bounce elimination, allows only two-key rollover, and generates a unique code for the key closure.

Required Hardware

(SEE COMMENTS ON ATTACHED SOURCE LISTING)

Required Software

Pointers to keyboard processing routines (PKYB1, PKYB2) must be assigned by the calling program before entering the subroutine. The handler can be executed on a programmed scan cycle, or on an interrupt basis if the hardware is modified allowing any key depression to generate an external hardware interrupt.

Input Parameters

Output Results

B-REGISTER = key switch code
(Re-arranging the keyboard look-up table assigns a different key switch code to the switch matrix crosspoints.)

Registers Modified: A,B,C,D,E,H,L	Programmer: VITO A. TRUJILLO
RAM Required: 3 bytes	Company: COMPUTER CONSULTANT
ROM Required: D4H bytes	Address: 2940 BRAUN COURT
Maximum Subroutine Nesting Level: (Depends on calling program)	City: GOLDEN
Assembler/Compiler Used: ICOM FDOS-II Macroassembler	State: COLORADO 80401

Ref.# AB77

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 ☐ Other_____ (use additional sheets if necessary)

Program Title

LOAD

Function

To allow loading of ISIS-II absolute object modules into the area of ROM occupied by ISIS-II. (i.e. below 3180H)

Required Hardware

Intellec MDS-800
Single or Dual Drive Floppy Disk System
System Console

Required Software

To Compile: PL/M-80 and ASM80
To Execute: ISIS-II

Input Parameters

Name of file to be LOADED
e.g.

- LOAD :F1:TEST.PRC

Output Results

The program will print out the load address and start address of LOADED program on the system console and then exit to the monitor. To execute the program just loaded, use the monitor "GO" command with the start address given by the "LOAD" program.

Available in Object Code with a source listing. Diskette available in source code.

Registers Modified: All	**Programmer:** Ron Williams
RAM Required:	**Company:**
ROM Required:	**Address:**
Maximum Subroutine Nesting Level:	**City:**
Assembler/Compiler Used: PLM-80, ASM80	**State:**

THIS PROGRAM ALLOWS THE LOADING OF ISIS-II OBJECT MODULES WHICH HAVE A LOAD ADDRESS WITHIN THE ISIS-II PROGRAM AREA (I.E. BELOW 3180H). THE PROGRAM TO BE LOADED MUST BE IN ABSOLUTE BINARY FORMAT. LOADING OF RELOCATABLE MODULES IS NOT ALLOWED EITHER BY THE 'LOAD' PROGRAM OR BY THE ISIS-II BINARY LOADER.

TO EXECUTE THE 'LOAD' PROGRAM, SIMPLY TYPE THE WORD "LOAD" FOLLOWED BY ONE OR MORE SPACES AND THE NAME OF THE FILE TO BE LOADED.

FOR EXAMPLE: LOAD :F1:COMP.ABS

AFTER THE FILE HAS BEEN LOADED, THE PROGRAM WILL PRINT THE LOAD ADDRESS AND START ADDRESS ON THE SYSTEM CONSOLE AND THEN EXIT TO THE INTELLEC MONITOR. YOU CAN THEN EXECUTE YOUR PROGRAM BY TYPING "G" FOLLOWED BY YOUR PROGRAM'S START ADDRESS.

FOR EXAMPLE: G100

****

TO INSTALL THE 'LOAD' PROGRAM IN YOUR ISIS-II SYSTEM USE THE FOLLOWING STEPS:

1) USING THE ISIS-II TEXT EDITOR, GENERATE THE "SUBMIT" FILE SHOWN BELOW...

```
PLM80 :F1:LOAD.SRC
^E  (CONTROL/E)
ASM80 :F1:MOVER.SRC
LINK :F1:LOAD.OBJ,:F1:MOVER.OBJ,SYSTEM.LIB TO &
:F1:LOAD.LNK
LOCATE :F1:LOAD.LNK  CODE (3200H)
```

2) THE CONTROL/E ABOVE ALLOWS YOU TO TAKE OUT THE PL/M-80 DISKETTE AND INSERT THE ISIS-II DISKETTE BEFORE CONTINUING WITH THE INSTALLATION. AFTER INSERTING THE ISIS-II DISKETTE, TYPE A CONTROL/E TO CONTINUE WITH THE ASSEMBLY.

3) THE "SUBMIT" FILE ALONG WITH THE TWO SOURCE FILES SHOULD ALL BE ON DRIVE 1. BY PUTTING THE "SUBMIT" FILE ON A FILE CALLED 'LOAD.CSD' IN DRIVE 1 AND TYPING IN:

```
SUBMIT :F1:LOAD
```

YOU WILL HAVE THE 'LOAD' PROGRAM ON DRIVE 1.

Ref. # AE6

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 ☐ Other_____ (use additional sheets if necessary)

Program Title

SCAN

Function

Scans a listing file held on a diskette and prints assembly error on the line printer

Required Hardware

MDS 800, Diskette, Line Printer, Terminal

Required Software

ISIS Operating System

Input Parameters

Filename (from terminal)

Output Results

Prints assembly errors on line printer.

Registers Modified:	Programmer: C. D. Saunders
RAM Required: 955 Bytes + ISIS	Company: Marconi Instruments
ROM Required:	Address: Longacres St. Albans
Maximum Subroutine Nesting Level: 2	City: Hertfordshire
Assembler/Compiler Used: (MDS 800) 8080	State: U.K.

Ref.# AB78

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 ☐ Other_____ (use additional sheets if necessary)

Program Title

Driver for Tektronix 4010
Grafic Screen

Function

Procedures for controlling Tektronix 4010.
These procedures make it possible to use the
4010 as a grafic output unit for 8080.

Required Hardware

TTY on port 0 and 1.

Required Software

Console input and console output routines.

Input Parameters

Output Results

Registers Modified: all	Programmer: Henning Nielsen
RAM Required: 3/4 K	Company: Institut for Elektroniske Systemer
ROM Required:	Address: Aalborg University Centre
Maximum Subroutine Nesting Level: 3	City: P.O.Box 159 9100 Aalborg
Assembler/Compiler Used:	State: Denmark

Ref.# AB79

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 ☐ Other_____ (use additional sheets if necessary)

Program Title

T.I. SILENT 700 - SBC 80 MONITOR INTERFACE

Function

Allows SBC 80 monitor to be used with T.I. Silent 700 terminal in place of teletype. Tape operations operate at 120 characters/second.

Required Hardware

A. SBC, power supply, etc., set-up for 1200/300 baud operation
B. T.I. 733 ASR terminal equipped as follows:
   1) ADC (Automatic Device Control) or RDC (Remote Device Control)
   2) 1200 baud -- If 1200 baud option is not available, replace the statement
      CALL HIGH
      with three (3) NOP instruction wherever CALL HIGH appears (at RDRON + 3 and PCHON).

Required Software

SBC monitor with modifications as described.

Input Parameters

Accepts all monitor commands as specified in Intel documentation, with two exceptions: an additional carriage return is required after the issuance of and one is required at the completion of the R and W commands to permit tape operations at 120 characters/sec. The ADC or RDC option may be omitted if the tape playback/record mechanism is operated manually.

Output Results

Identical in operation with SBC 80 monitor, excepting changing terminal speed (optional).

Registers Modified: A, C, F/F's	Programmer: William M. Seifert
RAM Required: Same as SBC 80P monitor	Company: Los Alamos Scientific Laboratory
ROM Required: 192 BYTES	Address: P.O. Box 1663, MS/317
Maximum Subroutine Nesting Level: 3	City: Los Alamos
Assembler/Compiler Used: MAC80 Macro Assembler Ver. 2.4	State: New Mexico 87545

6/8/77

Ref. # AB80

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 ☐ Other_____ (use additional sheets if necessary)

Program Title

GLANCE

Function

To provide an easy way to visually examine a lengthy diskette resident text file. The file is transfered, line by line, to the CRT console with tab characters replaced by spaces. The display can be frozen and the speed of output changed. Quick jumps of maximum 25600 characters can be specified, both forward and backward. After such a jump the CRT screen will be filled up with text and the display frozen.

Required Hardware

MDS 800 with diskette and CRT.

Required Software

ISIS or ISIS-II
Monitor routines: Console status test (CSTS)
Console input (CI)

Input Parameters

Filename of the text file to be examined.
ISIS command: GLANCE filename.

Run time commands: 1. N (numeric) will change the speed of output
2. +N forward jump } by $50 \cdot 2^N$ characters
3. -N backward jump }
4. Control C to exit to ISIS

Output Results

Text file output to console.

Registers Modified:	Programmer: Esko Lehtinen
RAM Required:	Company: AB Bofors, dept. KCN
ROM Required:	Address: P.O. Box 500
Maximum Subroutine Nesting Level:	City: S-690 20 Bofors
Assembler/Compiler Used: Intellec MDS Macro Assembler	State: Sweden

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref.# AB81

☐ 4004 ☐ 4040 ☐ 8008 ☐ 8080 ☐ 3000

(use additional sheets if necessary)

Program Title

Keyboard Scanner

Function

Allows one to use a key matrix keyboard as the Console I/O device for a SDK 80 board, (converts key matrix to ASC II.)

Required Hardware

SDK 80 board with standard I/O (8255 at ØF4H). Any switch closure key matrix keyboard IK RAM on SDK 80.

Required Software

SDK 80 Monitor

Input Parameters

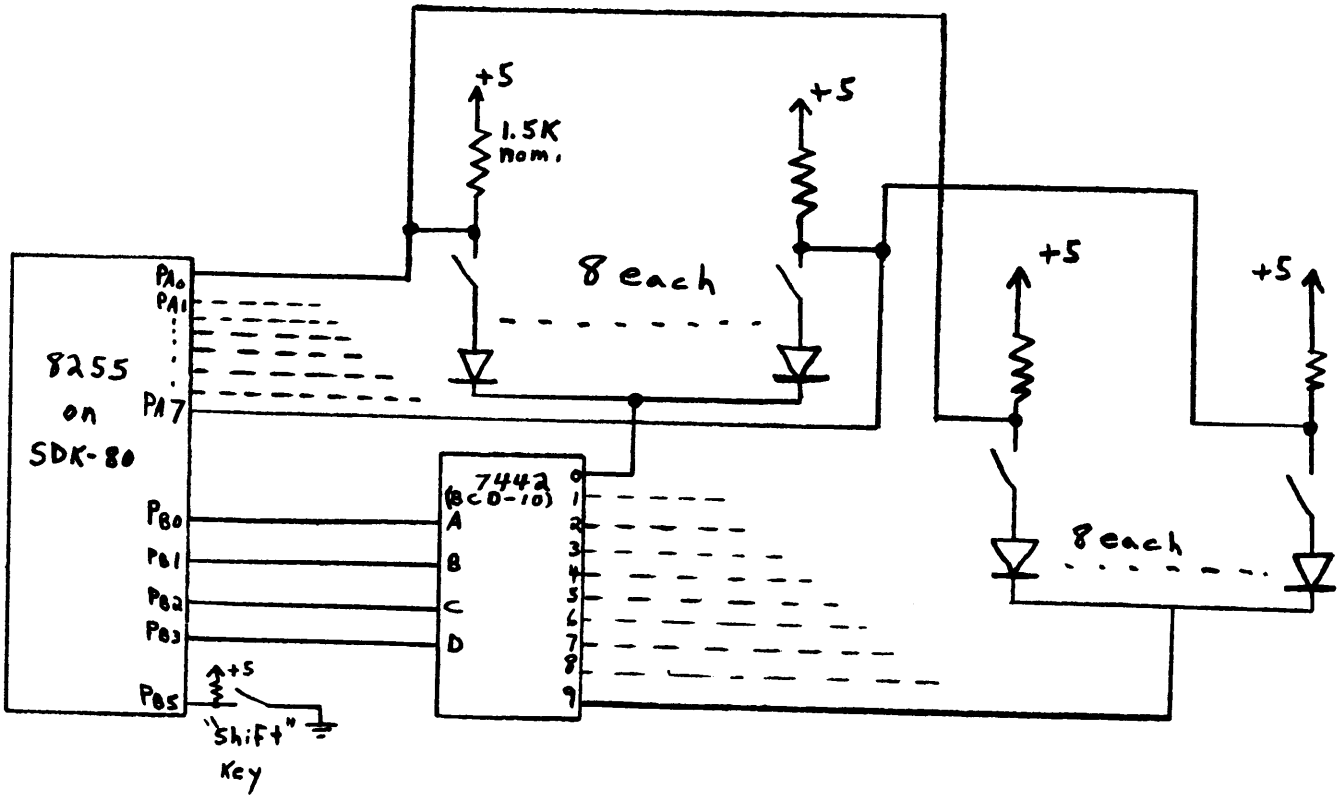
See attached sheet

Output Results

ASC II coded character in register A.

Registers Modified: A, Flags	**Assembler/Compiler Used:** Intellec MCS 80 Assembler
RAM Required: IK	**Programmer:** Lindsay Weaver
ROM Required: 256 bytes	**Company:** Mass. General Hospital Anesth.
Maximum Subroutine Nesting Level: 2	**Address:** Bio-Eng. Unit Fruit St., Boston, Mass. 02114

98-034C



Example of How to Interface

to an 80 Key Matrix

```
                ;REF. NO. AB81
                ;PROGRAM TITLE KEYBOARD SCANNER
                ;
                ;
                ;
                ;
1000                    ORG     1000H
                        ;WHEN THESE ROUTINES ARE USED TO REPLACE CI AND BREAK
                        ;IN THE SDK MONITOR, THE INSTRUCTIONS:
                        ;MVI  A,98H / OUT  0F7H  SHOULD BE ADDED TO THE
                        ;INITIALIZATION SECTION TO INITIALIZE THE 8255

1000 CD1510     CI:     CALL    SCAN    ;SUBSTITUTE FOR THE SDK CONSOLE IN ROUTI
1003 D20010             JNC     CI
1006 C9                 RET

0218            FRET    EQU     218H
0343            SRET    EQU     343H
1007 CD1510     BREAK:  CALL    SCAN    ;SUBSTITUTE FOR THE SDK BREAK ROUTINE
100A D20710             JNC     BREAK
100D FE1B               CPI      27     ;27=ESC THE SDK BREAK CHAR.
100F CA4303             JZ      SRET
1012 C31802             JMP     FRET

                SCAN:
                        ;SCAN IS A SUBROUTINE THAT SCANS THE KEYS ONCE.
                        ;IF A CHAR. IS TO BE RETURNED THE CARRY BIT IS SET
                        ;AND THE CHAR. IS IN A ON RETURN, IF NO CHAR IS TO BE
                        ;RETURNED THE CARRY IS ZERO ON RRETURN
                        ;DESTROYS: A,F/F'S
                        ;INPUTS: NONE
                        ;OUTPUTS: A,C-FLAG
                        ;CALLS: NOTHING

00F4            PA      EQU     0F4H
00F6            PC      EQU     0F6H
00F7            PCN     EQU     0F7H    ;8255 PORT ADDRESSES
1015 3E09               MVI     A,9
1017 D3F6               OUT     PC      ;SET OUTPUT FOR FIRST SCAN
1019 C5                 PUSH    B
101A D5                 PUSH    D
101B E5                 PUSH    H       ;SAVE REGISTERS
101C 0E04               MVI     C,4     ;THIS IS A DELAY TO PREVENT KEY BOUNCE F
101E 06FF       SC2:    MVI     B,255   ;CAUSING MULTIPLE ENTRIES
1020 05         SC4:    DCR     B
1021 C22010             JNZ     SC4
1024 0D                 DCR     C
1025 C21E10             JNZ     SC2
1028 4F                 MOV     C,A     ;INIT. C
1029 060A               MVI     B,10    ;PUT INDEX IN B
```

```
102B 219F11             LXI     H,KEY-1
102E 11A911             LXI     D,KEYN-1        ;GET POINTERS TO SCAN TABLES
1031 0D         SC05:   DCR     C
1032 23                 INX     H
1033 13                 INX     D
1034 DBF4               IN      PA      ;GET RESULTS
1036 F5                 PUSH    PSW     ;STORE A
1037 B6                 ORA     M
1038 2F                 CMA             ;(KEYS)*NOT(KEY(J)T-1)
1039 EB                 XCHG
103A 77                 MOV     M,A     ;STORE KEYN(J)
103B 79                 MOV     A,C
103C D3F6               OUT     PC      ;SELECT 8 KEYS
103E EB                 XCHG
103F F1                 POP     PSW
1040 2F                 CMA
1041 77                 MOV     M,A     ;STORE KEY(J)
1042 05                 DCR     B
1043 C23110             JNZ     SC05    ;DO THE LOOP 10 TIMES
1046 21AA11             LXI,    H,KEYN
1049 060A       SC20:   MVI     B,10    ;LOAD INDEX IN B
104B 7E         SC25:   MOV     A,M     ;GET BYTE
104C B7                 ORA     A       ;SET FLAGS
104D C25A10             JNZ     SC30    ;IF ONE OF THE BITS IS SET GO FIND ASCII
1050 23                 INX     H
1051 05                 DCR     B
1052 C24B10             JNZ     SC25    ;DO LOOP 10 TIMES
1055 37                 STC             ;IF WE GET HERE NO KEY WAS PRESSED
1056 3F                 CMC             ;CLEAR CARRY AND RETURN
1057 C37E10             JMP     SC99
105A 4F         SC30:   MOV     C,A     ;SAVE A
105B 78                 MOV     A,B
105C 3D                 DCR     A
105D 87                 ADD     A
105E 87                 ADD     A
105F 87                 ADD     A
1060 47                 MOV     B,A     ;B=(B-1)*8=J*8
1061 79                 MOV     A,C
1062 1F         SC35:   RAR
1063 DA6A10             JC      SC40    ;GET OUT OF LOOP IF CARRY SET
1066 04                 INR     B
1067 C26210             JNZ     SC35
106A DBF6       SC40:   IN      PC      ;OFFSET NOW IN B, NOW CHECK SHIFT
106C 2F                 CMA
106D E620               ANI     20H
106F CA7610             JZ      SC45
1072 3E50               MVI     A,80
1074 80                 ADD     B
1075 47                 MOV     B,A     ;ADD 80 TO B IF SHIFT KEY IS PRESSED
1076 210011     SC45:   LXI     H,ASCII
```

```
1079 7D                 MOV     A,L
107A 80                 ADD     B
107B 6F                 MOV     L,A     ;PUT ADDRESS OF ASCII CODE IN H,L
107C 7E                 MOV     A,M     ;GET ASCII
107D 37         SC50:   STC             ;SET CARRY FOR TRUE RETURN
107E E1         SC99:   POP     H
107F D1                 POP     D
1080 C1                 POP     B
1081 C9                 RET

                        ;TEST1 ALLOWS ONE TO FIGURE OUT THE REQUIRED ENTRIES FOR
                        ;THE TABLE ASCII FOR ANY KEYBOARD

1082 210011     TEST1:  LXI     H,ASCII
1085 06A0               MVI     B,160
                        XRA     A
1088 77         T2:     MOV     M,A
1089 23                 INX     H
108A 3C                 INR     A
108B 05                 DCR     B
108C C28810             JNZ     T2
108F 3E98       T3:     MVI     A,98H
1091 D3F7               OUT     PCN
1093 AF                 XRA     A
1094 060A               MVI     B,10
1096 21A011             LXI     H,KEY
1099 77         T4:     MOV     M,A
109A 23                 INX     H
109B 05                 DCR     B
109C C29910             JNZ     T4      ;INIT.  TABLE KEY
109F CD1510     T5:     CALL    SCAN
10A2 D29F10             JNC     T5
10A5 CDC302             CALL    02C3H   ;CALL NMOUT
10A8 0E0D               MVI     C,0DH   ;PUT CR IN C
10AA CDF401             CALL    01F4H   ;CALL   ECHO
10AD C39F10             JMP     T5

                        ;ONCE THE TABLE ASCII HAS BEEN FILLED TEST2 ALLOWS ONE
                        ;TO TEST SCAN BEFORE INSTALLATION ON PROM

10B0 3E98       TEST2:  MVI     A,98H
10B2 D3F7               OUT     PCN
10B4 AF                 XRA     A
10B5 060A               MVI     B,10
10B7 21A011             LXI     H,KEY
10BA 77         T24:    MOV     M,A
10BB 23                 INX     H
10BC 05                 DCR     B
10BD C2BA10             JNZ     T24
10C0 CD1510     T25:    CALL    SCAN
```

```
10C3 D2C010             JNC     T25
10C6 4F                 MOV     C,A
10C7 CDE301             CALL    01E3H   ;CALL    CO
10CA C3C010             JMP     T25

1100                    ORG     1100H
1100            ASCII:  DS      160     ;THIS IS A TABLE OF ASCII CODES FOR EACH
                                        ;IT SHOULD BE IN RAM FOR DEBUGGING AND R
                                        ;FINAL INSTALLATION
11A0            KEY:    DS      10      ;KEY AND KEYN SHOULD BE IN RAM DURING BO
11AA            KEYN:   DS      10      ;AND DEBUGGING

0000                    END
```

Ref.# AB82

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004 ☐4040 ☐8008 ☒8080 ☐3000 ☐Other_____ (use additional sheets if necessary)

Program Title

P2708 PROM Programming Routine

Function

Programs 2708 PROMS with data from ISIS-11 Locater/Assembler. An 8K RAM area is loaded. Any data outside this area is ignored (not loaded). An offset parameter allows selection of user address to Load. Automatic test for erased PROM before programming. Range bell sounds at end of Programming. True/False data. Prints legal commands on request. Compare and Move (Transfer). Filename set at initial start-up or can be changed in Load command.

Required Hardware

MDS with 32K RAM, UPP, 2708 Personality Card

Required Software

ISIS-11 V2.2 or Greater

Input Parameters

Disk File with absolute object code--can be produced by locater or assembler (using aseg). (Will reject any file that is not Absolute.) See sample run.

REVISED 8/8/77

Output Results

Programmed and Verified 2708 PROMS. See sample run.

Paper tape not available.
Program offerred on Diskette only.
Diskette is priced at $35.00.

Registers Modified: All	Programmer: John E. Mahony
RAM Required: Non Resident	Company: Electra-Physics Labs Inc.
ROM Required: Non Resident	Address: 715 Sutter Street Mall
Maximum Subroutine Nesting Level: Non Resident	City: Folsom
Assembler/Compiler Used: ISIS-11 8080/8085 Assembler V1.0	State: California 95630

```
-P2708 :F1:P2708                        CALL PROGRAM AND SPECIFY FILE

2708 PROM PROGRAMMER, V2.0

:L3680                                  LOAD COMMAND

:LO                                     ATTEMPT TO LOAD WITH "WRONG" OFFSET
*** NO DATA LOADED ***

:L2000 XCMD.OBJ                         CHANGE OF FILE NAME

ERROR 13 USER PC 3A4C                   FILE NON-EXISTANT (ISIS-II ERROR)

:L2000 :F1:LCMD.OBJ                     ATTEMPT TO LOAD RELOCATABLE FILE
*** NOT AN ABSOLUTE FILE ***
*** NO DATA LOADED ***

:L3680 :F1:P2708                        LOAD ORIGINAL FILE

:PO                                     PROGRAM A RROM
INSERT PROM, TYPE CR TO CONTINUE

PROM NOT ERASED                         TESTS AUTOMATICALLY FOR ERASED PROM
INSERT PROM, TYPE CR TO CONTINUE        NEW PROM INSERTED; IT PROGRAMS & COMPARES OK
                                        LOWER CASE COMMANDS NOT RECOGNIZED
:e

UNRECOGNIZED COMMAND

:E                                      EXIT
-P2708                                  CALL PROGRAM WITHOUT FILE NAME

2708 PROM PROGRAMMER, V2.0

:L                                      DEFAULT OFFSET = 0  IF NONE SPECIFIED
ENTER FILE NAME: P2708                  REQUESTS A FILE NAME
*** NO DATA LOADED ***

:MO                                     MOVE A PROM INTO BLOCK 0
INSERT PROM, TYPE CR TO CONTINUE

:*FCBB
.S5000 31-00 1F-00 42-00 0E-01 04-02 11-03
.G                                      CHANGE A FEW RAM LOCATIONS

*

:CO                                     COMPARE PROM WITH RAM; CHANGES MADE ARE FOUND
INSERT PROM, TYPE CR TO CONTINUE
PROM ADDR=0000H  GOOD=00H BAD=31H
PROM ADDR=0001H  GOOD=00H BAD=1FH
PROM ADDR=0002H  GOOD=00H BAD=42H
PROM ADDR=0003H  GOOD=01H BAD=0EH
PROM ADDR=0004H  GOOD=02H BAD=04H
PROM ADDR=0005H  GOOD=03H BAD=11H

:X                                      PRINT OUT LEGAL COMMANDS
LEGAL COMMANDS ARE:

P  (PROGRAM PROM)
L  (LOAD DISK FILE INTO 8K BUFFER)
E  (IF PROMPT, RETURN TO ISIS)
   (IF IN A COMMAND, ABORT THE COMMAND)
T  (SET TRUE DATA) (DEFAULT VALUE)
F  (SET FALSE DATA)
C  (COMPARE PROM WITH BLOCK IN 8K BUFFER)
M  (MOVE PROM INTO BLOCK IN 8K BUFFER)
X  (FOR THIS LIST OF COMMANDS)

P, M, AND C REQUIRE A 1K BLOCK NUMBER FROM 0 TO 7.
L REQUIRES AN OFFSET PARAMETER THAT
SETS THE LOWEST USER ADDRESS IN THE 8K BUFFER.
DEFAULT VALUE=0, I.E. USER ADDRESSES FROM 0000H TO 1FFFH ARE LOADED.
WITH A VALUE OF 2000, ADDRESSES 2000H TO 3FFFH ARE LOADED, ETC.
L ALSO REQUIRES A FILENAME IF THE CURRENT FILE IS TO BE CHANGED,
OTHERWISE THE LAST NAME ENTERED WILL BE USED.
*** THE PROGRAM WILL REQUEST ANY PARAMETERS THAT IT REQUIRES.

:E
-                                       EXIT
```

Ref.# AB83

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 ☐ 8080 ☐ 3000 ☐ Other ISIS (use additional sheets if necessary)

Program Title

Data General to Intellec MDS Diskette Transport Package

Function

RDOS/ISIS IS A Data General computor resident diskette file maintenance package supporting Intel ISIS - format files on IBM compatible diskette drives.

Required Hardware

Data General computer, TTY console, 32-K memory, IBM compatible diskette drive and controller

Required Software

Data General RDOS, ALGOL, and assembler.
User - written diskette I/O driver

Input Parameters

1. Initialized ISIS diskettes.
2. User files to be copied to diskette.
3. Operator console commands

Output Results

1. Updated ISIS diskette.
2. User diskette files read from diskette
3. Diskette information:
   a. Sector Usage
   b. Directory Contents

Registers Modified: N/A	Programmer: James T. Reynolds
RAM Required: N/A	Company: SCI Systems, Inc.
ROM Required: N/A	Address: 8600 South Memorial Parkway
Maximum Subroutine Nesting Level: N/A	City: Huntsville
Assembler/Compiler Used: D 6 ALGOL & Assembler	State: ALABAMA 35802

6/8/77

Ref.# AB84

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 ☐ Other_____ (use additional sheets if necessary)

Program Title

Main routine DDUMP
Diskette DUMP routine

Function

Subroutines SCAN, VDUMP, MULT

The DDUMP routine dumps the content of the MDS diskette on the LIST device in both Hex and ASCII representation, and displays the error message (if any) on the CONSOLE device. The DDUMP is entered to the CONSOLE device, as a system command of the format DDUMP (:Fx:) (start track), (start sector), (end track), (end sector). (Refer to attached documents for details).

Required Hardware

MDS, MDS-DOS, CONSOLE device (CRT), LIST device (Line Printer)

Required Software

MDS monitor
ISIS-II

Input Parameters

DDUMP command parameters, (ISIS-II system call READ)

SCAN routine parameter block

DDUMP IOPB Block (10 bytes)

DDUMP Buffer for sector dump (128 bytes)

Output Results

Disk dump displayed on LIST device in Hex and ASCII representation, 16 bytes across.

Error message (if any) on the CONSOLE device.

Registers Modified:	Programmer: Raymond Sin
RAM Required:	Company: ITT Industries of Canada, Ltd. - SEL Division
ROM Required:	Address: P.O. Box 138, Toronto-Dominion Centre
Maximum Subroutine Nesting Level:	City: Toronto M5K 1H1
Assembler/Compiler Used:	State: Ontario

Ref.# AE8

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004 ☐4040 ☐8008 ☒8080 ☐3000 ☐Other_____ (use additional sheets if necessary)

Program Title

Cross Reference for PAS8Ø PASCAL programs - XREF8Ø

Function

XREF8Ø processes PAS8Ø source programs to give a cross reference listing of all identifiers and frequency tally of key words.

Required Hardware

Intellec MDS system with 64K bytes RAM memory, dual floppy disks, and CRT/TTY console.

Required Software

ISIS-II operating system and PAS8Ø resident PASCAL interpreter.

Input Parameters

The input parameters are the MDS file name of a PASCAL source program to be cross referenced and the destination file name of the cross reference listing.

Output Results

The output result is the specified destination file which has a line numbered listing of the PASCAL source program followed by the cross reference of identifiers by line number occurrence and the key word frequencies.

Description

XREF8Ø is run by loading the PAS8Ø system and then executing the XREF program (XREF8Ø is written in PASCAL). The following is an example of a sample entry with all users inputs underlined.

```
-PAS8Ø
PAS8Ø VMØ1-ØØ
*XREF(SOURCE.PAS,OUTPUT.XRF)
TERMINATED   AT LINE ØØ343
*
```

This sequence would create a file called OUTPUT.XRF which is the cross reference listing of the PASCAL source program SOURCE.PAS.

Registers Modified: ALL	Programmer: BOB EICHENLAUB
RAM Required: 64K BYTES	Company: JOHN FLUKE MFG. CO.INC.
ROM Required: NONE	Address: P. O. BOX 43210
Maximum Subroutine Nesting Level:	City: MTLK. TERRACE
Assembler/Compiler Used: PAS8Ø PASCAL COMPILER	State: WASHINGTON

Ref.# AE9

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☒ 8008 ☐ 8080 ☐ 3000 ☐ Other SBC 80 (use additional sheets if necessary)

Program Title

SBC 80/10 INTERACTIVE MONITOR

Function

Resident Interactive Monitor for the SBC 80/10, including the following operator commands: ABORT, DISPLAY, FILL, PROGRAM EXECUTE, INSERT, MOVE, READ PAPER TAPE, SUBSTITUTE, EXAMINE.

Required Hardware

SBC 80/10 Singel Board Computer, ASR-33 Tetetype Machine (or equivalent).

Required Software

(none)

Input Parameters

Keyboard commands are described fully in the listing.

Output Results

Some features of the Interactive Monitor are as follows: All commands are checked for validity before being executed. Paper tape input is buffered to allow checksum validation before being installed. The "Program Execute" command permits the setting and clearing of breakpoints. Provisions are made for a front-panel hardware interrupt switch. The entire Interactive Monitor occupies only 1024 bytes.

Program offered on diskette only.

Registers Modified: All	Programmer: Wayne Stahnke
RAM Required: 64 Bytes	Company: Wayne Stahnke Company
ROM Required: 1024 Bytes	Address: 2513 28th Street
Maximum Subroutine Nesting Level: 10	City: Santa Monica
Assembler/Compiler Used: 8080 MDS Macro Assembler, V2.2	State: California 90405

Ref.# AB85

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 ☐ Other_____ (use additional sheets if necessary)

Program Title

I/O Routine for TI Silent 700 Terminal

Function

Operate silent 700 on interrupt driven basis, and perform data transfer in blocks without supervision of main program or executive program

Required Hardware

8251 Programmable communication interface using TxRDY and RxRDY as interrupt requests

8214 Priority interrupt control unit using INT output to strobe interrupt vector into input port 8212

Required Software

Memory location 0038_H reserved for RST-7 branching point

Input Parameters

INPUT ROUTINE	OUTPUT ROUTINE
Entry from KSR 733	HL-Address of first output character B- Number of output characters Output buffer memory (ADR 1250-$129F_H$)

Output Results

Input buffer memory (ADR 1200-$124F_H$) containing input data block terminated with CR character

Character counter (ADR 12A0-$12A1_H$) Number of characters in input data block

Registers Modified: None	Programmer: Fred Lee
RAM Required: 6 + 160 bytes for I/O buffer	Company: University of California
ROM Required: 256 bytes to store routine & branch table	Address: Department of Pharmacology
Maximum Subroutine Nesting Level: 3	City: School of Medicine
Assembler/Compiler Used: 8080 Macro Assembler, Ver. 2.0	State: Los Angeles, CA 90024

insite™ INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/40 ☐8008 ☒8080 ☐8048 ☐8085 ☐3000 ☐Other _____ (use additional sheets if necessary)

Program Title

TTY DIAGNOSTIC

Function

The TTY characters are output from the CPU to the TTY punch and console verifying correct operation of the CPU to TTY path. The paper tape which is punched can be read into the CPU, ("Echoing" characters back to the TTY console), to verify the TTY to CPU Data Path.

Required Hardware

TTY on Port 0 & 1.

Required Software

8080 V3.0 Monitor (ORG 38ØØ)

Input Parameters

Origin of Punch Routine is 22ØØ H.
Origin of Read Routine is 221C H.

Output Results

The TTY Character set is output to the TTY Console and punch in the first portion of the program. Following a RESET and RESTART at 221C H the punched paper tape is read. The characters are output back to the TTY for comparison with the original data. The paper tape may also be verified in the local mode.

Registers Modified: A, B, C, D, Flags	**Programmer:** W. E. Barkman
RAM Required: NONE	**Company:** Union Carbide Corp.
ROM Required: 40_{10} words	**Address:** Box Y, Bldg. 9998, Stop 2
Maximum Subroutine Nesting Level: 1	**City:** Oak Ridge
Assembler/Compiler Used: 8080 V4.1	**State:** Tennessee 37830

```
                                                ;REF. NO. AA14
                                                ;PROGRAM TITLE TTY DIAGNOSTIC
                                                ;
                                                ;
                                                ;TTY INTERFACE DIAGNOSTIC
                                                ;CHARACTERS ARE OUTPUT FROM
                                                ;CPU AND PUNCHED ON TAPE
                                                ;CONTINUOUSLY.  READING THE
                                                ;TAPE ECHOS CHARACTERS BACK
                                                ;TO THE TTY.
2200                                    ORG   2200H
380C                                PO  EQU   380CH
3806                                RI  EQU   3806H
3809                                CO  EQU   3809H
2200 163A       START:  MVI   D,58      ;SET LOOP COUNTER
2202 0E21               MVI   C,21H     ;SET INITIAL CHARACTER
2204 CD0C38     NEXT:   CALL  PO
2207 0C                 INR   C
2208 15                 DCR   D
2209 C20422             JNZ   NEXT
220C 0E0A               MVI   C,0AH     ;LINE FEED
 20E CD0C38             CALL  PO
2211 0E0D               MVI   C,0DH     ;CARRIAGE RETURN
2213 CD0C38             CALL  PO
2216 C30022             JMP   START     ;PRESS RESET TO STOP
2219 00                 NOP
221A 00                 NOP
221B 00                 NOP
221C CD0638     NULL:   CALL  RI        ;READ CHARACTER
221F 4F                 MOV   C,A
2220 CD0938             CALL  CO        ;ECHO TO TTY
2223 C31C22             JMP   NULL
0000                    END
```

Ref.#AB86

insite™ INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/40 ☐8008 ☒8080 ☐8048 ☐8085 ☐3000 ☐Other _____ (use additional sheets if necessary)

Program Title

RECOVR

Function

The function is to find and recover to a diskette file data accidentally lost while using the ISIS text editors

Required Hardware

MDS 800 (32K RAM), Single density DOS, and CRT or TTY

Required Software

Standard ISIS I or ISIS II system diskette containing ISIS Text Editor V1.1 or V1.6

Input Parameters

See page 2 of attached procedure.

Output Results

A file on the diskette containing the text editor input buffer contents or text buffer contents as selected.

Registers Modified: All	Programmer: Ross E. Morgan
RAM Required: 32K	Company: Intel Corporation (ext. 2031)
ROM Required:	Address: 2880 Northwestern Parkway
Maximum Subroutine Nesting Level: Stack usage 6 Bytes	City: Santa Clara
Assembler/Compiler Used: ISIS 8080 Macro Assembler V1.0	State: California 95051

8/8/77

HOW TO USE THE RECOVERY PROGRAM

RECOVR is a program that will find and recover to a diskette file data accidentally lost while using the ISIS TEXT EDITORS.

The data can be lost in one of two places:

1. If you thought you were doing an insert (I) and you get a message:

   " " ILLEGAL IN THIS CONTEXT

   then the data is lost in the input buffer. To recover this data use the "I" switch in the command tail. It doesn't make any difference whether ISIS-I or ISIS-II is being used, when data is lost in the input buffer.

2. If you had data in the EDITOR and suddenly it is gone, or someone hits Interrupt 1, or you try to exit the EDITOR and get an ERROR 7 (full diskette); then the data is lost in the text buffer. Whether ISIS-I or ISIS-II is being used is important when recovering data from the text buffer. If ISIS-II was being used, RECOVR defaults to ISIS-II text buffer and no switch is necessary. If ISIS-I is being used, then the "1" switch must be set.

An ISIS-I based RECOVR can recover data from the ISIS-II buffers and visa-versa.

COMMAND SYNTAX: -RECOVR□filename [□switch]

"filename" is the name of a diskette file that is to contain the recovered data.

[ ] means optional. Omitting this portion will default RECOVER to the ISIS-II text buffer.

"switch" is either the letter "I" for the Text Editor input buffer or the number "1" if ISIS-I was used when the data was lost.

N O T E:

When recovering data from the input buffer the first twelve characters of the file will be permanently lost.

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. #AB87

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 (use additional sheets if necessary)

Program Title

MSAVE/MLOAD utilities for MDS-800 with DOS.

Function

MSAVE saves a block of RAM memory into an ISIS file.
MLOAD loads RAM memory with an ISIS file.

Required Hardware

MDS -800, DOS, 32K RAM memory

Required Software

MDS monitor V2.0, ISIS I or II

Input Parameters

MSAVE Filename$_S$ ↵
From? xxxx ↵
To? yyyy ↵

MLOAD Filename$_L$ ↵
At? zzzz ↵

Output Results

MSAVE creates Filename$_S$ with length yyyyH - xxxxH + 1

MLOAD places in memory Filename$_L$ at location zzzzH

Registers Modified: ALL	**Assembler/Compiler Used:** ISIS II 8080/8085 Assembler V1.0
RAM Required: 32K	**Programmer:** (317) 353 3249 Carl Harcourt, B/834
ROM Required: -	**Company:** NAVAL AVIONICS FACILITY, Indpls
Maximum Subroutine Nesting Level: 8/4	**Address:** 6000 East 21st Street Indianapolis, Indiana 46218

98-034C

Ref. #AB88

insite™ INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/40 ☐ 8008 ☒ 8080 ☐ 8048 ☐ 8085 ☐ 3000 ☐ Other _____ (use additional sheets if necessary)

Program Title

DISPLAY

Function

A set of general purpose routines for using a Video-memory display device.

Required Hardware

A Video-memory type of CRT display controller attached to the 8080 bus. The display should present one character for each byte of memory. It should display reverse video (black on white) for each character with bit 7 set.

Required Software

None.

Input Parameters

The variable CHR contains the character to be displayed by the routine DSWRT. All other routines require no parameters. The address and size of the video memory are Assembly-time variables.

Output Results

The video-memory will display the desired characters performing scrolling when necessary and allowing full cursor movement. The cursor is simulated by inverting bit 7 of certain memory locations to display reverse video.

Registers Modified: All	Programmer: Jeff Kravitz
RAM Required: 9 bytes + stack + video memory	Company:
ROM Required: 170 (hex) bytes	Address: 197 Fairhaven Blvd.
Maximum Subroutine Nesting Level: 5	City: Woodbury
Assembler/Compiler Used: 8080 Macro Assembler	State: New York 11797

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 ☐ Other_____ (use additional sheets if necessary)

Program Title

Hazeltine 2000 CRT Function Driver

Function

Provides an interface for all ten special functions of the Hazeltine 2000 CRT. See enclosed sheet for details.

Required Hardware

Hazeltine 2000 CRT

Required Software

None

Input Parameters

The user selects the following assembly parameters:
(a.) Output port
(b.) Register pair for (x,y) cursor addressing
(c.) The user may choose to assemble out any of the first seven functions on the enclosed sheet via an EQU to reduce memory needs.
(d.) Functions are invoked simply by CALL. The cursor addressing function expects a cursor position, (x,y), in the register pair selected in (b.) above.

Output Results

The given function is performed on the CRT.

Reference: Hazeltine 2000 Operating Manual
Hazeltine Corporation
Greenlawn, N.Y. 11740

Registers Modified: A only	Programmer: Dan Lasley
RAM Required: Zero (excluding stack)	Company: Memphis State University
ROM Required: 37 to 73 bytes	Address: Computer Services
Maximum Subroutine Nesting Level: One	City: Memphis
Assembler/Compiler Used: MAC 80 Vers 2.0, Xerox Sigma 9	State: TN 38111

```
                ;REF. NO. AB89
                ;PROGRAM TITLE 'HAZELTINE 2000 CRT FUNTION DRIVER'
                ;
                ;
                ;
                        TITLE   'HAZELTINE 2000 CRT FUNCTION DRIVER'
                ;
                ;
                ;       FUNCTION DRIVER FOR HAZELTINE 2000 CRT
                ;
                ;       WRITTEN BY DAN LASLEY
                ;       COMPUTER SERVICES
                ;       MEMPHIS STATE UNIVERSITY
                ;       MEMPHIS, TN 38152
                ;
                ;       BEGIN INVARIANT DATA...
                ;
                ;
0000            BPAIR   EQU     0               ;BC REGISTERS
0002            DPAIR   EQU     2               ;DE REGISTERS
0004            HPAIR   EQU     4               ;HL REGISTERS
                ;
                ;
                ;       BEGIN USER-SELECTABLE DATA:
                ;
                ;       (A)  SELECT CRT OUTPUT PORT...
                ;
0000            HAZEL   EQU     0               ;ARBITRARILY ZERO
                ;
                ;       (B)  SELECT REGISTER PAIR AS INPUT TO THE "ADCUR" (ADDR
                ;            CURSOR) ROUTINE.
                ;
0004            REG     EQU     HPAIR           ;ARBITRARILY "HL"
                ;
                ;       (C)  SELECTIVELY ENABLE THE FOLLOWING FUNCTIONS BY SETT
                ;            GIVEN FLAG NON-ZERO:
                ;
                ;            (1)  "ZXMIT"  -  TRANSMIT
                ;            (2)  "ZDELN"  -  DELETE LINE
                ;            (3)  "ZINLN"  -  INSERT LINE
                ;            (4)  "ZSBAC"  -  SET BACKGROUND
                ;            (5)  "ZCFOR"  -  CLEAR FOREGROUND
                ;            (6)  "ZSFOR"  -  SET FOREGROUND
                ;            (7)  "ZPRIN"  -  PRINT
                ;
0001            ZXMIT   EQU     1               ;INCLUDE ALL FUNCTIONS BY DE
 001            ZDELN   EQU     1
0001            ZSBAC   EQU     1
0001            ZINLN   EQU     1
0001            ZCFOR   EQU     1
```

```
0001            ZSFOR   EQU     1
0001            ZPRIN   EQU     1
                ;
                ;
                ;       SELECT A START ADDRESS FOR THE CODE.
                ;
1000            HERE    EQU     1000H           ;PUT IT ANY OLD WHERE
                ;
                ;
                ;       BEGIN CODE...
                ;
                ;
                ;
1000                    ORG     HERE
                ;
                ;
                ;*****************************************************************
                ;*      TRANSMIT FUNCTION....
                ;*****************************************************************
                        IF      ZXMIT           ;MAY BE SKIPPED...
1000 3E0E       XMIT:   MVI     A,14            ;14 IS CONTROL-N
 302 C32A10             JMP     DAR             ;"DUMP AND RETURN"
                        ENDIF
                ;
                ;
                ;*****************************************************************
                ;*      DELETE LINE FUNCTION...
                ;*****************************************************************
                ;
                ;
                        IF      ZDELN
1005 3E13       DELN:   MVI     A,19            ;CONTROL-S
1007 C33310             JMP     DAW             ;"DUMP AND WAIT"
                        ENDIF
                ;
                ;
                ;*****************************************************************
                ;*      SET BACKGROUND FUNCTION....
                ;*****************************************************************
                ;
                ;
                        IF      ZSBAC
100A 3E19       SBACK:  MVI     A,25            ;CONTROL-Y
100C C32A10             JMP     DAR
                        ENDIF
                ;
                ;
                ;*****************************************************************
                ;*      INSERT LINE FUNCTION....
                ;*****************************************************************
```

```
                        ;
                        ;
                                IF      ZINLN
100F 3E1A               INLN:   MVI     A,26            ;CONTROL-Z
1011 C33310                     JMP     DAW
                                ENDIF
                        ;
                        ;
                        ;*******************************************************************
                        ;*      CLEAR FOREGROUND FUNCTION...
                        ;*******************************************************************
                        ;
                        ;
                                IF      ZCFOR
1014 3E1D               CFORE:  MVI     A,29            ;CONTROL-SHIFT-M
1016 C33310                     JMP     DAW
                                ENDIF
                        ;
                        ;
                        ;*******************************************************************
                        ;*      SET FOREGROUND FUNCTION...
                        ;*******************************************************************
                        ;
                        ;
                                IF      ZSFOR
1019 3E1F               SFORE:  MVI     A,31            ;CONTROL-SHIFT-O
101B C32A10                     JMP     DAR
                                ENDIF
                        ;
                        ;
                        ;*******************************************************************
                        ;*      PRINT FUNCTION...
                        ;*******************************************************************
                        ;
                        ;
                                IF      ZPRIN
101E 3E1E               PRINT:  MVI     A,30            ;CONTROL-SHIFT-N
1020 C32A10                     JMP     DAR
                                ENDIF
                        ;
                        ;
                        ;*******************************************************************
                        ;*      CLEAR SCREEN FUNCTION...
                        ;*******************************************************************
                        ;
                        ;
 323 3E1C               CLEAR:  MVI     A,28            ;CONTROL-SHIFT-L
1025 C33310                     JMP     DAW
                        ;
                        ;
```

```
                ;*************************************************************
                ;*      HOME CURSOR FUNCTION...
                ;*      FALL INTO "DAR".
                ;*************************************************************
1028 3E12       HOME:   MVI     A,18            ;CONTROL-R
102A F5         DAR:    PUSH    PSW             ;SAVE CURRENT FUNCTION
102B 3E7E               MVI     A,126           ;DUMP LEAD-IN (CONTROL-SHIFT
102D D300               OUT     HAZEL           ;  PERIOD) TO CRT.
102F F1                 POP     PSW             ;RESTORE FUNCTION
1030 D300               OUT     HAZEL           ;PERFORM FUNCTION AND RETURN
1032 C9                 RET                     ;  CALLER WITHOUT DELAY.
                ;
                ;
                ;*************************************************************
                ;*      "DUMP AND WAIT" ROUTINE.
                ;*************************************************************
                ;
                ;
1033 CD2A10     DAW:    CALL    DAR             ;OUTPUT LEAD-IN AND FUNCTION
1036 3E7F               MVI     A,127           ;CERTAIN FUNCTIONS REQUIRE 0
1038 D300               OUT     HAZEL           ;  MORE DELAY CHARACTERS AT
 103A D300              OUT     HAZEL           ;  BAUD RATES ( >= 1200 BAUD
103C C9                 RET
                ;
                ;
                ;*************************************************************
                ;*      ADDRESS CURSOR FUNCTION.  LEFT REGISTER OF REGISTER PAI
                ;*      CONTAINS THE X (COLUMN) COORDINATE.  RIGHT REGISTER CON
                ;*      THE Y (ROW) COORDINATE.  NOTE THAT USER MAY SELECT ANY
                ;*      PAIRS BC, DE, OR HL VIA ASSEMBLY PARAMETERS.
                ;*************************************************************
                ;
                ;
103D 3E11       ADCUR:  MVI     A,17            ;CONTROL-Q
103F CD2A10             CALL    DAR             ;OUTPUT LEAD-IN AND CONTROL-
1042 7C                 MOV     A,REG           ;MOVE X COORDINATE
1043 D300               OUT     HAZEL           ;  TO OUTPUT PORT.
1045 7D                 MOV     A,REG+1         ;MOVE Y COORDINATE
1046 D300               OUT     HAZEL           ;  TO OUTPUT PORT.
1048 C9                 RET
0000                    END
```

Ref.#AB90

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004 ☐4040 ☐8008 ☒8080 ☐3000 ☐Other_____ (use additional sheets if necessary)

Program Title

Intel Format Hex Data File Load/Read

Function

Loads Hex files from paper tape, etc., into memory.

Required Hardware

8080 System. Paper Tape Reader.

Required Software

None

Input Parameters

Status bit for read data available is expected on Port 2, bit 7.
Data is read and echoed on Port 3.

Output Results

Data read is converted to binary and placed in memory beginning at address specified in beginning of record.

Registers Modified: A,B,D,H,L,'M'	Programmer: Gary Tock/Doug Aamold
RAM Required: 144 Bytes	Company: Micro Control Company
ROM Required: None	Address: 7956 Main Street, North East
Maximum Subroutine Nesting Level: 2	City: Minneapolis
Assembler/Compiler Used: ISIS 8080 Macro Assembler	State: Minnesota 55432

```
                ;REF. NO. AB90
                ;PROGRAM TITLE INTEL FORMAT HEX DATA FILE LOAD/READ
                ;
                ;
                ;
                ;
0000                    ORG     0000H


                        ;THIS IS A PROGRAM FOR READING INDUSTRY STANDARD
                        ;(INTEL) FORMAT HEX DATA
                        ;
                        ;THE 'READ DATA AVAILABLE' SIGNAL IS EXPECTED
                        ;ON PORT 2 BIT 7.  THE ASCII DATA IS EXPECTED
                        ;ON PORT 3.  THE DATA READ IS ALSO ECHOED ON
                        ;PORT 3.

0000 CD4200     LOOP1:  CALL    INCH            ;GET AND ASCII CHAR.  W/O PARITY.
0003 FE3A               CPI     ':'             ;BEGINNING OF RECORD INDICATOR?
0005 C20000             JNZ     LOOP1           ;KEEP LOOKING IF NOT
0008 CD2800             CALL    GETBT           ;GET A BINARY BYTE
                                                ;LENGTH FOR THE RECORD.

000B B7                 ORA     A               ;IF ZERO.  WE'RE DONE.
000C CA2700             JZ      EOF             ;END OF FILE.  HALT.
000F 47                 MOV     B,A             ;SAVE LENGTH INDICATOR.
0010 CD2800             CALL    GETBT           ;GET 1ST ADDRESS FOR RECORD.
0013 67                 MOV     H,A             ;MOST SIGNIFICANT BYTE
0014 CD2800             CALL    GETBT
0017 6F                 MOV     L,A             ;LEAST SIGNIFICANT BYTE
0018 CD2800             CALL    GETBT           ;SKIP RECORD TYPE INDICATOR.

001B CD2800     LOOP2:  CALL    GETBT           ;GET A BINARY BYTE OF DATA
001E 77                 MOV     M,A             ;STORE IN MEMORY.
001F 23                 INX     H               ;NEXT ADDRESS.
0020 05                 DCR     B               ;DECREMENT LENGTH INDICATOR.
0021 C21B00             JNZ     LOOP2           ;MORE BYTE THIS RECORD?
0024 C30000             JMP     LOOP1           ;GET NEXT RECORD.

0027 76         EOF:    HLT                     ;END OF FILE STOP

0028 CD3500     GETBT:  CALL    INDIG           ;GET A NUMBER
002B 87                 ADD     A               ;MOVE LOW 4 BITS
002C 87                 ADD     A               ;TO HIGH 4 BITS
002D 87                 ADD     A
002E 87                 ADD     A
 32F 57                 MOV     D,A             ;SAVE RESULT.
0030 CD3500             CALL    INDIG           ;GET ANOTHER NUMBER
0033 B2                 ORA     D               ;'ADD IN' TO FORM BINARY BYTE
0034 C9                 RET
```

```
0035 CD4200    INDIG:  CALL    INCH            ;GET A CHARACTER (NUMBER).
0038 FE3A              CPI     '9'+1           ;0-9 ?
003A FA3F00            JM      IND1            ;STRIP OFF UPPER BITS.
003D C609              ADI     9
003F E60F      IND1:   ANI     0FH             ;STRIP OFF UPPER 4 BITS.
0041 C9                RET
0042 DB02      INCH:   IN      2               ;GET STATUS.
0044 E680              ANI     80H             ;READ DATA AVAILABLE ?
0046 CA4200            JZ      INCH            ;KEEP LOOKING IF NOT READY?
0049 DB03              IN      3               ;GET A CHARACTER.
004B D303              OUT     3               ;ECHO CHAR.  TO PRINT DEVICE
004D E67F              ANI     7FH             ;STRIP OFF PARITY
004F C9                RET
0000           END
```

Ref. #AB91

insite™ INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/40 ☐ 8008 ☒ 8080 ☐ 8048 ☐ 8085 ☐ 3000 ☐ Other _____ (use additional sheets if necessary)

Program Title

"DATCON.B1" Analog to Digital Conversion Program

Function

The A/D Conversion Program represents the minimum software of Burr Brown's MP 8416 analog I/O system. It works in dialogue mode. The data conversion process begins after selection of the desired range. Each voltage data is represented by a 12 bit two's complement binary number.
The program converts this binary number in a real decimal number and outputs one value per line with sign and voltage specification. After acquisition of ten measurements the program ends or continues depending on user's input. During console input incorrect numbers are recognized and an error is announced. After input of "DATCON.B1" and carriage return the program starts under ISIS control.

Required Hardware

TTY, MDS 800, Floppy Disc, Analog I/O System MP8416 (Burr Brown)

Required Software

MDS 800 software and ISIS software (16K version is sufficient)

Input Parameters

The only necessary input parameters are the four numbers "1,2,3,4" and the two characters "Y and N"

Output Results

Output results are the commentary of the dialogue program and the converted voltage data.

Registers Modified: ./.	Programmer: Helmut Klie
RAM Required: 850 bytes	Company: Biomed. Technik, Medizinische Hochschule Hannover
ROM Required: ./.	Address: Nobelring 25
Maximum Subroutine Nesting Level: 3	City: 3000 Hannover 61
Assembler/Compiler Used: ISIS 8080 Assembler (16K version)	State: Federal Republic of Germany

Ref.#AB93

insite™ INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/40 ☐8008 ☒8080 ☐8048 ☐8085 ☐3000 ☐Other_____ (use additional sheets if necessary)

Program Title

FAST

Function

This program will reset the CRT USART to operate at four times the normal baud rate. This is done by using the divide by 4 rate instead of the divide by 16 which the monitor uses.

Required Hardware

MDS with 16K, Disk, CRT on ports 246 and 247

Required Software

MDS monitor, ISIS

Input Parameters

No input parameters. Called from ISIS

Eg.: -FAST

Output Results

A message will appear asking you to change the CRT Baud rate and type any character. At program completion, control is returned to ISIS.

Registers Modified: N/A	Programmer: Gerry de Koning
RAM Required: 108 BYTES	Company: Toronto Transit Commission
ROM Required: NONE	Address: 1900 Yonge Street
Maximum Subroutine Nesting Level: 1 call	City: Toronto
Assembler/Compiler Used: MDS Macro Assembler	State: Ontario M4S 1Z2

```
;;REF. NO. AB93
;PROGRAM TITLE FAST
;

;
;       PROGRAM:   FAST
;
;
;       PROGRAMMER:   GERRY DE KONING
;                     C. I. S.  PROJECT
;                     TORONTO TRANSIT COMMISSION
;
;
;       ENVIRONMENT:  HARDWARE:   INTELLEC MDS WITH CRT CONSOLE A
;                                 DISK SYSTEM.
;                     SOFTWARE:   MDS MONITOR V2.0
;                                 ISIS V1.2
;
;
;       DESCRIPTION:  THIS PROGRAM WILL SWITCH THE MDS INTERFACE
;                     TO FOUR TIMES THE DEFAULT SPEED.  A MESSAG
;                     THE USER TO SWITCH TO CRT TO THE FAST BAUD
;                     AFTER THIS IS DONE, THE USER ENTERS A CHAR
;                     ON THE CONSOLE INPUT DEVICE.  THE PROGRAM W
;                     RETURN TO ISIS.
;
;
;
```

```
                        ;
                        ;       EQUATES:
                        ;
00F7                    CRTCONTROL EQU  0F7H
00F6                    CRTDATA EQU     0F6H
0040                    ISIS    EQU     40H
F809                    CO      EQU     0F809H
F803                    CI      EQU     0F803H
                        ;
                        ;
                        ;       MAIN PROGRAM
                        ;
                        ;
3300                            ORG     3300H
3300 0628               FAST:   MVI     B,MSGLEN        ;OUTPUT MESSAGE
3302 213F33                     LXI     H,MSG
3305 4E                 FAST0:  MOV     C,M
3306 CD09F8                     CALL    CO
3309 23                         INX     H
330A 05                         DCR     B
 30B C20533                     JNZ     FAST0
                        ;
330E 21FFFF             DELAY:  LXI     H,0FFFFH        ;DELAY SO THAT LF CAN PRINT
3311 110100                     LXI     D,1
3314 2B                 DLY0:   DCX     H
3315 19                         DAD     D
3316 2B                         DCX     H
3317 D21433                     JNC     DLY0
                        ;
331A F3                         DI
331B AF                         XRA     A               ;CLEAR CRT USART
331C D3F7                       OUT     CRTCONTROL
331E D3F7                       OUT     CRTCONTROL
3320 D3F7                       OUT     CRTCONTROL
3322 3E40                       MVI     A,40H           ;RESET USART
3324 D3F7                       OUT     CRTCONTROL
3326 3E4E                       MVI     A,4EH           ;USART MODE:
                                                        ;   ASYNC  *16
                                                        ;   8 BITS
                                                        ;   NO PARITY
                                                        ;   ONE STOP BIT
3328 D3F7                       OUT     CRTCONTROL
332A 3E37                       MVI     A,37H           ;USART COMMAND:
                                                        ;   RTS, ERROR RESET, R*EN, DTR,
332C D3F7                       OUT     CRTCONTROL
 32E FB                         EI
 32F DBF6                       IN      CRTDATA         ;READ TO CLEAR DATA
                        ;
3331 CD03F8                     CALL    CI              ;WAIT TILL CHARACTER HIT.
```

```
3334 0E09              MVI     C,9             ;EXIT TO ISIS
3336 113733            LXI     D,$+1
3339 CD4000            CALL    ISIS
333C C30800            JMP     8               ;ISIS RESTART IF UNSUCCESSFUL EX
                       ;
333F 53455420 MSG:     DB      'SET CRT RATE TO FAST AND HIT CHARACTER',13,10
3343 43525420
3347 52415445
334B 20544F20
334F 46415354
3353 20414E44
3357 20484954
335B 20434841
335F 52414354
3363 45520D0A
0028           MSGLEN  EQU     $-MSG
3300                   END     FAST
```

insite™ INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/40 ☐ 8008 ☒ 8080 ☐ 8048 ☐ 8085 ☐ 3000 ☐ Other _____ (use additional sheets if necessary)

Program Title

SLOW

Function

SLOW is the undoing of FAST. If the CRT baud rate has been changed, SLOW will restore the CRT USART to its default speed.

Required Hardware

MDS with 16K, Disk, CRT on ports 246 and 247

Required Software

MDS Monitor, ISIS

Input Parameters

No input parameters; called from ISIS

Eg.: -SLOW

Output Results

A message will appear asking you to change the CRT baud rate and type any character. At program completion, control is returned to ISIS.

Registers Modified: N/A	Programmer: Gerry de Koning
RAM Required: 108 BYTES	Company: Toronto Transit Commission
ROM Required: NONE	Address: 1900 Yonge Street
Maximum Subroutine Nesting Level: 1 call	City: Toronto
Assembler/Compiler Used: MDS Macro Assembler	State: Ontario M4S 1Z2

```
;;REF. NO. AB93A
;PROGRAM TITLE SLOW

;
;       PROGRAM:  SLOW
;
;
;       PROGRAMMER:  GERRY DE KONING
;                    C. I. S.  PROJECT
;                    TORONTO TRANSIT COMMISSION
;
;
;       ENVIRONMENT:  HARDWARE:  INTELLEC MDS WITH CRT CONSOLE A
;                                DISK SYSTEM.
;                     SOFTWARE:  MDS MONITOR V2.0
;                                ISIS V1.2
;
;
;       DESCRIPTION:  THIS PROGRAM WILL SWITCH THE MDS INTERFACE
;                     TO THE DEFAULT BAUD RATE.  A MESSAGE WILL
;                     USER TO SET THE CRT TO THE SLOW BAUD RATE.
;                     THIS IS DONE, THE USER ENTERS A CHARACTER
;                     CONSOLE INPUT DEVICE, AND THE PROGRAM RETU
;                     ISIS.
;
;
;
```

```
                        ;
                        ;       EQUATES:
                        ;
00F7                    CRTCONTROL EQU  0F7H
00F6                    CRTDATA EQU     0F6H
0040                    ISIS    EQU     40H
F803                    CI      EQU     0F803H
F809                    CO      EQU     0F809H
                        ;
                        ;
                        ;       MAIN PROGRAM
                        ;
                        ;
3300                            ORG     3300H
3300 0628               SLOW:   MVI     B,MSGLEN        ;OUTPUT MESSAGE
3302 213F33                     LXI     H,MSG
3305 4E                 SLOW0:  MOV     C,M
3306 CD09F8                     CALL    CO
3309 23                         INX     H
330A 05                         DCR     B
 30B C20533                     JNZ     SLOW0
                        ;
330E 21FFFF             DELAY:  LXI     H,0FFFFH        ;DELAY SO THAT LF CAN PRINT
3311 110100                     LXI     D,1
3314 2B                 DLY0:   DCX     H
3315 19                         DAD     D
3316 2B                         DCX     H
3317 D21433                     JNC     DLY0
                        ;
331A F3                         DI
331B AF                         XRA     A               ;CLEAR CRT USART
331C D3F7                       OUT     CRTCONTROL
331E D3F7                       OUT     CRTCONTROL
3320 D3F7                       OUT     CRTCONTROL
3322 3E40                       MVI     A,40H           ;RESET USART
3324 D3F7                       OUT     CRTCONTROL
3326 3E4F                       MVI     A,4FH           ;USART MODE:
                                                        ;  ASYNC  *64
                                                        ;  8 BITS
                                                        ;  NO PARITY
                                                        ;  ONE STOP BIT
3328 D3F7                       OUT     CRTCONTROL
332A 3E37                       MVI     A,37H           ;USART COMMAND:
                                                        ;  RTS, ERROR RESET, R*EN, DTR,
332C D3F7                       OUT     CRTCONTROL
 32E FB                         EI
332F DBF6                       IN      CRTDATA         ;READ TO CLEAR DATA
                        ;
3331 CD03F8                     CALL    CI              ;WAIT TILL CHARACTER ENTERED.
```

```
3334 0E09            MVI     C,9             ;EXIT TO ISIS
3336 113733          LXI     D,$+1
3339 CD4000          CALL    ISIS
333C C30800          JMP     8               ;ISIS RESTART IF UNSUCCESSFUL EX
                     ;
333F 53455420 MSG:   DB      'SET CRT RATE TO SLOW AND HIT CHARACTER',13,10
3343 43525420
3347 52415445
334B 20544F20
334F 534C4F57
3353 20414E44
3357 20484954
335B 20434841
335F 52414354
3363 45520D0A
0028            MSGLEN  EQU     $-MSG
3300                    END     SLOW
```

Ref. #AB94

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 ☐ Other _____ (use additional sheets if necessary)

Program Title

5 LEVEL (BARDOT) TO 8 LEVEL (ASCII) PAPER TAPE CONVERSION

Function

CONVERTS 5 LEVEL BARDOT PAPER TAPES FROM TELEX, TWX, OR AMATEUR RTTY TO ASCII.

Required Hardware

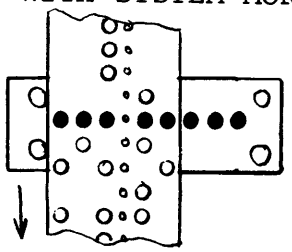
INTELLEC 8/MOD 80 WITH ASR 33 TTY ON PORTS 0 AND 1.

Required Software

SYSTEM MONITOR VERSION 3.0

Input Parameters

BARDOT TAPE IS PLACED IN TTY READER AS SHOWN BELOW.
READER = 1 IS ASSIGNED WITH SYSTEM MONITOR.



Output Results

WHEN READER INPUT ROUTINE IS CALLED THE PROPER ASCII CHARACTER IS RETURNED IN THE ACCUMULATOR.

Registers Modified: A,F (HL RESTORED)	Programmer: MARK D. HANSEN
RAM Required:	Company: INSTRUMENTATION SPECIALTIES
ROM Required: 90 HEX BYTES	Address: P.O. BOX 5347
Maximum Subroutine Nesting Level:	City: LINCOLN, NEBRASKA 68505
Assembler/Compiler Used: version 3.0	State:

```
                ;REF. NO. AB94
                ;PROGRAM TITLE 5 LEVEL (BARDOT) TO 8 LEVEL (ASCII) PAPER TAPE CO
                ;
                ;
                ;
                ;
                ;
                ;       5 TO 8 LEVEL PAPER TAPE CONVERSION PROGRAM
                ;
                ;       MARK D. HANSEN          5/28/77
                ;
                ;       REQUIRES INTELLEC 8/MOD 80
                ;       SYSTEM MONITOR VERSION 3.0
                ;
                ;       PLACE 5 LEVEL (BARDOT) TAPE IN ASR 33
                ;       READER WITH THE TAPE COVERING THE LEFT
                ;       5 READER PINS.
                ;
                ;       READER ASSIGNMENT = 1 FOR CHARACTERS
                ;                         = 2 FOR FIGURES
                ;
                ;       USE MONITOR COMMAND  "AR=1" TO CHANGE TO
                ;       BARDOT MODE
                ;
                ;       READER JUMP VECTORS ARE AT 3706H (R=1)
                ;       AND 3709H (R=2)
                ;
3706             ORG 3706H
3706 C31B37      JMP LTRIN
3709 C33437      JMP FIGIN
                ;
371B                 ORG  371BH
                LTRIN:
371B CD6137      CALL FAKE          ;TTY READER IN
371E D8          RC                 ;RETURN IF NO TAPE
371F E61F        ANI 00011111B      ;MASK OFF LOWER 5 BITS
3721 FE1B        CPI FIGS           ;COMPARE WITH FIGURES CHARACTER
3723 CA4B37      JZ FLIP            ;IF FIGURES CHANGE TO READER=2
3726 FE1F        CPI LTRS           ;IF LETTERS CHARACTER
3728 CA1B37      JZ LTRIN           ;IGNORE AND GET ANOTHER CHARACTER
372B E5          PUSH H
372C 216537      LXI H,TABLTR       ;POINT TO START OF CONVERSION TABLE
                FIN:
372F 85          ADD L              ;ADD INPUT CHARACTER TO POINT TO
3730 6F          MOV L,A            ;PROPER TABLE ENTRY
 3731 7E         MOV A,M            ;GET PROPER ASCII CHARACTER
 3732 E1         POP H              ;RESTORE H&L REGISTERS
3733 C9          RET                ;RETURN WITH PROPER ASCII CHARACTER IN A
                ;
                FIGIN:
```

```
3734 CD6137     CALL FAKE           ;GET READER INPUT ROM TTY
3737 D8         RC                  ;RETURN IF NO TAPE LEFT
3738 E61F       ANI 00011111B       ;MASK OFF LOWER 5 BITS
373A FE1F       CPI LTRS            ;COMPARE WITH LETTERS CODE AND
373C CA5637     JZ FLOP             ;SWITCH TO LETTERS MODE (R=1)
373F FE1B       CPI FIGS            ;IF FIGURES CHARACTER IGNORE
3741 CA3437     JZ FIGIN            ;AND GET ANOTHER
3744 E5         PUSH H
3745 218537     LXI H, TABFIG       ;POINT TO FIGURES TABLE
3748 C32F37     JMP FIN             ;GO FINISH LIKE LETTERS
                ;
                FLIP:
374B 3A0300     LDA 3               ;SWITCH FROM LETTERS TO FIGURES MODE
374E F604       ORI 00000100B       ;BY CHANGING THE I/O STATUS BYTE
3750 320300     STA 3               ;FROM READER=1 TO READER=2
3753 C33437     JMP FIGIN
                ;
                FLOP:
3756 3A0300     LDA 3               ;SWITCH FROM FIGURES TO LETTERS MODE
3759 E6FB       ANI 11111011B       ;BY CHANGING THE I/O STATUS BYTE
 75B 320300     STA 3               ;FROM READER=2 TO READER=1
375E C31B37     JMP LTRIN
                ;
                FAKE:
3761 E5         PUSH H              ;PUSH DOWN ONE LEVEL OF STACK TO
                                    ;COMPENSATE FOR POP AT END
                                    ;OF TTY MONITOR ROUTINE
3762 C3AF3E     JMP 3EAFH           ;TTY READER ROUTINE IN MONITOR
                ;
                ;         LETTERS MODE LOOKUP TABLE
                ;
                TABLTR:
3765 00          DB 0     ;BLANK
3766 54          DB 54H   ;T
3767 0D          DB 0DH   ;CR
3768 4F          DB 4FH   ;O
3769 20          DB 20H   ;SPACE
376A 48          DB 48H   ;H
376B 4E          DB 4EH   ;N
376C 4D          DB 4DH   ;M
376D 0A          DB 0AH   ;LF
376E 4C          DB 4CH   ;L
376F 52          DB 52H   ;R
3770 47          DB 47H   ;G
3771 49          DB 49H   ;I
 772 50          DB 50H   ;P
3773 43          DB 43H   ;C
3774 56          DB 56H   ;V
3775 45          DB 45H   ;E
3776 5A          DB 5AH   ;Z
```

```
3777 44          DB 44H  ;D
3778 42          DB 42H  ;B
3779 53          DB 53H  ;S
377A 59          DB 59H  ;Y
377B 46          DB 46H  ;F
377C 58          DB 58H  ;X
377D 41          DB 41H  ;A
377E 57          DB 57H  ;W
377F 4A          DB 4AH  ;J
3780 00          DB 00H  ;FIGS
3781 55          DB 55H  ;U
3782 51          DB 51H  ;Q
3783 4B          DB 4BH  ;K
3784 00          DB 00H  ;LTRS
                 ;
                 ;          FIGURES MODE LOOKUP TABLE
                 ;
                 TABFIG:
3785 00          DB 00H   ;BLANK
3786 35          DB 35H   ;5
 787 0D          DB 0DH   ;CR
3788 39          DB 39H   ;9
3789 20          DB 20H   ;SPACE
378A 23          DB 23H   ;#
378B 2C          DB 2CH   ;,
378C 2E          DB 2EH   ;.
378D 0A          DB 0AH   ;LF
378E 29          DB 29H   ;)
378F 34          DB 34H   ;4
3790 26          DB 26H   ;&
3791 38          DB 38H   ;8
3792 30          DB 30H   ;0
3793 3A          DB 3AH   ;:
3794 3B          DB 3BH   ;;
3795 33          DB 33H   ;3
3796 22          DB 22H   ;"
3797 24          DB 24H   ;$
3798 3F          DB 3FH   ;?
3799 07          DB 07H   ;BELL
379A 36          DB 36H   ;6
379B 21          DB 21H   ;!
379C 2F          DB 2FH   ;/
379D 2D          DB 2DH   ;-
379E 32          DB 32H   ;2
379F 27          DB 27H   ;'
 7A0 00          DB 00H   ;FIG
 7A1 37          DB 37H   ;7
37A2 31          DB 31H   ;1
37A3 28          DB 28H   ;(
37A4 00          DB 00H   ;LTRS
```

```
                ;
001B            FIGS EQU 00011011B
001F            LTRS EQU 00011111B
                ;
0000              END
```

Ref. #AA15

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/4040 ☒ 8008 ☐ 8080 ☐ 8048 ☐ 8085 ☒ Other SDK-80 or SBC 80/10 (use additional sheets if necessary)

Program Title

Sample Automatic Test Equipment Program

Function

Test program for a single board microcomputer-based Automatic Test Equipment to test digital circuits up to 24 I/O lines. When more memory is available the system can easily be expanded to test circuits up to 156 I/O lines. In that case, an MDS microcomputer system with diskett is advisable. Then the test data for large circuits can be stored on a diskett. (For more information refer to attached thesis).

Required Hardware

TTY or CRT, special ATE interface circuitary which costs less than $2.00 per I/O line. This interface can be mounted on the unused space of the SDK-80 single board for 24 lines of I/O.

Required Software

SKD-80 or SBC 80/10 monitor.

Input Parameters

1. Tests to be applied to the circuit under test written in a special format. This data can be stored into the RAM via keyboard or can be stored on a ROM for the circuits more frequently tested. A high level language program is in progress to provide the test data in the required format.
2. Commands to the test program (test data address and repetition paramter). For more information refer to Chap. IV of my thesis (ATE user's instructions).

Output Results

Test results will be written on console. If only the detection of faults requested the output message will indicate whether the circuit is all right or faulty (4000 tests/sec. less memory requirement for test data). If the location of the fault is then appropriate, diagnostic messages will be written on console

Ideal to test SSI, MSI integrated circuits. Can be used to test IC's in digital labs and for educational purposes. Single board ATE can be constructed with approximate hardware cost of $300.00

Registers Modified: ALL	Programmer: Hamid T. Hashemi
RAM Required: 129 bytes minimum	Company: Computer Science Dept.
ROM Required: 1K bytes, C2708 or equivl.	Address: SUNY at Potsdam, N.Y.
Maximum Subroutine Nesting Level: 3	City: Potsdam,
Assembler/Compiler Used: None	State: New York 13676

Ref. #AB95

insite™ INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/40 ☐ 8008 ☒ 8080 ☐ 8048 ☐ 8085 ☐ 3000 ☒ Other SBC80/10 (use additional sheets if necessary)

Program Title

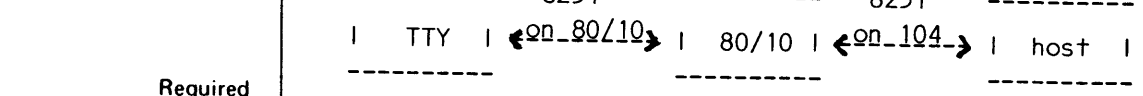
SBC COMMUNICATOR

Function

This routine runs in an Intel SBC 80/10(20) and provides the ability to load and execute programs from 1) the terminal (TTY) connected to the 80/10 board USART or 2) from any host CPU connected to the USART on an Intel SBC 104 board. It also allows the microcomputer board(s) to become transparent so that regular terminal communication with the host CPU is possible.

Required Hardware

Intel SBC 80/10 and SBC 104



Required Software

None if program is allowed to reside at location 00H
SBC 80P MONITOR is useful.

Input Parameters

Right bracket followed by L indicates load stream coming (ASC11 5DH)
Right bracket followed by S indicates start address coming
any other characters get passed directly to the other serial stream (from TTY to host or from host to TTY)

Output Results

Host communication or
loaded file in the SBC memory or
started SBC program.

Registers Modified: ALL	Programmer: Rex Tracy
RAM Required: 2 loc. + stack	Company: Colorado State University
ROM Required: 18BH locations + SBC Monitor if used	Address: Elec. Engr. Dept. - CSU
Maximum Subroutine Nesting Level: ?	City: Ft. Collins
Assembler/Compiler Used: PDP11 Cross Assembler	State: Colorado 80523

Ref. #AB96

insite™ INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/40 ☐ 8008 ☒ 8080 ☐ 8048 ☐ 8085 ☐ 3000 ☐ Other _____ (use additional sheets if necessary)

Program Title

IBM SELECTRIC INPUT PROGRAM

Function

THE PROGRAM ENABLES SELECTRIC TO BE USED AS AN INPUT DEVICE AND OPTIONALLY PRINT ON TELETYPE

Required Hardware

INTELLEC 8/MOD 80 SYSTEM, IBM SELECTRIC 735, SELECTRIC INPUT INTERFACE AS PER ATTACHED CIRCUIT DIAGRAM, TELETYPE ON PORT 0 AND 1 WITH TAPE PUNCH IF DESIRED.

Required Software

8080 MONITOR VER. 1.0

Input Parameters

SELECTRIC INPUT DATA READ FROM PORTS 20H AND 21H INTO ACCUMULATOR

Output Results

CONVERTED ASC11 CHARACTER IN REGISTER C AND PRINTED/PUNCHED ON TTY.

Registers Modified: A, B, C, H & L	Programmer: D. M. VAIDYA
RAM Required: NONE	Company: WESTFIELD COLLEGE
ROM Required: 328 BYTES	Address: KIDDERPORE AVENUE
Maximum Subroutine Nesting Level: -	City: LONDON, NW3 7ST
Assembler/Compiler Used: CROSS ASSEMBLER	State: ENGLAND

Ref. #AB98

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/4040 ☐ 8008 ☒ 8080 ☐ 8048 ☐ 8085 ☐ Other _____ (use additional sheets if necessary)

Program Title

INSERT TAB CHARACTERS FOR SPACES

Function

A tab (CTRL-I) is substituted for each group of 8 spaces or less. Single spaces are left as they are.

Required Hardware

Intellec MDS and appropriate input and output devices.

Required Software

ISIS diskette Operating system

Input Parameters

Provide input file and output file when program is called.

ie: SPTAB File 1 to File 2

File 1 and File 2 can be any legal file (:TR:, :CI:, :F1:PCM, SRC, etc)

Output Results

Only single spaces are left in the output file; all other groups of spaces are changed to an appropriate number of TAB characters.

Registers Modified:	Programmer: John Goode
RAM Required: 64K (less with smaller buffers)	Company: Dow Chemical USA
ROM Required:	Address: P.O. Drawer K Bldg. B-1219
Maximum Subroutine Nesting Level:	City: Freeport
Assembler/Compiler Used: PLM80 V3.0	State: Texas 77541

Ref.#AB99

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004 ☐4040 ☐8008 ☒8080 ☐3000 ☐Other_____ (use additional sheets if necessary)

Program Title

PROM PROGRAMMER FOR SDK

Function

Programmes 27o8 EPROM with data from RAM/ROM area on SDK board. Number of progamming pulses is set to 5 times needed number of pulses to change the data in all addresses. Pouls 1 ms with one wait state.

Required Hardware

INTEL MCS-8o SDK, ASR-3 TTY
Interface circuit for EPROM, see diagram.

Required Software

SDK MONITOR PROM

Input Parameters

Start address, end address (RAM/ROM area), start address in EPROM.

Output Results

Instructions: 1) Write data into RAM area (I command) or place ROM to be duplicated into normal ROM socket on SDK board.

2) Type: G12oo (CR)
Output: MON PROM SEE PROC ADRS C/R:

3) 26V and PPWR off, place EPROM into socket, PPWR on.
Type addresses: xxxx,yyyy.zzzz (CR)
Output: ALL ONE (EPROM erased in area)
26V ON C/R

4) 26V on , type: (CR)
Wait for programming min 5 sec max 256 sec
Output: XX (number (HEX) of used progamming pulses)
MON PROM SEE PROC ADRS C/R:

cont. next page.

Registers Modified: All	Programmer: Helge Lassen
RAM Required: 379 byte	Company: Soenderborg tekniske skole
ROM Required: SDK MONITOR PROM	Address: Skovvej 26
Maximum Subroutine Nesting Level: NA	City: Soenderborg
Assembler/Compiler Used: ISIS II 8o8o/8o85 ASSM., V1.o	State: Denmark

5) Repeat from step 3 (with new addresses or EPROM)
or return to monior type: not hex char.

CAUTION: It is only allowed to shift EPROM with kit RESET or when program type MON PROM and 26V and PPWR is off .

Error type out:

NOT ALL ONE : EPROM not erased in area

ADR OVERFLOW : RAM/ROM area greater than 1K or area not placed correct in EPROM.

BAD ROM : More than 256 pulses needed for programming or not ok after last programming puls.

Ref. #AE10

insite™ INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/40 ☐8008 ☐8080 ☐8048 ☐8085 ☐3000 ☒Other SDK80 (use additional sheets if necessary)

Program Title

SDK80 TRAP

Function

Inserts a branch to SDK80 monitor from the user program and a return path to the original program without loss of program flow. Allows all monitor facilities to be used to debug a program in RAM or inspect program operation.

Required Hardware

SDK80 with monitor routines.

Required Software

Program uses trap buffer memory 1000 to 102F if only 3 traps are used, leaving locations 1030 to 11FF for user programs. If extra memory is available the trap buffer origin may be shifted by altering locations 1240, 128F and 12A5 in the listing. The number of traps able to be used, up to 16, is determined by available buffer memory, requiring 16 (decimal) locations per trap.

Input Parameters

1. To insert a trap in a user program, return to monitor (RESET) and type .G12ØØ (CR). Next type IJ,XXXX where I is the trap identifier, J is the range (either 3, 4 or 5) of the skipped code, and XXXX is the address where the trap is to be inserted.

e.g. for the following program segment:

LOC	CODE	MNEMONIC
183D	3E3C	MVIA,3C
183F	CDCDAB	CALL ABCD

a) to insert trap number 2 at 183D type 25, 183D
b) to insert trap number 1 at 183F type 13, 183F

Output Results

Note that J must be 3, 4 or 5 and must completely span a program statement (or statements).

2. To remove trap I, enter the trap routine as above and type I (CR). Where I is the trap identifier number to be removed.

3. To return to the user program from the monitor after entering a trap type G (CR) when the program will resume at the left of PC value (as for the SDK monitor) or G XXXX (CR) to resume from location XXXX.

When the trap point in the program is encountered TI is printed where I is the trap identifier, and control is passed to the SDK80 monitor.

Source Listing available for $15.00.
Paper Tape or Diskette Not Offered.

Registers Modified: All	Programmer: C G Brickell
RAM Required: 1K and Buffer	Company: Fisher & Paykel Ltd
ROM Required: SDK Monitor	Address: Private Bag
Maximum Subroutine Nesting Level:	City: Panmure
Assembler/Compiler Used:	State: ~~AUCKLAND~~ NEW ZEALAND

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. #AE11

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000

(use additional sheets if necessary)

Program Title

TRACE AND REGISTER PRINT OUT.

Function

TO PRINT OUT REGISTERS ON COMMAND OR TO STEP THROUGH A PROGRAM INSTRUCTION BY INSTRUCTION AND TO HAVE REGISTERS PRINTED OUT AFTER EVERY INSTRUCTION EXECUTED. IF TRACE PROGRAM IS ENDED LAST ADDRESS EXECUTED WILL BE DISPLAYED.

Required Hardware

TTY --- STANDARD INTEL SETUP.

Required Software

NONE.

Input Parameters

NONE.

Output Results

REGISTER HEADING AND CONTENTS.

Registers Modified: NONE.	**Assembler/Compiler Used:** INTELLEC MDS 800
RAM Required: 275H RAM LOCATIONS.	**Programmer:** R. A. POYNER
ROM Required: NONE.	**Company:** NAVAL OCEAN SYSTEMS CENTER
Maximum Subroutine Nesting Level: NONE.	**Address:** PT. LOMA SAN DIEGO, CALIF. 92152

98-034C

Ref. #AB100

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 ☒8080 ☐8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

EXEC

Function

To provide character oriented access to ISIS files, and a 'clean' return to ISIS.

Required Hardware

MDS with at least 18K RAM, disk system, and console

Required Software

MDS monitor and ISIS V1.2

Input Parameters

Output Results

Registers Modified:	Programmer: Gerry de Koning
RAM Required: initially to 4600H, then to 39C2	Company: Toronto Transit Commission
ROM Required:	Address: 1900 Yonge Street
Maximum Subroutine Nesting Level:	City: Toronto, Ontario
Assembler/Compiler Used: ISIS 8080 MACRO ASSEMBLER V1.1	State: CANADA M4S 1Z2

Ref. #AB102

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/4040 ☐ 8008 ☒ 8080 ☐ 8048 ☐ 8085 ☐ Other _____ (use additional sheets if necessary)

Program Title

MDS BACK TO BACK DATA TRANSFER

Function

Enables data to be transmitted between two MDS systems using the high speed punch and reader interfaces

Required Hardware

Two MDS systems. One 7474 D flip flop

Required Software

PL/M 80

Input Parameters

The sending MDS system invokes the attached PLM program.
The receiving MDS system uses the copy command as follows:

```
COPY  :HR: TO filename
```

Output Results

A file is transferred

Registers Modified: ALL REGISTERS	Programmer: Bill Holmes
RAM Required: 8CH	Company: NASA
ROM Required: ADH	Address: Goddard Space Flight Center
Maximum Subroutine Nesting Level: 3	City: Code 533 Greenbelt
Assembler/Compiler Used: PL/M 80	State: Maryland 20771

```
ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE DPU
OBJECT MODULE PLACED IN :F1:AB102.OBJ
COMPILER INVOKED BY:  PLM80 :F1:AB102


                $PAGEWIDTH(80)

                /*COPY FROM DISK TO :HP: */

  1             DPU:
                    DO;
  2    1            DECLARE ROUTINE ADDRESS INITIAL(0FD5FH);
  3    1            DECLARE I BYTE;
  4    1            DECLARE NULL BYTE INITIAL(34H);
  5    1            DECLARE BUFFER(128) BYTE;
  6    1            DECLARE ACTUAL$COUNT ADDRESS;
  7    1            DECLARE STATUS ADDRESS;
  8    1            DECLARE AFT$IN ADDRESS;
  9    1            DECLARE READ$ACCESS LITERALLY '1';

 10    1            OPEN:
                        PROCEDURE(AFT,FILE,ACCESS,MODE,STATUS) EXTERNAL;
 11    2                DECLARE  (AFT,FILE,ACCESS,MODE,STATUS) ADDRESS;
 12    2            END OPEN;

 13    1            CLOSE:
                        PROCEDURE(AFT,STATUS) EXTERNAL;
  1    2                DECLARE  (AFT,STATUS) ADDRESS;
 15    2            END CLOSE;

 16    1            READ:
                        PROCEDURE(AFT,BUFFER,COUNT,ACTUAL,STATUS) EXTERNAL;
 17    2                DECLARE  (AFT,BUFFER,COUNT,ACTUAL,STATUS) ADDRESS;
 18    2            END READ;

 19    1            EXIT:
                        PROCEDURE EXTERNAL;
 20    2                DECLARE STATUS ADDRESS;
 21    2            END EXIT;

                /* GET FILE NAME AND OPEN */

 22    1            CALL READ(1,.BUFFER,128,.ACTUAL$COUNT,.STATUS);
 23    1            CALL OPEN(.AFT$IN,.BUFFER,READ$ACCESS,0,.STATUS);

                /* PUT OUT A DUMMY CHARACTER TO :HP: */
 24    1            CALL ROUTINE(NULL);

                /* COPY FILE TO :HP: */
 25    1            ACTUAL$COUNT=1;
 26    1            DO WHILE ACTUAL$COUNT <> 0;
 27    2                CALL READ(AFT$IN,.BUFFER,128,.ACTUAL$COUNT,.STATUS);
 28    2                IF ACTUAL$COUNT<>0 THEN
 29    2                DO I=0 TO ACTUAL$COUNT-1;
 30    3                CALL ROUTINE(BUFFER(I));
 31    3                END;
```

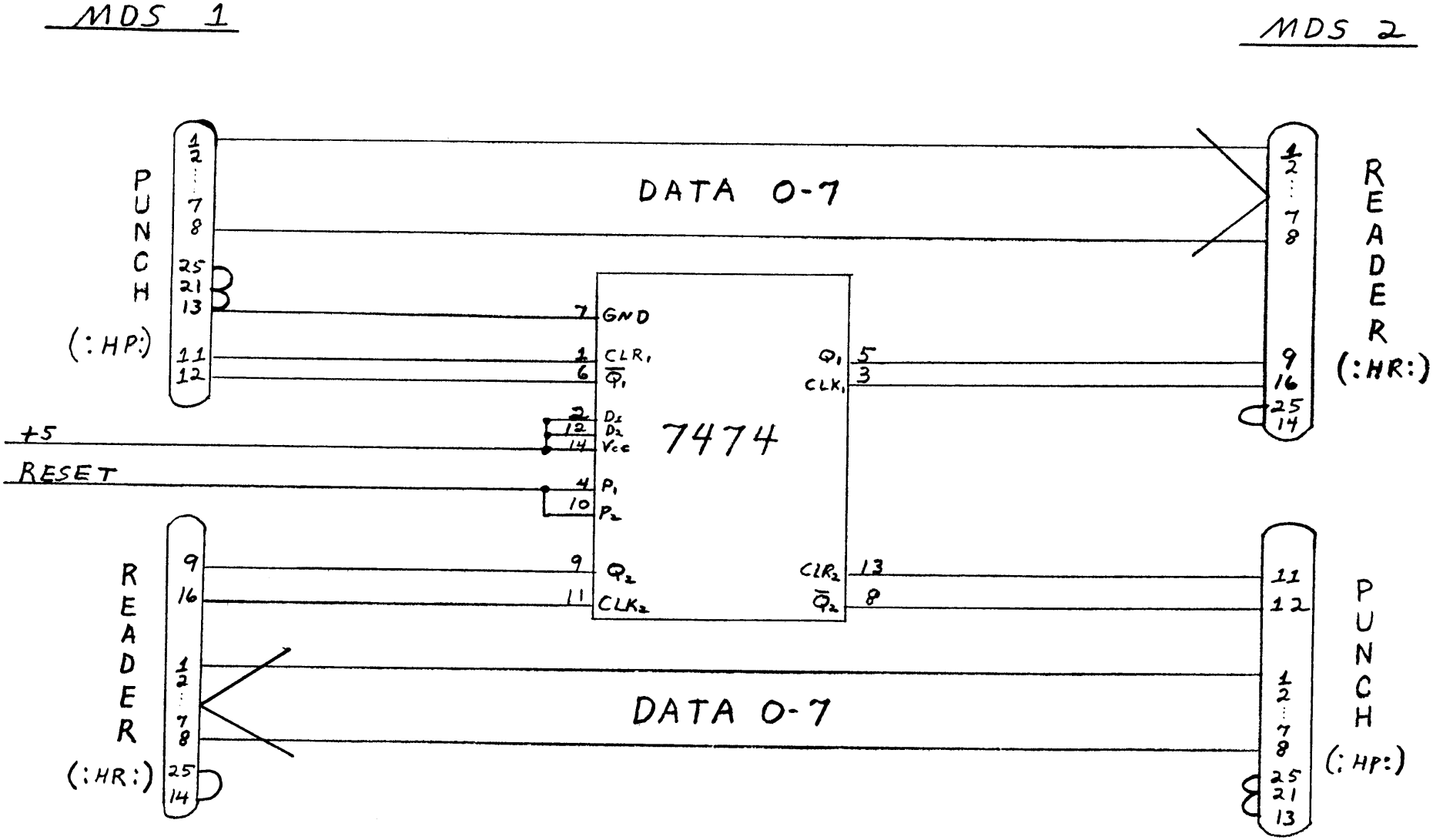
```
 32   2          END;

 33   1          CALL CLOSE(AFT$IN, STATUS);
 34   1          CALL EXIT;
 35   1          END DPU;
```

MODULE INFORMATION:

```
     CODE AREA SIZE     = 00ADH    173D
     VARIABLE AREA SIZE = 008CH    140D
     MAXIMUM STACK SIZE = 0008H      8D
     56 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```



MDS BACK TO BACK

Ref. #AB103

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/4040 ☐ 8008 ☒ 8080 ☐ 8048 ☐ 8085 ☐ Other_____ (use additional sheets if necessary)

Program Title

FDUMP

Function

FDUMP dumps an ISIS-II diskette file to another ISIS-II file in printable form: Hexadecimal, Octal and ASCII representations are included.

Required Hardware

MDS, MDS-DOS, Console Device

Required Software

MDS Monitor
ISIS-II

Input Parameters

FDUMP is called via a system command:

-FDUMP file to file

Output Results

Error message (if any) to console.
The output file is formatted as 72 byte lines (each followed by <CR> <LF>) containing Hexadecimal, Octal and ASCII interpretations of 8 bytes of the input line.

Registers Modified:	Programmer: Garth Eaglesfield
RAM Required:	Company: Micro Focus Ltd.
ROM Required:	Address: 18, Vernons Yard
Maximum Subroutine Nesting Level:	City: London W.11.
Assembler/Compiler Used:	State: ENGLAND

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

ENHANCED MDS TEXT EDITOR, X111

Function

Adds Powerful features to Intel's V1.6 Text Editor

Required Hardware

MDS-800 with disk operating system

Required Software

ISIS-II

Revised 12/78

Input Parameters

*Series-II - 220 Users! Please request special macro for use woth integrated drive.

Output Results

Program offered in Object Code on diskette only for $50.00.

Registers Modified:	Programmer:
RAM Required:	Company:
ROM Required:	Address:
Maximum Subroutine Nesting Level:	City:
Assembler/Compiler Used:	State:

The Enhanced Text Editor Maintains all the commands of Intel's current text editor (V1.6) with the addition of many new powerful features. Some of them are:

1) Auto append. Text is automatically read into the text buffer from the specified file.

2) Text rearranging. Blocks of text can be switched around.

3) Macros. Command strings can be assigned to single letter or single control key designations. These macros can be written to a file called EDIT.MAC which is automatically read in when the editor is called.

4) V-Markers. Eight different pointer positions can be designated to delineate the text area for various commands and block moves.

5) Value stack. This gives the editor certain math capabilities so that operations like "counting the number of times a string exists" can be done.

Ref. # AA16

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

DISKETTE RECOVERY PROGRAM, RECOVERY 1

Function

To permit recovery of files on an ISIS diskette whose directory file has been destroyed, but which is otherwise intact.

Required Hardware

Disk based Intellec MDS-800

Required Software

ISIS-II

Input Parameters

Output Results

DISKETTE AVAILABLE IN LOCATED OBJECT CODE ONLY. PAPER TAPE AVAILABLE ON HEX ONLY. LISTING NOT AVAILABLE.

Registers Modified: ALL	Programmer:
RAM Required:	Company:
ROM Required:	Address:
Maximum Subroutine Nesting Level:	City:
Assembler/Compiler Used: PL/M 80 or CROSS PL/M Compiler	State:

Ref.# AB106

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 ☐ Other_____ (use additional sheets if necessary)

Program Title

PRINT

Function

Lists any ISIS disk file on the line printer up to 255 times. Each copy is paged automatically.

Required Hardware

INTEL MDS-DOS
Centronics 306C line printer
LS1 ADM-3 VDU

Required Software

ISIS Version 1.2
Monitor Version 2.0

Input Parameters

Enter Program name, space, and then file name.

Output Results

Listed File with tabulator characters implemented.

Registers Modified: All	Programmer: R. C. Taylor B.Sc.
RAM Required: As for ISIS	Company: McMichael Limited
ROM Required: Monitor	Address: Wexham Road
Maximum Subroutine Nesting Level: 3	City: Slough
Assembler/Compiler Used: ISIS Macro V 1.1	State: BERKS, SL2 5EL

Ref.# AB107

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004 ☐4040 ☐8008 ☒8080 ☐3000 ☐Other_____ (use additional sheets if necessary)

Program Title

TYPE

Function

Lists any ISIS disk file on the system console (VDU) and prompts for a return key every page of information.

Required Hardware

INTEL MDS-DOS
Centronics 306C line printer
LS1 ADM-3 VDU

Required Software

ISIS Version 1.2
Monitor Version 2.0

Input Parameters

Enter Program name, space, and then file name.

Output Results

Listed File with tabulator characters implemented.

Registers Modified: All	Programmer: R. C. Taylor B.Sc.
RAM Required: As for ISIS	Company: McMichael Limited
ROM Required: Monitor	Address: Wexham Road
Maximum Subroutine Nesting Level: 3	City: Slough
Assembler/Compiler Used: ISIS Macro V 1.1	State: BERKS, SL2 5EL

Ref.# AB108

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004 ☐4040 ☐8008 ☒8080 ☐3000 ☐Other_____ (use additional sheets if necessary)

Program Title

JOIN

Function

Merges 2 hex files from separate assemblies into a single hex file for HEXBIN. Transfer address of primary file is retained.

Required Hardware

MDS

Required Software

ISIS

Input Parameters

ISIS command syntax:

-JOIN <primary file>, <secondary file>

Output Results

Secondary file is appended to primary file; transfer address of secondary file is deleted; transfer address of primary file is applied to the total file.

Registers Modified:	Programmer: Richard Kucia
RAM Required: From 3200H through 353CH	Company: Realistic Controls
ROM Required:	Address: 3530 Warrensville Ctr. Rd.
Maximum Subroutine Nesting Level:	City: Shaker Heights
Assembler/Compiler Used: ISIS Assembler 1.1	State: Ohio

Ref.# AB109

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 ☒8080 ☐8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

PRINT PROGRAM FOR G.E. TERMINET - 1200 PRINTER

Function

Writes a diskette file to :TO:, inserting correct delays for CR LF characters for a G.E. Terminet - 1200 so the printer doesn't overprint short lines.

Required Hardware

Intel disk drive, G.E. Terminet-1200 Printer on :TO: port, :TO: port strapped for 1200 Baud

Required Software

ISIS-II DOS

Input Parameters

When ISIS command prompt shows on console, press "ON LINE" button on Terminet, enter "PRINT to <filename> CR" on console, where <filename> is an ISIS diskette file.

Output Results

Prints file on Terminet Printer with correct CR LF delays. If error or when done, displays a message on the console.

Registers Modified: ALL	Programmer: John S. Santic
RAM Required: 617 BYTES DECIMAL	Company: Western Union Data Services
ROM Required: --	Address: 70 McKee Drive
Maximum Subroutine Nesting Level: --	City: Mahwah
Assembler/Compiler Used: PL/M Resident Compiler	State: New Jersey 07430

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/4040 ☐ 8008 ☒ 8080 ☐ 8048 ☐ 8085 ☐ Other_____ (use additional sheets if necessary)

Program Title

PRINT OUT SOURCE FILE OF FLOPPY DISK

Function

The function of this program is to print out source files stored on the floppy disk to the designated list device. This program recognizes the tab character. The tab in this program is set at every 8 spaces.

Required Hardware

MDS 800
Console
Floppy Disk

Required Software

ISIS-II, V2.2
Monitor Program

Input Parameters

Go to monitor and assign the list device to console.
Go to ISIS II, V2.2
Enter P followed by a space and the filename to be printed.

Output Results

The program will read the designated file and output this file to the console using the monitor output routine. The tab character will be recognized by this program and set at every 8 spaces.

Registers Modified: ALL	Programmer: Bill Uhlarik
RAM Required: Ø172H Bytes	Company: ITT Barton
ROM Required: --	Address: 900 So. Turbull Cyn Rd.
Maximum Subroutine Nesting Level: --	City: City of Industry
Assembler/Compiler Used: INTELLEC MDS MACRO ASSEMBLER	State: California

```
ASM80 :F1:AB110

ISIS-II 8080/8085 ASSEMBLER, V1.0        MODULE                                  PAGE    1

 LOC  OBJ          SEQ          SOURCE STATEMENT

                     0 ;REF. NO. AB110
                     1 ;PROGRAM TITLE P
                     2 ;
                     3 ;
                     4 ;
                     5 ;
                     6 ;
                     7 ;
                     8 ;
                     9 ;.....................................................................
                    10 ;
                    11 ;    THE FOLLOWING PROGRAM PRINTS OUT A SPECIFIED FILE ON THE CONSOLE
                    12 ;    DEVICE AND IS INVOKED BY THE COMAND
                    13 ;
                    14 ;     P FILENAME
                    15 ;
                    16 ;    BY BILL UHLARIK        SEPT 21,1977
                    17 ;.....................................................................
                    18 ;
                    19 ;
                    20 ;
0000                21 OPEN     EQU     0
0001                22 CLOSE    EQU     1
0003                23 READ     EQU     3
0009                24 EXIT     EQU     9
000C                25 ERROR    EQU     12
000A                26 ENDL     EQU     0AH         ;LINE FEED CHARACTER
0008                27 ICOUNT   EQU     08H         ;TAB COUNT = 8 SPACES
0009                28 TAB      EQU     09H         ;TAB CHARACTER
F80F                29 LO       EQU     0F80FH      ;ADDRESS OF OUTPUT ROUTINE IN MONITOR PGM
                    30 ;
                    31          EXTRN   ISIS
                    32 ;
A000                33          ORG     0A000H
A000 CDA2A0         34 BEGIN:   CALL    ADV             ;CLEAR :LP: BUFFER
A003 210400    S    35          LXI     H,STACK+4
A006 F9             36          SPHL
A007 0E03           37          MVI     C,READ          ;READ THE CONSOLE
A009 11DEA0         38          LXI     D,RBLK
A00C CD0000    E    39          CALL    ISIS
A00F 3AECA0         40          LDA     STATUS
A012 B7             41          ORA     A
A013 C2C0A0         42          JNZ     ERR
                    43 ;
A016 0E00           44          MVI     C,OPEN          ;OPEN THE INPUT FILE
A018 11D0A0         45          LXI     D,OBLK
A01B CD0000    E    46          CALL    ISIS
A01E 3AECA0         47          LDA     STATUS
A021 B7             48          ORA     A
A022 C2C0A0         49          JNZ     ERR
A025 2ADEA0         50          LHLD    AFT
A028 22DAA0         51          SHLD    CAFT
                    52 ;
A02B 3E08           53          MVI     A,ICOUNT        ;INITIALIZE TAB COUNT
A02D 32F0A0         54          STA     COUNT
A030 0E03           55 LOOP:    MVI     C,READ          ;READ THE INPUT FILE
A032 11DEA0         56          LXI     D,RBLK
A035 CD0000    E    57          CALL    ISIS
A038 3AECA0         58          LDA     STATUS
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

A03B B7             59          ORA     A
A03C C2C0A0         60          JNZ     ERR
A03F 2AE8A0         61          LHLD    ACTUAL
A042 7C             62          MOV     A,H
A043 B5             63          ORA     L
A044 CAADA0         64          JZ      DONE
                    65 ;
                    66                                  ;START OF PRINT
A047 21F1A0         67          LXI     H,BUFFER        ;INITIALIZE MEMORY POINTER
A04A 2271A1         68          SHLD    BP
A04D 2AE8A0         69          LHLD    ACTUAL          ;INITIALIZE BUFFER COUNTER FOR PRINT LOOP
A050 2B             70          DCX     H               ;    ACTUAL =ACTUAL - 1
A051 22E8A0         71          SHLD    ACTUAL



A054 2A71A1         72 PRNT:    LHLD    BP
A057 7E             73          MOV     A,M
A058 FE09           74          CPI     TAB             ;CHECK FOR TAB
A05A CA83A0         75          JZ      SPACE
A05D FE0A           76          CPI     ENDL            ;CHECK FOR END OF LINE
A05F CA9AA0         77          JZ      RESET1
A062 4E             78 CONT1:   MOV     C,M             ;OUTPUT CHARACTER
A063 CD0FF8         79          CALL    LO
A066 3AF0A0         80          LDA     COUNT           ;UPDATE TAB COUNT
A069 3D             81          DCR     A
A06A 32F0A0         82          STA     COUNT
A06D CA92A0         83          JZ      RESET
A070 23             84 CONT:    INX     H               ;UPDATE BUFFER POINTER
A071 2271A1         85          SHLD    BP
A074 2AE8A0         86          LHLD    ACTUAL          ;CHECK FOR END OF BUFFER
A077 7C             87          MOV     A,H
A078 B5             88          ORA     L
A079 CA30A0         89          JZ      LOOP
A07C 2B             90          DCX     H               ;UPDATE BUFFER COUNTER
A07D 22E8A0         91          SHLD    ACTUAL          ;    ACTUAL =ACTUAL - 1
A080 C354A0         92          JMP     PRNT
A083 0E20           93 SPACE:   MVI     C,20H           ;OUTPUT SPACE
A085 CD0FF8         94          CALL    LO
A088 3AF0A0         95          LDA     COUNT           ;CHECK IF TAB IS REACHED
A08B 3D             96          DCR     A
A08C 32F0A0         97          STA     COUNT
A08F C254A0         98          JNZ     PRNT
A092 3E08           99 RESET:   MVI     A,ICOUNT        ;RE-INITIALIZE TAB COUNT
A094 32F0A0        100          STA     COUNT
A097 C370A0        101          JMP     CONT
A09A 3E09          102 RESET1:  MVI     A,ICOUNT+1      ;RE-INITIALIZE TAB COUNT
A09C 32F0A0        103          STA     COUNT
A09F C362A0        104          JMP     CONT1
                   105 ;
A0A2 0E0D          106 ADV:     MVI     C,0DH           ;CARRIAGE RETURN & LINE FEED ROUTINE
A0A4 CD0FF8        107          CALL    LO
A0A7 0E0A          108          MVI     C,0AH
A0A9 CD0FF8        109          CALL    LO
A0AC C9            110          RET
                   111 ;
A0AD CDA2A0        112 DONE:    CALL    ADV
A0B0 0E01          113          MVI     C,CLOSE         ;CLOSE THE INPUT FILE
A0B2 11DAA0        114          LXI     D,CBLK
A0B5 CD0000    E   115          CALL    ISIS
A0B8 0E09          116          MVI     C,EXIT          ;NORMAL EXIT
A0BA 11EAA0        117          LXI     D,XBLK
A0BD CD0000    E   118          CALL    ISIS
                   119 ;
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

A0C0 0E0C          120 ERR:     MVI     C,ERROR         ;ERROR MESSAGE
A0C2 11ECA0        121          LXI     D,EBLK
A0C5 CD0000    E   122          CALL    ISIS
A0C8 0E09          123          MVI     C,EXIT          ;ERROR EXIT
A0CA 11EAA0        124          LXI     D,XBLK
A0CD CD0000    E   125          CALL    ISIS
                   126 ;
A0D0 DEA0          127 OBLK:    DW      AFT
A0D2 F1A0          128          DW      BUFFER
A0D4 0100          129          DW      1
A0D6 0000          130          DW      0
A0D8 ECA0          131          DW      STATUS
                   132 ;
                   133 CBLK:
A0DA               134 CAFT:    DS      2
A0DC ECA0          135          DW      STATUS
                   136 ;
                   137 RBLK:
A0DE 0100          138 AFT:     DW      1
A0E0 F1A0          139          DW      BUFFER
A0E2 8000          140          DW      128
A0E4 E8A0          141          DW      ACTUAL
A0E6 ECA0          142          DW      STATUS
                   143 ;
A0E8               144 ACTUAL:  DS      2
                   145 ;
A0EA ECA0          146 XBLK:    DW      STATUS
                   147 ;
                   148 EBLK:
A0EC               149 STATUS:  DS      2
A0EE ECA0          150          DW      STATUS


                   151 ;
A0F0               152 COUNT:   DS      1
A0F1               153 BUFFER:  DS      128
                   154 ;
                   155 ;
A171               156 BP:      DS      2
                   157 ;
A000               158          END     BEGIN
```

PUBLIC SYMBOLS

EXTERNAL SYMBOLS
ISIS E 0000

USER SYMBOLS

```
ACTUAL A A0E8   ADV    A A0A2   AFT    A A0DE   BEGIN  A A000   BP     A A171   BUFFER A A0F1
  CAFT   A A0DA
CBLK   A A0DA   CLOSE  A 0001   CONT   A A070   CONT1  A A062   COUNT  A A0F0   DONE   A A0AD
  EBLK   A A0EC
ENDL   A 000A   ERR    A A0C0   ERROR  A 000C   EXIT   A 0009   ICOUNT A 0008   ISIS   E 0000
  LO     A FE0F
LOOP   A A030   OBLK   A A0D0   OPEN   A 0000   PRNT   A A054   RBLK   A A0DE   READ   A 0003
  RESET  A A092
RESET1 A A09A   SPACE  A A083   STATUS A A0EC   TAB    A 0009   XBLK   A A0EA
```

ASSEMBLY COMPLETE, NO ERROR(S)

Ref.# AB111

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 ☐8080 ☐8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

ONLINE, UPLOAD, DOWNLOAD

Function

To use the Intellec MDS-800 as a terminal or to transfer files to/from any timesharing system. (Specifically PDP-10)

Required Hardware

A Serial Port
(Attached documentation describes addition of an RS232 Port)

Required Software

User must write a driver on the host computer to communicate with upload and download.
User must also modify programs to compile on the compiler chosen and to run on the user's system configuration.

Input Parameters

Output Results

NOTE: ONLINE is available in Assembly Language only. UPLOAD and DOWNLOAD are available in PL/M only.

Registers Modified: ALL	Programmer:
RAM Required:	Company:
ROM Required:	Address:
Maximum Subroutine Nesting Level:	City:
Assembler/Compiler Used: See Note	State:

ONLINE, UPLOAD, DNLD

To use the MDS as a terminal, or to transfer files to/from the PDP-10, the user needs three MDS programs: ONLINE, UPLOAD, DNLD. The user may need to make changes to the hardware configuration as specified in the attached spec. Once the hardware changes are made, the general algorithm is:

- Use program ONLINE to call up the PDP-10.
- Hit carriage return twice (should get period'.' prompt)
- Log on to the PDP-10 (or your host computer)
- When ready to transfer files, hit the 'break' key, which will return you to ISIS.
- Type what needs done:

  UPLOAD MDS-filename to PDP-filename
  DNLD PDP-filename to MDS-filename

- To get back to the PDP-10, re-type ONLINE (no need to log in again)
- Be sure to log off.

MDS Modifications for PDP-10 Communication

When choosing a serial port, there are two alternatives on the Intellec MDS. The first you may use the teletype port, which is a current loop serial line. Otherwise, a second monitor board may be added to the system, in parallel to the original monitor board, giving you an extra RS232 port.

Modifications are made to this extra monitor board so that its CRT port can be used as the PDP-10 port:

First, the base address of the "CRT" port on the new monitor board must be changed from FØ to DØ. This entails cutting the trace to pin 15 of chip A34, and jumping pin 14 of the same chip to the feed-through adjacent to pin 16 (which is where pin 15 used to be connected).

Second, the new monitor board needs a special harness which connects to the CRT connector (PDP-10 line). There are four wires in the harness and one junper

On the board connector, 2 wires from pin A15 go to cable connector pin 7. One wire from board connector pin 14 to cable connector pin 3. One wire from board connector pin 15 to cable connector pin 2. One jumper on board connector from pin A4 to A10.

Software

Be sure your software contains the correct port numbers for the serial line you chose to use.

Baud Rate Changes

If a baud rate of 300 is desired fo the following on the second monitor card: Change jumper from 2400 baud (standard 19 to 20) to 300 baud (13 to 14).

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref.# AE12

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 (use additional sheets if necessary)

Program Title

STEP

Function

Single-steps an assembly language program and halts in the monitor after each instruction. The registers may then be examined or changed, or the program continued. The STEP program can be used in conjunction with the Monitor's "GO" command to execute subroutines, then continue the single-step mode. May be resident completely in ROM.

Required Hardware

8080 CPU console

Required Software

Monitor version 3.0 (can be changed)

Input Parameters

User program via the Monitor "GO" command using at least 1 breakpoint. After executing the breakpoint, the program may be single stepped in one of two ways (assuming STEP is in ROM):

1) Place a JMP to STEP in an unused restart location, and interrupt the CPU with the console switches set to the proper value
2) Modify the monitor to have another command (may be "O") and type that letter.

Output Results

The STEP program will execute the next instruction and then print the new PC, all registers are saved. This program will not single step in ROM, as it uses the restart 1 breakpoint logic.

Registers Modified: All	**Assembler/Compiler Used:** Macro V.4.0
RAM Required: Monitor exit template	**Programmer:** Gary Saxer
ROM Required: 246	**Company:** ATE ASSOCIATES, INC.
Maximum Subroutine Nesting Level: 1	**Address:** 8448 Reseda Blvd., #201 Northridge, CA 91324

98-034C

```
8Ø8Ø V3.Ø
·D1ØØ, 1ØF  (Little program from 100-10A)
Ø1ØØ 21 ØØ Ø2 ØØ 3E ØF 23 3C C3 Ø6 Ø1 ØØ ØØ ØØ ØØ 26
·S1Ø ØØ-C3 ØØ- ØØ-35  (Place JMP to STEP at RST 2 location, and set console switches to D7H
·G1ØØ,1ØØ (Start Program)
*Ø1ØØ   (Stops at beginning)
.X  (Register contents)
A=ØØ B=ØØ C=ØØ D=ØØ E=ØØ F=Ø2 H=ØØ L=ØØ M=ØØØØ P=Ø1ØØ S=Ø1ØØ
·<——— Interrupt switch depressed, causes RST 2 to be executed
*Ø1Ø3
·X                                  note change
A=ØØ B=ØØ C=ØØ D=ØØ E=ØØ F=Ø2 H=Ø2 L=ØØ M=Ø2ØØ P=Ø1Ø3 S=Ø1ØØ
·<——— RST 2
*Ø1Ø4
.X                  no change (NOP)
A=0Ø/ B=ØØ C=ØØ D=ØØ E=ØØ F=Ø2 H=Ø2 L=ØØ M=Ø2ØØ P=Ø1Ø4 S=Ø1ØØ
·<——— RST 2
*Ø1Ø6
.X     note change
A=ØF B=ØØ C=ØØ D=ØØ E=ØØ F=Ø2 H=Ø2 L=ØØ M=Ø2ØØ P=Ø1Ø6 S=Ø1ØØ
·<——— RST 2
*Ø1Ø7
.X                                  note change
A=ØF B=ØØ C=ØØ D=ØØ E=ØØ F=Ø2 H=Ø2 L=Ø1 M=Ø2Ø1 P=Ø1Ø7 S=Ø1ØØ
·<——— RST 2
*Ø1Ø8
.X
A=1Ø B=ØØ C=ØØ D=ØØ E=ØØ F=12 H=Ø2 L=Ø1 M=Ø2Ø1 P=Ø1Ø8 S=Ø1ØØ
·<——— RST 2
*Ø1Ø6 <———————— note that the target of the JMP instruction was trapped
·X
A=1Ø B=ØØ C=ØØ D=ØØ E=ØØ F=12 H=Ø2 L=Ø1 M=Ø2Ø1 P=Ø1Ø6 S=Ø1ØØ
·<——— RST 2
*Ø1Ø7
·X
A=1Ø B=ØØ C=ØØ D=ØØ E=ØØ F=12 H=Ø2 L=Ø2 M=Ø2Ø2 P=Ø1Ø7 S=Ø1ØØ
·<——— RST 2
*Ø1Ø8
.X
A=11 B=ØØ C=ØØ D=ØØ E=ØØ F=Ø6 H=Ø2 L=Ø2 M=Ø2Ø2 P=Ø1Ø8 S=Ø1ØØ
·<——— RST 2
*Ø1Ø6
.G,1Ø8 <——"GO" command break points at 108
*Ø1Ø8
.X
A=12 B=ØØ C=ØØ D=ØØ E=ØØ F=Ø6 H=Ø2 L=Ø3 M=Ø2Ø3 P=Ø1Ø8 S=Ø1ØØ
·<——— RST 2  "Step past 108"
*Ø1Ø6
.G,1Ø8 <——Then do instructions and STOP at 108 again
*Ø1Ø8
.X
A=13 B=ØØ C=ØØ D=ØØ E=ØØ F=Ø2 H=Ø2 L=Ø4 M=Ø2Ø4 P=Ø1Ø8 S=Ø1ØØ
```

```
                        ;REF. NO. AE12
                        ;PROGRAM TITLE STEP
                        ;
                        ;
                        ;
                                TITLE    ISTRUCTION STEP 11-17-76'
                        ; INSTRUCTION STEP FUNCTION  G. SAXER 11-17-76
386B                    START   EQU     386BH
3809                    CO      EQU     3809H
                        ;VERSION 3.0 EQUATES (NOTE MEMSIZ NOT MEMCHK)
3A26                    CO3     EQU     3A26H   ;CONTINUE WITH PROGRAM (V3)
3E01                    MEMSIZ  EQU     3E01H
                        ;
                        ; THE FOLLOWING SPECIAL CODE IS NEEDED IF THE ROUTINE IS
                        ;NOT CALLED DIRECTLY FROM THE MONITOR (E.G. RESTART)
                        ;
0000 CD013E                     CALL    MEMSIZ  ;GET TOP OF MEM
0003 11F8FF                     LXI     D,-8    ;POINT AT END OF REGS
0006 19                         DAD     D       ;HL->TOP OF STACK FOR MONITOR
0007 F9                         SPHL            ;FAKE MONITOR STACK
                        ;
                        ;       END OF SPECIAL AREA
                        ;
0008 211300             INCR:   LXI     H,0013H ;GET OLD PC
000B 39                         DAD     SP
000C 46                         MOV     B,M
000D 2B                         DCX     H
000E 4E                         MOV     C,M
000F 1E01                       MVI     E,01H   ;E=BYTES TO SKIP
0011 211400                     LXI     H,0014H ;POINT TO TLOC
0014 39                         DAD     SP
0015 E5                         PUSH    H       ;SAVE TLOC
0016 0A                         LDAX    B       ;LOAD NEXT OP CODE
0017 FEC3                       CPI     0C3H    ;JMP?
0019 CA3300                     JZ      INCRA   ;YES
001C FECD                       CPI     0CDH    ;CALL?
001E CA3300                     JZ      INCRA   ;YES
0021 FE3F                       CPI     03FH    ;<= MVI A?
0023 DA3C00                     JC      INCRB   ;YES
0026 FEC1                       CPI     0C1H    ;> POP B?
0028 D25500                     JNC     INCRC   ;YES
002B 1D                         DCR     E       ;FAKE E=0
002C 1C                 INCRD:  INR     E       ;E=E+1
002D CD9D00             INCRE:  CALL    TRAP    ;TRAP AT (BC)+(E)
0030 C3263A             INCRF:  JMP     CO3     ;CRLF AND RESUME PROG
                        ;
                        ;   UNCONDITIONAL JUMPS
                        ;
0033 CDE700             INCRA:  CALL    LDBI    ;PLACE LOC IN BC
0036 CDA200                     CALL    TRAPA   ;TRAP INST
```

```
0039 C33000             JMP     INCRF   ;RESUME PROC
                    ;
                    ;   OP CODES <= 03E
                    ;
003C CD9300     INCRB:  CALL    SIXE    ;8B IMMEDIATE? (YES=NO RETURN)
003F FE01               CPI     01H     ;16B IMMED?
0041 CA5100             JZ      THREE   ;YES
0044 7A                 MOV     A,D     ;GET OP CODE



0045 FE20               CPI     020H    ;< LXI H?
0047 DA2D00             JC      INCRE   ;YES E=1
004A E607               ANI     07H     ;CLEAR UPPER
004C FE02               CPI     02H     ;ADR?
004E C22D00             JNZ     INCRE   ;NO E=1
0051 1C         THREE:  INR     E
0052 C32C00             JMP     INCRD   ;E=3
                    ;
                    ;  OP CODES > C1
                    ;
0055 CD9300     INCRC:  CALL    SIXE    ;8B IMMED?
0058 FE08               CPI     08H     ;RET?
005A CAB400             JZ      TRPRT   ;YES
005D B7                 ORA     A       ;RET?
005E CAB400             JZ      TRPRT   ;YES
0061 1F                 RAR             ;EVEN OP CODE?
0062 DA7100             JC      IOST?   ;NO
0065 CDE700             CALL    LDBI    ;GET LOC TO TRAP
0068 CDA200             CALL    TRAPA   ;TRAP (BC)
006B CDF000             CALL    FIX
006E C35100             JMP     THREE   ;TRAP NEXT LOC
0071 7A         IOST?:  MOV     A,D     ;CHECK EXCEPTIONS
0072 FED3               CPI     0D3H    ;OUT?
0074 CA2C00             JZ      INCRD   ;YES E=2
0077 FEDB               CPI     0DBH    ;IN?
0079 CA2C00             JZ      INCRD   ;YES
007C FEC9               CPI     0C9H    ;REAL RET? (UNCOND)
007E CAB400             JZ      TRPRT   ;YES
0081 FEE9               CPI     0E9H    ;PCHL?
0083 C22D00             JNZ     INCRE   ;NO E=1
                    ;
0086 211100             LXI     H,0011H ;YES GET HL INTO BC
0089 39                 DAD     SP      ;POINT TO HL LOC
008A 4E                 MOV     C,M
008B 23                 INX     H
008C 46                 MOV     B,M
008D CDA200             CALL    TRAPA
0090 C33000             JMP     INCRF   ;OUT OF HERE
                    ;  CHECK FOR IMMEDIATE INST
                    ;
0093 57         SIXE:   MOV     D,A     ;SAVE OP CODE
0094 E607               ANI     07H     ;CLEAR UPPER 5
```

```
0096 FE06               CPI     06H
0098 C0                 RNZ                     ;NOT 8B IMMED
0099 E1         SIXXT:  POP     H               ;IGNORE OUR CALL
009A C32C00             JMP     INCRD           ;E=2
                ;
                ;  SET TRAP AT (BC)+(E)
                ;
009D 03         TRAP:   INX     B               ;BC+E
009E 1D                 DCR     E
009F C29D00             JNZ     TRAP
00A2 E1         TRAPA:  POP     H               ;RET ADDR IN HL
00A3 E3                 XTHL                    ;SWAP WITH TLOC
00A4 78         MOV     A,B
00A5 B1                 ORA     C               ;NO TRAP AT ZERO
00A6 CAC800             JZ      STOP
00A9 71                 MOV     M,C             ;SAVE TRAP LOC
00AA 23                 INX     H
00AB 70                 MOV     M,B
00AC 23                 INX     H
00AD 0A                 LDAX    B               ;LOAD OP CODE
00AE 77                 MOV     M,A             ;SAVE IT
00AF 23                 INX     H               ;POINT TO NEXT TLOC
00B0 3ECF               MVI     A,0CFH          ;LOAD RST 1
00B2 02                 STAX    B               ;SET TRAP
00B3 C9                 RET
                ;


                ;  TRAP A RETURN
                ;
00B4 210900     TRPRT:  LXI     H,0009H ;OLD SP LOC
00B7 39                 DAD     SP
00B8 56                 MOV     D,M
00B9 2B                 DCX     H
00BA 5E                 MOV     E,M             ;GET SP POINTER
00BB EB                 XCHG                    ;POINT TO RET ADDR
00BC CDEA00             CALL    LDBI2
00BF CDA200             CALL    TRAPA
00C2 CDF000             CALL    FIX
00C5 C32D00             JMP     INCRE           ;AFTER RET INST
                ;
                ;  ILLEGAL TRAP
                ;
00C8 21D900     STOP:   LXI     H,MSG
00CB 060E               MVI     B,0EH
00CD 4E                 MOV     C,M
00CE CD0038             CALL    CO
00D1 05                 DCR     B
00D2 23                 INX     H
00D3 C2CD00             JNZ     $-6
00D6 C36B38             JMP     START
00D9 54524150 MSG:      DB      'TRAP TO ZERO',0DH,0AH
```

```
00DD 20544F20
00E1 5A45524F
00E5 0D0A
                ;
                ;  TRAP A LOC POINTED TO IN MEMORY
                ;
00E7 C5         LDBI:   PUSH    B       ;PUT BC IN HL
00E8 E1                 POP     H
00E9 23                 INX     H       ;POINT PAST OP CODE
00EA C5         LDBI2:  PUSH    B       ;SAVE BC IN DE
00EB D1                 POP     D
00EC 4E                 MOV     C,M     ;GET NEW BC
00ED 23                 INX     H
00EE 46                 MOV     B,M
00EF C9                 RET     ;TRAPA NEXT(TLOC ON TOP CANT CALL HERE)
                ;
                ;  RESTORE LDBI SAVED STUFF
                ;
00F0 E3         FIX:    XTHL            ;GET RET ADDR
00F1 E5                 PUSH    H       ;PUT IT BACK (TLOC PUT BACK TOO)
00F2 D5                 PUSH    D
00F3 C1                 POP     B       ;RESTORE BC
00F4 1E01               MVI     E,1     ;RESET E
00F6 C9                 RET
0000                    END
```

Ref.# AE13

insite™ INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/40 ☐8008 ☐8080 ☐8048 ☐8085 ☐3000 ☐Other_____ (use additional sheets if necessary)

Program Title

Program Test-Loader

Function

Load/Dump/Test ROMSIM (8048 Support Software)

Required Hardware

Intel ROM Simulator, Intellec MDS DOS System

Required Software

Input Parameters

See attached documentation

Output Results

See attached documentation

Available on diskette only.

Registers Modified: N/A	Programmer: John Kovach
RAM Required: N/A	Company: Magnavox
ROM Required: N/A	Address: 1700 Magnavox Way
Maximum Subroutine Nesting Level: N/A	City: Fort Wayne
Assembler/Compiler Used: ISIS I ASM 80 V1.1	State: Indiana

1.0 INTRODUCTION

The Program Test Loader is an Intel 8080 program designed to test, load, and dump the Intel ROM Simulator from the Intellec MDS system operating under either ISIS-I or ISIS-II. It was writtento provide the 8048 developer with a means to load and dump 8048 object files to and from locations DØØØH to DFFFH of the ROM SIMULATOR. It also provides a programmable feature that tests the RAM located in the ROM SIM.

2.1 INVOKING THE PROGRAM

The Program Test-Loader is invoked from either ISIS-I or ISIS-II by typing PTL and a carriage return in response to the system dash prompt (-). The program must exist in binary format on diskette before it can be invoked.

2.2 OPTIONS

The Program Test Loader provides the user with the options tabulated in Table 2A. Each option is invoked by its name or by the short hand notation given for that option (in parenthesis). The format for specifying options is as follows: OPTION, SPACE, LIST, CARRIAGE RETURN.

OPTIONS may be entered after the program has given the question mark prompt ("?"). Each time an option has been completed, the program returns a message to verify the option performed and then issues the question mark prompt ("?") to indicate it is ready to receive another option request. Table 2-B tabulates the responses given to each option request.

Ref.#AE14

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

Symbol Table Program for 8080/8085 *Version 1.2

Function

To print a symbol table of a previously located program sorted alphanumerically or by address, and printed in columns. Publics, Local Symbols, and PL/M Line numbers are included.

Required Hardware

Intellec MDS with disk drive

Required Software

ISIS-II

Input Parameters

See reverse page 4-516

Output Results

See reverse page 4-516

*Revised 12/78

- 20% faster
- now accepts more than 255 symbols per module.

Available in object code on diskette or HEX code on paper tape.

Registers Modified:	All	Programmer:	Gary Carleton
RAM Required:	32K	Company:	Intel Corp.
ROM Required:	--	Address:	
Maximum Subroutine Nesting Level:	--	City:	
Assembler/Compiler Used:	PL/M-80 V3.0	State:	

SYMBOL TABLE PROGRAM

Description:

The program is invoked by entering:

SYMBOL FILE1 [TO FILE2] [PAGEWIDTH(n)] [ADDR] [NOTRUNC]
(Brackets indicate optional controls)

FILE1: Previously located object file program.

FILE2: ISIS file where symbol table is to be written. Can be :LP:, :CO:, a disk file, etc.

PAGEWIDTH(n): Number of characters per line similar to PL/M and assembler controls. n must be less than 200.

ADDR: Specifies sort by address instead of sort by name.

NOTRUNC: Symbol names are normally truncated to the first ten characters. NOTRUNC causes no truncation.

DEFAULT VALUES

These are the values used when the optional controls are not specified:

FILE2 - FILE1.SYM

PAGEWIDTH(n) - n=80

ADDR - Sort by symbol name (or line number)

NOTRUNC - Truncate at tenth character

As with the LOCATE symbol table, the original assemblies or compilations must have used the DEBUG control for the symbol table to be included in the object module. The normal sequence of program execution when using SYMBOL is:

1. Assemblies or Compilations (with DEBUG)
2. LINK (If necessary)
3. LOCATE (without PURGE)
4. SYMBOL

If you use the PURGE control, the following step should be added:

5. LOCATE (with PURGE)

For convenience, steps 3, 4, and 5 may be included in a submit file.

Notes:

The sort algorithm used is not the most efficient. Because of this, large modules may cause the Intellec to pause for a minute or more.

Also, SYMBOL should be on a system diskette for ISIS to properly regain control after execution.

PROGRAM INVOKED BY:
SYMBOL :F1:SAMPLE TO :TO:

PUBLICS:

3B3FH	@P0014	3B40H	@P0015	3B43H	@P0016
3B44H	@P0017	3B4CH	@P0029	3B4EH	@P0030
3B6BH	@P0031	3B6EH	@P0032	3B6FH	@P0033
3B76H	@P0034	3B78H	@P0035	3B8AH	@P0048
3B8DH	@P0049	3B94H	@P0096	3B97H	@P0097
3B9EH	@P0101	3BA1H	@P0102	3A84H	BCDASCIITO
3872H	BINTOBCD	368BH	BLANKS	3A10H	CHAREQL
3BA9H	ERROR	3BBEH	EXIT	3901H	HEADING
37BDH	HEXADDRTOA	3C36H	INAFTN	0040H	ISIS
3789H	ISISERR	3AD2H	LINECONV	3C3AH	LONGNAMES
3C38H	OUTAFTN	3C3DH	OUTFILE	3C3CH	PAGEWIDTH
3BCDH	READ	398FH	READREC	3C3BH	SORTBYADDR
3BF1H	WRITE	39BFH	WRITEREC		

MODULE: MAINPROGRAM
SYMBOLS:

3C36H	INAFTN	3C3AH	LONGNAMES	3C9DH	MEMORY
3C38H	OUTAFTN	3C3DH	OUTFILE	3C3CH	PAGEWIDTH
3C3BH	SORTBYADDR				

LINE NUMBERS:

3680H	7	3686H	8	3689H	9

MODULE: UTILS
SYMBOLS:

3C57H	ASCIIINDEX	3C5BH	BCDADDR	3A84H	BCDASCIITO
3C78H	BCDPTR	3C59H	BINADDR	3C7AH	BINPTR
3872H	BINTOBCD	368BH	BLANKS	3C6BH	BUFFADDR
3A10H	CHAREQL	3C80H	DESTINPTR	3C5DH	DIGIT
3753H	ERRMSG1	376FH	ERRMSG2	3C4DH	ERRNUM
3901H	HEADING	3C55H	HEX	37BDH	HEXADDRTOA
3C58H	HEXINDEX	3C7EH	HEXLINPTR	3C51H	HEXPTR
3C60H	I	3C77H	I	3C7DH	I
3C84H	INDEX	3789H	ISISERR	3C6DH	LEN
3C76H	LEN	3C7CH	LEN	3AD2H	LINECONV
3C9DH	MEMORY	3C82H	NAMLENPTR	3C6FH	NUMOFLINES
3C69H	READCNT	398FH	READREC	3C65H	RECLEN
3C5EH	REMAINDER	3C53H	RESULTPTR	3C4FH	STATUS
3C61H	STATUS	3C67H	STATUS	3C70H	STATUS
3C72H	STRG1PTR	3C74H	STRG2PTR	3C63H	TEMBYTES
39BFH	WRITEREC				

LINE NUMBERS:

3789H	17	378FH	19	37A0H	20
37A8H	21	37B9H	22	37BCH	23
37BDH	24	37C7H	26	37CEH	27

37D6H	28	37DBH	29	37E7H	30
3804H	31	3814H	32	3822H	33
3826H	34	383EH	35	384EH	36
385CH	37	3860H	38	3865H	39
3871H	40	3872H	41	387CH	43
388CH	44	3895H	45	38A9H	46
38B4H	47	38BAH	48	38D0H	49
38DAH	50	38E4H	51	3900H	52
3901H	53	3901H	55	3908H	56
3918H	57	392DH	58	3938H	59
393EH	60	394DH	61	3958H	62
395CH	63	395FH	64	3975H	65
3981H	66	3989H	67	398EH	68
398FH	69	3995H	71	39AAH	72
39B6H	73	39BEH	74	39BFH	75
39CEH	77	39D6H	78	39DDH	80
39E0H	81	39E8H	82	39E8H	83
39FBH	84	3A07H	85	3A0FH	86
3A10H	87	3A1FH	89	3A24H	90
3A4FH	91	3A53H	92	3A56H	93
3A60H	94	3A63H	95	3A7EH	96
3A81H	97	3A84H	98	3A84H	99
3A93H	101	3A98H	102	3A9DH	103
3AA7H	104	3ACAH	105	3ACEH	106
3AD1H	107	3AD2H	108	3AE3H	110
3AEFH	111	3AF4H	112	3B04H	113
3B08H	114	3B0BH	115	3B14H	116
3B34H	117	3B3EH	118		

STARTING ADDRESS: 3680H

Ref.# AB112

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 ☐8080 ☒8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

8048 - Seven Segment Display Interface Subroutines -- SCAN

Function

Collection of utility subroutines which may be used with the 8048 family to scan a keyboard matrix, debounce and encode key depressions, and drive a multiplexed seven segment display. Also included are utilities to translate Hexidecimal data into segment patterns, right or left entry to the display registers, and clearing or writing character sequences to the display.

Required Hardware

Simple X-Y matrix of up to 64 switches, seven-segment display up to eight digits, and high current segment and digit drivers. Matrix & display size may be increased arbitrarily with the addition of external decoders.

Required Software

User-written background program (game, calculator, telephone-dialer, etc.) requiring low-cost keyboard and display capability.

Input Parameters

Determined by user's program and utilities selected.

Output Results

Live keyboard, continuous flicker-free multi-character display.

Program offered on diskette only.

Registers Modified: Pointers & one in Bank0 and four in Bank1	Programmer: J. Wharton
RAM Required: 12 bytes	Company:
ROM Required: 250_{10} bytes	Address:
Maximum Subroutine Nesting Level: 3	City:
Assembler/Compiler Used: ASM48 V2.0	State:

Ref.# AB113

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 ☐8080 ☐8048 ☒8085 ☐Other_____ (use additional sheets if necessary)

Program Title

AP29 "USING THE 8085 SERIAL I/O LINES"

Function

Two software packages using the 8085 SID and SOD pins for serial I/O. The first set of subroutines may be used to interface an 8085 to a CRT at a wide range of baud rates, with automatic rate identification. The second set includes two low-level utilities for recording and reloading bytes of data using an inexpensive audio cassette recorder and simple interface.

Required Hardware

Required Software

Input Parameters

Output Results

This code was used in the appendix of Intel Application Note AP-29, "USING THE INTEL 8085 SERIAL I/O LINES". The software and hardware required is fully described in the note.

The Application note AP-29 will be included with the program.

Registers Modified: All	Programmer: J. Wharton
RAM Required: 4 Bytes & Stack	Company:
ROM Required: 326 Bytes	Address:
Maximum Subroutine Nesting Level: 3	City:
Assembler/Compiler Used: ASM80, V1.0	State:

Ref.# AB114

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 ☐8080 ☒8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

8048 TUNE GENERATOR

Function

Generates fixed sequence of tones and spaces (i.e. Music)

Required Hardware

See attached diagram.

Required Software

None

Input Parameters

This program will run repetitively through the sequence of notes and spaces starting address TAB, ending when OFFH character is encountered. The pitch and tempo will vary with basic clock frequency but can be altered in the program (this program was run using ALE = 5 m.sec.)

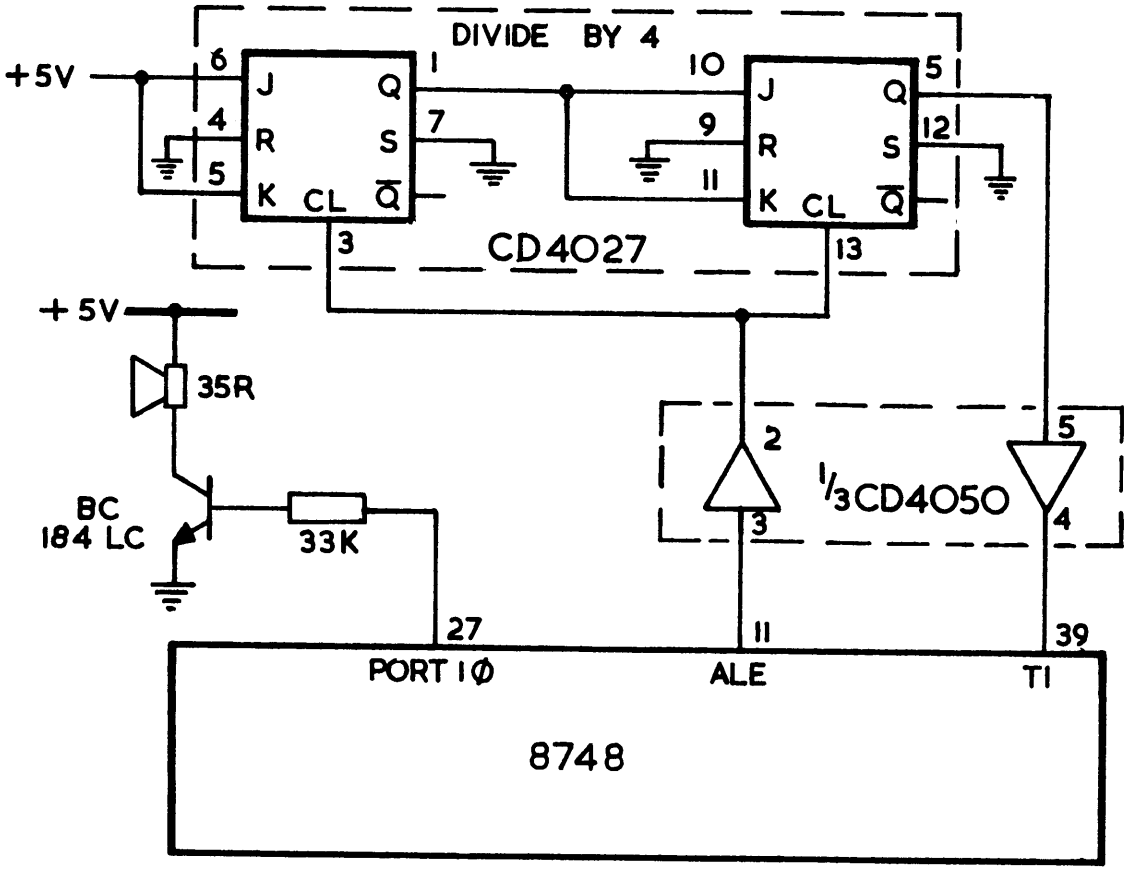
Output Results

Programmed tune

Registers Modified: R0. R1. R2. R3 + Flags F0 + F1	Programmer: T. Harvey
RAM Required: 4 Bytes for registers	Company: Avery-Hardoll Ltd.
ROM Required: 57 + Tune	Address: Downley Road
Maximum Subroutine Nesting Level: --	City: HAVANT, Hants.
Assembler/Compiler Used: ISIS-II 8048 Assembler V1.2	State: ENGLAND

Ref. # AB114

AVERY-HARDOLL LTD. HAVANT HANTS | THIRD ANGLE PROJECTION



				3						
				2						
FEATURE	CHARACT — ERISTIC	TOLERANCE	DATUM	1						
GEOMETRICAL TOLERANCES				ISS	MOD. No.	ZONE	ALTERATION		SIG.	DATE

ALL SCREW THREADS TO B.S. 3643 UNLESS OTHERWISE STATED.
SURFACE FINISH MICROMETER SYMBOLS ARE TO B.S. 1134 CLA.
SYSTEM SURFACES MARKED ✓ TO BE MACHINED TO FINISH INDICATED.

USED ON | USED ON

MAT'L. | SPEC.

FINISH. | C.S. No.

LIMITS UNLESS OTHERWISE STATED.

FROM TO	0 5·9	6 29·9	30 99·9	100 299·9	300 999·9	1000 2000
TOL.	±0·1	±0·2	±0·3	±0·5	±0·8	±1·2

PART NAME

HARDWARE REQUIRED FOR GENERATION OF TUNE SOFTWARE

4/78

SCALE:— —

SUB ASSY ——————

DRN. | TR. | CHK.

PART No.

SIZE A4

```
ASM48 :F1:AB114.SRC

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0                PAGE    1

  LOC  OBJ          SEQ          SOURCE STATEMENT

                      1
                      2          ;*****************************************
                      3          ; TUNE GENERATION ROUTINE
                      4          ;
                      5          ;            27-1-78
                      6          ;*****************************************
                      7
                      8          ; WHEN CALLED AS A SUBROUTINE, THIS PROGRAM
                      9          ; GENERATES A PREDETERMINED TUNE IN THE FORM
                     10          ; OF A SQUARE WAVE TONE ON PORT 1, BIT 0.
                     11          ; DURATION OF NOTES IS DETERMINED BY #TEMPO.
                     12          ; PAUSES (I.E. NO TONE) ARE SIGNIFIED IN THE
                     13          ; TUNE TABLE BY "P" (#00H).
                     14          ; TUNES SHOULD FINISH WITH #0FFH.
                     15
                     16
  0010               17          ORG 10H
                     18
  0010 1414          19 START:   CALL TUNE        ; CALL TUNE SUBROUTINE
  0012 0410          20          JMP START
                     21
  0014 234F          22 TUNE:    MOV A,#TAB       ; START ADDRESS OF TUNE SEQUENCE
  0016 A8            23          MOV R0,A
  0017 A3            24          MOVP A,@A        ; NOTE DIVISION FACTOR
  0018 A9            25          MOV R1,A
  0019 62            26          MOV T,A
  001A BAFF          27          MOV R2,#0FFH
  001C BB16          28          MOV R3,#TEMPO    ; DEFINES DURATION OF NOTES
  001E 45            29          STRT CNT
  001F 042D          30          JMP INIT
                     31
  0021 7625          32 LOOP1:   JF1 LOOP2        ;JUMP IF "PAUSE"
  0023 163E          33          JTF TIMER        ; JUMP IF TIMER OVERFLOW
                     34
                     35                           ; "NOTE DURATION" TIMING LOOP:
                     36
  0025 EA21          37 LOOP2:   DJNZ R2,LOOP1    ; JUMP BACK IF R2 NOT FULL
  0027 EB21          38          DJNZ R3,LOOP1    ;   "    "   " R3  "   "
  0029 BB16          39          MOV R3,#TEMPO
  002B 18            40          INC R0           ; INCREMENT NOTE ADDRESS
  002C A5            41          CLR F1           ; CLEAR "NO NOTE" FLAG
                     42
  002D F8            43 INIT:    MOV A,R0         ; NEW NOTE ADDRESS
  002E A3            44          MOVP A,@A
  002F A9            45          MOV R1,A         ; NEW DIVISION FACTOR
  0030 C637          46          JZ NOTONE        ; JUMP IF D.F = 00H (PAUSE)
  0032 37            47          CPL A
  0033 C63A          48          JZ FINISH        ; JUMP IF D.F=0FFH (END OF TUNE)
  0035 0421          49          JMP LOOP1        ; JUMP TO NOTE DURATION LOOP
                     50
  0037 B5            51 NOTONE:  CPL F1           ; SET "PAUSE" FLAG
  0038 0425          52          JMP LOOP2        ; JUMP AVOIDING COUNTER O/F TEST
```

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0 PAGE 2

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                   53
003A 99FE          54 FINISH:  ANL P1,#0FEH    ; RESET O/P BIT (PORT 1 BIT 0)
003C 65            55          STOP TCNT
003D 93            56          RETR            ; ** END OF ROUTINE **
                   57
                   58
                   59                          ; TIMER OVERFLOW ROUTINE:
                   60
003E 65            61 TIMER:   STOP TCNT
003F 37            62          CPL A
0040 95            63          CPL F0          ; FLAG SET HI & LO ON
                   64                          ; ALTERNATE CYCLES
0041 B647          65          JF0 OPRST
0043 8901          66          ORL P1,#01H     ; SET OUTPUT BIT (PORT 1 BIT 0)
0045 0449          67          JMP CONT
0047 99FE          68 OPRST:   ANL P1,#0FEH    ; RESET OUTPUT BIT
0049 F9            69 CONT:    MOV A,R1        ; NOTE DIVISION FACTOR
004A 37            70          CPL A
004B 62            71          MOV T,A         ; RELOAD TIMER
004C 45            72          STRT CNT
004D 0421          73          JMP LOOP1       ; RETURN TO "NOTE DURATION" LOOP
                   74
                   75
0016               76          TEMPO   EQU     16H     ;DEFINES LENGTH OF NOTES
                   77
                   78                                  ;NOTE:
003C               79          TC      EQU     60D     ;TOP C
0040               80          B       EQU     64D     ;    B
0043               81          A?      EQU     67D     ;    A#
0047               82          AA      EQU     71D     ;    A
004B               83          G?      EQU     75D     ;    G#
0050               84          G       EQU     80D     ;    G
0055               85          F?      EQU     85D     ;    F#
0059               86          F       EQU     89D     ;    F
005F               87          E       EQU     95D     ;    E
0065               88          D?      EQU     101D    ;    D#
006B               89          D       EQU     107D    ;    D
0071               90          C?      EQU     113D    ;    C#
0078               91          BC      EQU     120D    ;BOT. C
                   92
0000               93          P       EQU     00H     ;PAUSE- NO TONE FOR 1
                   94                                  ;CYCLE
                   95
                   96
                   97
004F 65            98 TAB:     DB D?,D?,P,D?,D?,P,F,F
0050 65
0051 00
0052 65
0053 65
0054 00
0055 59
0056 59
0057 00            99          DB P,D,D,D,P,D?,P,F
0058 6B
```

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0 PAGE 3

```
LOC  OBJ          SEQ          SOURCE STATEMENT

0059 6B
005A 6B
005B 00
005C 65
005D 00
005E 59
005F 59            100          DB F,P,G,G,P,G,G,P
0060 00
0061 50
0062 50
0063 00
0064 50
0065 50
0066 00
0067 4B            101          DB G?,G?,P,G,G,G,P,F
0068 4B
0069 00
006A 50
006B 50
006C 50
006D 00
006E 59
006F 00            102          DB P,D?,D?,P,F,F,P,D?
0070 65
0071 65
0072 00
0073 59
0074 59
0075 00
0076 65
0077 65            103          DB D?,P,D,D,P,D?,D?,D?
0078 00
0079 6B
007A 6B
007B 00
007C 65
007D 65
007E 65
007F 65            104          DB D?,P,D?,P,F,P,G
0080 00
0081 65
0082 00
0083 59
0084 00
0085 50
0086 00            105          DB P,G?,P,A?
0087 4B
0088 00
0089 43
008A 43            106          DB A?,P,A?,A?,P,A?,A?,P
008B 00
008C 43
008D 43
008E 00
008F 43
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

0090 43
0091 00
0092 43            107          DB A?,A?,A?,P,G?,P,G,G
0093 43
0094 43
0095 00
0096 4B
0097 00
0098 50
0099 50
009A 00            108          DB P,G?,G?,P,G?,G?,P,G?
009B 4B
009C 4B
009D 00
009E 4B
009F 4B
00A0 00
00A1 4B
00A2 4B            109          DB G?,P,G?,G?,G?,P,G,P
00A3 00
00A4 4B
00A5 4B
00A6 4B
00A7 00
00A8 50
00A9 00
00AA 59            110          DB F,F,P,G,G,P,G?,P
00AB 59
00AC 00
00AD 50
00AE 50
00AF 00
00B0 4B
00B1 00
00B2 50            111          DB G,P,F,P,D?,P,G,G
00B3 00
00B4 59
00B5 00
00B6 65
00B7 00
00B8 50
00B9 50
00BA 50            112          DB G,P,G?,P,A?,A?,P,TC
00BB 00
00BC 4B
00BD 00
00BE 43
00BF 43
00C0 00
00C1 3C
00C2 00            113          DB P,G?,P,G,G,P,F,F
00C3 4B
00C4 00
00C5 50
00C6 50
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0                PAGE     5

  LOC  OBJ          SEQ          SOURCE STATEMENT

  00C7 00
  00C8 59
  00C9 59
  00CA 00            114          DB P,D?,D?,D?,D?,D?,D?,P
  00CB 65
  00CC 65
  00CD 65
  00CE 65
  00CF 65
  00D0 65
  00D1 00
  00D2 00            115          DB P,P,P,P,P,P,P,0FFH
  00D3 00
  00D4 00
  00D5 00
  00D6 00
  00D7 00
  00D8 00
  00D9 FF
                     116
                     117          END

USER SYMBOLS
A?      0043    AA      0047    B       0040    BC      0078    C?      0071    CONT
E       005F    F       0059    F?      0055    FINISH  003A    G       0050    G?
LOOP2   0025    NOTONE  0037    OPRST   0047    P       0000    START   0010    TAB
TIMER   003E    TUNE    0014

ASSEMBLY COMPLETE,  NO ERRORS
```

Ref.# AB115

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004 ☐4040 ☐8008 ☒8080 ☐3000 ☐Other_____ (use additional sheets if necessary)

Program Title

Card Reader Driver, Hollerith to ASCII Conversion

Function

Punched card input utility.
Reads from Documation M-200 Card Reader,
performs Hollerith to ASCII translation,
writes to user-specified diskette file on Intellec MDS
using ISIS-II file management.

Required Hardware

1. Intellec MDS, any console, diskette drives.
2. Documation M-200 Card Reader.
3. CompuScan-designed card reader interface.

Required Software

1. ISIS-II Intel Systems Implementation Supervisor.
2. Executable utility file.

Input Parameters

Standard program invocation format is utilized; e.g. if the utility is filed under the name CDRDRV, then the program is called with the name of the destination diskette file as an argument: CDRDRV filename where filename follows ISIS-II file naming syntax, and may include a diskette drive specifier.

Output Results

Diskette file is created with user-specified name;
data is ASCII coded; card records are trimmed of trailing spaces, and a carriage return and line feed are appended to each record.

Notes:

A. Console break is provided: CTRL-A from the console causes reading to stop prior to the next card pick; the disk file is closed, and control returns to ISIS.

B. Normal file termination via end-of-file card consisting of a multi-punch of at least all eight punches 2 through 9 in column 1.

*C. Practicality of implementation depends upon card reader interface, but Hollerith to ASCII translation will be of general interest.

Registers Modified: not applicable	Programmer: George Cotsonas ~~Alex Schapira~~
RAM Required: code seg length: 2D6H ~~data seg length: A3H~~ stack seg length: 64H	Company: CompuScan, Inc.
ROM Required:	Address: 900 Huyler Street
Maximum Subroutine Nesting Level: unknown but minimal	City: Teterboro
Assembler/Compiler Used: ISIS-II 8080/8085 Assembler v.1.0	State: New Jersey 07608

Ref.# AB116

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 ☒8080 ☐8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

COMPARE files

Function

Compares two files, and indicates whether they are identical or not.

Required Hardware

MDS-800

Required Software

ISIS-I or II

Input Parameters

Execute by the command

-COMPAR file1 file2

(space or comma between filenames)

Output Results

If files identical, a message to that effect.
If not, the differences are listed on the console.
If more than 8 bytes differ, further messages are not output, but the total number of errors is stated at the end.
If the files are of different lengths, this is also stated.

Registers Modified: ALL	Programmer: D. W. Wright
RAM Required: 32K	Company: Standard Telecommunication Laboratories Limited
ROM Required: --	Address: London Road
Maximum Subroutine Nesting Level:	City: Harlow, Essex.
Assembler/Compiler Used: PL/M-80 and ASM80	State: UK

Ref.# AA17

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 ☒8080 ☐8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

SBC 8Ø/1Ø 8255 Test

Function

Test SBC 8Ø/1Ø 8255 I/O Ports
as Mode Ø outputs

Required Hardware

SBC 8Ø/1Ø, TTY/CRT console

Required Software

SBC 8Ø P Monitor

Input Parameters

To initiate test type G3DØØ on console.

Output Results

To Console;
- # ; on completion of test
- Port #, error bit pattern

; if an error occurrs during test

Registers Modified:	Programmer: P. N. Mark
RAM Required: ØØD9H	Company: Durand Machine Co. Ltd.
ROM Required:	Address: 101-11th Street
Maximum Subroutine Nesting Level:	City: New Westminster
Assembler/Compiler Used:	State: British Columbia, CANADA

```
ASM80 :F1:SBC80

ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0          MODULE   PAGE    1

  LOC  OBJ          SEQ          SOURCE STATEMENT

                      1
  FFFF                2 SBC80 SET 0FFFFH
                      3
                      4 ;TITLE'SBC80/10 8255 MODE 0 OUTPUT TEST PBC1V102077'
                      5
                      6 ;PERFORMS INCREASING& DECREASING BIT TEST ON SBC80/10
                      7 ;8255 PORTS (PROGRAMMED AS MODE 0 OUTPUTS)
                      8 ;IF ANY ERROR IS ENCOUNTERED,
                      9 ;THE PORT # &THE ERROR PATTERN IS OUTPUTED
                     10 ;TO THE SYSTEM'S CONSOLE
                     11 ;
                     12 ;PROGRAM EQUATES
                     13
  00E7               14 CWR1 EQU 0E7H ;GROUP 1 CNTL REG
  00EB               15 CWR2 EQU 0EBH ;   "    2  ""
  0080               16 MCW1 EQU 080H ;   "    1 MODE 0 OUTPUT
  0080               17 MCW2 EQU 080H ;   "    2  "  "
  00E4               18 POT0 EQU 0E4H ;PORT 0 GROUP 1
  00E5               19 POT1 EQU 0E5H ;  " 1
  00E6               20 POT2 EQU 0E6H ;  " 2
  00E8               21 POT3 EQU 0E8H ;PORT 3 GROUP 2
  00E9               22 POT4 EQU 0E9H ;  " 4
  00EA               23 POT5 EQU 0EAH ;  " 5
                     24 ;
                     25
  0600               26 SETFW EQU 0600H ;FORWARD TEST CONSTANT
  0605               27 SETBK EQU 0605H ;REVERSE  "  "
  01E8               28 CO    SET 01E8H ;SBC80P OUTPUT TO CONSOLE SUB
  0212               29 EXIT  SET 0212H ;    "  ERROR RECOVERY ROUTINE
  02C2               30 HEX   SET 02C2H ;   " HEX OUTPUT SUB
                     31
                     32 IF NOT SBC80
                     33 CO    SET 0381H  ;IF NOT SBC80P MONITOR
                     34 EXIT  SET 0150H
                     35 ENDIF
                     36
                     37
  3D00               38 ORG 3D00H
                     39
                     40
  3D00 3E80          41    MVI A,MCW1 ;INIT 8255'S
  3D02 D3E7          42    OUT CWR1   ;GROUP 1
  3D04 3E80          43    MVI A,MCW2
  3D06 D3EB          44    OUT CWR2   ;   "   2
  3D08 31FF3F        45    LXI SP,3FFFH ;INIT STACK
  3D0B 3E02          46    MVI A,2 ;SET TEST PASS CTR
  3D0D F5            47    PUSH PSW ;&SAVE
                     48
                     49 GO:
  3D0E 110006        50    LXI D,SETFW ;  (D=PORT CTR E=PORT ID) FORWARD TEST
  3D11 0600          51    MVI B,0       ;0=FWD 1=REVERSE  TEST DIRECTION
  3D13 21D13D        52    LXI H,LOWW    ;FWD TEST PATTERN PTR
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                    52 ;
                    54 PASS:
3D16 0E07           55     MVI C,7      ;SET PORT  BIT CTR
                    56 ;
                    57 ;MAIN TEST ROUTINES
                    58 ;
                    59 TEST:
3D18 C5             60     PUSH B       ;SAVE TEST DIR
3D19 01AD3D         61     LXI B,CHECK  ;SIMULATE CALL
3D1C C5             62     PUSH B
                    63
                    64 PORT0:   ;PORT 0 ROUTINE
3D1D AF             65     XRA A
3D1E 93             66     SUB E ;ID=PORT 0?
3D1F C22B3D         67     JNZ PORT1
3D22 7E             68     MOV A,M ;YES,GET PATTERN
3D23 D3E4           69     OUT POT0 ;OUTPUT TO PORT 0
                    70
3D25 CDC23D         71     CALL DELAY ;DELAY FOR .5 SEC
3D28 DBE4           72     IN POT0 ;READ BACK PATTERN
3D2A C9             73     RET     ;GO CHECK FOR MATCH
                    74 ;
                    75 PORT1:    ;PORT 1 ROUTINE
3D2B 3E01           76     MVI A,1  ;SEE PORT 0 ROUTINE FOR DESCRIPTION
3D2D 92             77     SUB E
3D2E C23A3D         78     JNZ PORT2
3D31 7E             79     MOV A,M
3D32 D3E5           80     OUT POT1
                    81
3D34 CDC23D         82     CALL DELAY
3D37 DBE5           83     IN POT1
3D39 C9             84     RET
                    85 ;
                    86 PORT2:   ;   "    "    "    "   "
3D3A 3E02           87     MVI A,2
3D3C 93             88     SUB E
3D3D C2493D         89     JNZ PORT3
3D40 7E             90     MOV A,M
3D41 D3E6           91     OUT POT2
                    92
3D43 CDC23D         93     CALL DELAY
3D46 DBE6           94     IN POT2
3D48 C9             95     RET
                    96 ;                ;
                    97 PORT3:    ;   "   "   "   "   "
3D49 3E03           98     MVI A,3
3D4B 92             99     SUB E
3D4C C2583D        100     JNZ PORT4
3D4F 7E            101     MOV A,M
3D50 D3E8          102     OUT POT3
                   103
3D52 CDC23D        104     CALL DELAY
3D55 DBE8          105     IN POT3
3D57 C9            106     RET
                   107 ;
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                   108 PORT4:   ;SEE PORT 0 ROUTINE FOR DESCRIPTION
3D58 3E04          109     MVI A,4
3D5A 93            110     SUB E
3D5B C2673D        111     JNZ PORT5
3D5E 7E            112     MOV A,M
3D5F D3E9          113     OUT POT4
                   114
3D61 CDC23D        115     CALL DELAY
3D64 DBE9          116     IN POT4
3D66 C9            117     RET
                   118 ;
                   119 PORT5:   ;DEFAULT TO TEST PORT 5
3D67 7E            120     MOV A,M
3D68 D3EA          121     OUT POT5
                   122
3D6A CDC23D        123     CALL DELAY
3D6D DBEA          124     IN POT5
3D6F C9            125     RET
                   126 ;
                   127 INDEX:
3D70 C1            128     POP B  ;RESTORE TEST DIR
3D71 0D            129     DCR C  ;DEC PORT BIT CTR
3D72 CA823D        130     JZ NEXT ;INDEX PORT CTR?
3D75 78            131     MOV A,B  ;DIRECTION?
3D76 A7            132     ANA A
3D77 C27E3D        133     JNZ BACK ;FORWARD?
3D7A 23            134     INX H    ;YES,INDEX PATTERN PTR
3D7B C3183D        135     JMP TEST  ;CONT FWD TEST
                   136 ;
                   137
                   138 BACK:
3D7E 2B            139     DCX H ;BACKWARD TEST DEC PATTERN PTR
3D7F C31D3D        140     JMP PORT0  ;CONT TEST
                   141
                   142 NEXT:
3D82 78            143     MOV A,B  ;DIRECTION?
3D83 A7            144     ANA A
3D84 CA8B3D        145     JZ INC  ;IF FWD INDEX TO HIGHER PORT
3D87 1D            146     DCR E  ;ELSE DEC TO LOWER PORT
3D88 C38C3D        147     JMP INC+1
                   148
                   149 INC:
3D8B 1C            150     INR E  ;INDEX TO NEXT PORT
3D8C 15            151     DCR D  ;DEC PORT CTR
3D8D CA9B3D        152     JZ DONE? ;IF ALL PORT TESTED CHANGE DIRECTION
3D90 78            153     MOV A,B  ;DIRECTION?
3D91 A7            154     ANA A
3D92 C2133D        155    JNZ GO+5  ;FWD,RESET FWD PATTERN PTR
                   156
                   157
                   158 BKWD:
3D95 21D03D        159     LXI H,HII ;REVERSE PATTERN TEST PTR
3D98 C3163D        160     JMP PASS  ;GO RESET PORT BIT CTR
                   161
                   162 DONE?:
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

3D9B F1            163     POP PSW    ;RESTORE TEST PASS CTR
3D9C 3D            164     DCR A   ;2 TEST PASSES?
3D9D F5            165     PUSH PSW   ;SAVE TEST PASS CTR FOR NEXT PASS
3D9E CA1202        166     JZ EXIT ;RETURN TO MONITOR IF DONE
                   167
3DA1 78            168     MOV A,B   ;DIRECTION?
3DA2 A7            169     ANA A
3DA3 C20E3D        170     JNZ GO   ;BRANCH TO GO IF REVERSE
3DA6 43            171     MOV B,E ;PREV WAS FWD
3DA7 110506        172     LXI D,SETBK ;REVERSE TEST PATTERN
3DAA C3953D        173     JMP BKWD   ;GO TO REVERSE TEST
                   174
                   175 ;CHECK FOR PATTERN MATCH
                   176 ;IF ERROR ENCOUNTERED, PRINT OUT PORT#
                   177 ;  ERROR BIT PATTERN, & RETURN CONTROL TO MONITOR
                   178 ;
                   179 CHECK:
3DAD BE            180     CMP M   ;MATCH?
3DAE CA703D        181     JZ INDEX ;INDEX TO NEXT BIT IF MATCH
                   182
                   183
                   184 IF SBC80
3DB1 F5            185     PUSH PSW ;ERROR,SAVE ERROR PATTERN
3DB2 7B            186     MOV A,E
3DB3 CDC202        187     CALL HEX   ;PRINT PORT#
3DB6 0E20          188     MVI C,' '
3DB8 CDE801        189     CALL CO   ;   "  SPACE
3DBB F1            190     POP PSW
3DBC CDC202        191     CALL HEX   ;   " ERROR PATTERN
3DBF C31202        192     JMP EXIT ;RETURN CONTROL TO MONITOR
                   193 ENDIF
                   194
                   195 IF NOT SBC80
                   196     PUSH PSW
                   197     MOV A,E
                   198     RST 4
                   199     MVI C,' '
                   200     CALL CO
                   201     POP PSW
                   202     RST 4
                   203     JMP EXIT
                   204 ENDIF
                   205
                   206
                   207 ;
                   208 ;.5 SEC DELAY SUBROUTINE
                   209
                   210 DELAY:
3DC2 C5            211     PUSH B
3DC3 0EFF          212     MVI C,0FFH ;SET 500MS  CONSTANT
                   213
                   214 DLY1:
3DC5 06FF          215     MVI B,0FFH ;  "  2MS     "
                   216
                   217 DLY2:
```

```
  LOC  OBJ          SEQ          SOURCE STATEMENT

  3DC7 05            218     DCR B
  3DC8 C2C73D        219     JNZ DLY2  ;2 MS?
  3DCB 0D            220     DCR C
  3DCC C2C53D        221     JNZ DLY1  ;500 MS?
  3DCF C1            222     POP B
  3DD0 C9            223     RET
                     224 ;
                     225
                     226 ;TEST PATTERNS
                     227
  3DD1 0001          228 LOWW: DW 0100H
  3DD3 0307          229       DW 0703H
  3DD5 0F1F          230       DW 1F0FH
  3DD7 3F            231       DB 3FH
  3DD8 7FFF          232 HII: DW 0FF7FH
                     233
                     234 END
```

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USER SYMBOLS

BACK	A 3D7E	BKWD	A 3D95	CHECK	A 3DAD	CO	A 01E8	CWR1	A 00E
DLY1	A 3DC5	DLY2	A 3DC7	DONE?	A 3D9B	EXIT	A 0212	GO	A 3D0
INC	A 3D8B	INDEX	A 3D70	LOWW	A 3DD1	MCW1	A 0080	MCW2	A 008
PORT0	A 3D1D	PORT1	A 3D2B	PORT2	A 3D3A	PORT3	A 3D49	PORT4	A 3D5
POT1	A 00E5	POT2	A 00E6	POT3	A 00E8	POT4	A 00E9	POT5	A 00E
SETFW	A 0600	TEST	A 3D18						

ASSEMBLY COMPLETE, NO ERRORS

Ref. #AA18

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/4040 ☐ 8008 ☒ 8080 ☐ 8048 ☐ 8085 ☒ Other µScope 820 (use additional sheets if necessary)

Program Title

µScope™ 820 Test Instrument, iSBC™ 80/10 Diagnostic Program

Function

Performs GO/NO GO test and diagnostics on all iSBC™ 80/10 functional areas: CPU, RAM, ROM, SERIAL I/O, PARALLEL I/O OFF-BOARD RAM, & SELF-CHECK.

Required Hardware

iSBC™ 80/10, µScope™ 820 Microprocessor System Console, 8080A µScope Probe

Required Software

2716 EPROM with Diagnostic Program

Input Parameters

None

Output Results

See attached page

Available on non-system diskette only--$35.00. Source object and list file included The Applications Note AP-42 also included with program.

Registers Modified: N/A	Programmer:
RAM Required: N/A	Company:
ROM Required: N/A	Address:
Maximum Subroutine Nesting Level: N/A	City:
Assembler/Compiler Used: N/A	State:

µSCOPE™ 820 TEST INSTRUMENT, iSBC™ 80/10 DIAGNOSTIC PROGRAM

This program used in conjunction with Intel's µScope™ 820 Test Instrument dramatically reduces the costs to support microprocessor based systems in the field. For example, using this program a field service technician can walk up to a down iSBC™ 80/10 based computer system and,

1st with less than 10 key strokes uniquely configure the test program to the System's Options,

2nd run a complete CPU exercise with diagnostics to make sure the heart of the system is fine,

3rd test each of the major areas of the system and provide diagnostics easily understood by the technicians, and

4th provide advanced capabilities for finding intermittents and providing fault isolation to the compontent level.

The tests performed include:

- CPU at System Clock Rates
- RAM and ROM up to 64K
- Serial, Parallel, and Multibus™ I/O
- Self-Check on the Test Program itself

The program allows the user to create a 2716 EPROM which plugs into the socket on the µScope™ 820 Console front panel. The iSBC™ 80/10 Diagnostic Program is written to allow the user to easily modify sections to customize the program for his unique requirements. The program is heavily commented to aid the user. A more detailed discussion of the program is presented in Intel's Applications Note #42,"Writing Diagnostics with the µScope™ 820 Microprocessor System Console".

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 ☒8080 ☐8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

IOTEST - I/O Test Program for the SBC 80/20

Function

To exercise all output lines of the 8255's and the 8251 USART for a teletype interface on an SBC 80/20 board.

Required Hardware

An SBC 80/20 board and a teletype adapter such as the SBC 530. The program can be easily configured for a CRT at a higher baud rate.

Required Software

NONE

Input Parameters

NONE

Output Results

Each 8 bit output port will operate as an 8 bit synchronous up counter allowing all I/O lines and drivers to be checked as well as testing the interface to the logic external to the board.
The USART will transmit an ASCII 0 continuously until another character is received after which it will then repeat the new character. The tape reader control line is also toggled.

Registers Modified: A,B,C,H,L	Programmer: Lee Mandell
RAM Required: NONE	Company: LJM Associates
ROM Required: 77 Bytes	Address: 6331 Glade Ave., Suite 318
Maximum Subroutine Nesting Level: 0	City: Woodland Hills
Assembler/Compiler Used: ISIS II 8080/8085 MACRO V2.0	State: California 91367

```
ASM80 :F1:IOTEST

ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0          IOTEST   PAGE    1

 LOC  OBJ          SEQ          SOURCE STATEMENT

                     1 ; I/O TEST PROGRAM FOR THE SBC 80/20
                     2          NAME IOTEST
 3800                3 RAM      EQU 3800H
 00ED                4 TTYRP2   EQU 0EDH
                     5 ; UART INITIALIZATION FOR TELETYPE
 0000 3ECE           6 START:   MVI A,0CEH
 0002 D3ED           7          OUT 0EDH
 0004 3E27           8          MVI A,27H
 0006 D3ED           9          OUT 0EDH
                    10 ; 8251 INITIALIZATION - 1.76 KHZ FOR TIMER #2
 0008 3EB6          11          MVI A,0B6H
 000A D3DF          12          OUT 0DFH
 000C 3E63          13          MVI A,63H
 000E D3DE          14          OUT 0DEH
 0010 3E02          15          MVI A,02H
 0012 D3DE          16          OUT 0DEH
                    17 ; OUTPUT PORTS INITIALIZATION
 0014 3E80          18          MVI A,80H
 0016 D3E7          19          OUT 0E7H
 0018 D3EB          20          OUT 0EBH
                    21 ;
                    22 ; OUTPUT 0 FROM TELETYPE CONTINUOUSLY
                    23 ; UNLESS CHARACTER FROM TELETYPE ENTERED
                    24 ; THEN TRANSMIT NEW CHARACTER CONTINUOUSLY
                    25 ;
                    26 ; TOGGLE READER CONTROL AT 10 HZ
                    27 ; TOGGLE ALL OTHER I/O AT LOOP RATE
                    28 ;
 001A 013027        29 LOOP0:  LXI B,2730H
 001D 210000        30          LXI H,0
 0020 23            31 LOOP:   INX H
 0021 7D            32          MOV A,L
 0022 D3E4          33          OUT 0E4H
 0024 D3E5          34          OUT 0E5H
 0026 D3E6          35          OUT 0E6H
 0028 D3E8          36          OUT 0E8H
 002A D3E9          37          OUT 0E9H
 002C D3EA          38          OUT 0EAH
 002E DBED          39          IN TTYRP2
 0030 E601          40          ANI 1
 0032 CA3E00        41          JZ CONT1
 0035 78            42          MOV A,B
 0036 EE02          43          XRI 2
 0038 47            44          MOV B,A
 0039 D3ED          45          OUT TTYRP2
 003B 79            46          MOV A,C
 003C D3EC          47          OUT TTYRP2-1
 003E DBED          48 CONT1:  IN TTYRP2
 0040 E602          49          ANI 2
 0042 CA2000        50          JZ LOOP
 0045 DBEC          51          IN TTYRP2-1
 0047 E67F          52          ANI 7FH
```

```
  LOC  OBJ          SEQ          SOURCE STATEMENT

  0049 4F             53          MOV C,A
  004A C32000         54          JMP LOOP
                      55          END

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USER SYMBOLS
CONT1  A 003E    LOOP   A 0020    LOOP0  A 001A    RAM    A 3800    START  A 000

ASSEMBLY COMPLETE,  NO ERRORS
```

Ref.#AB117

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004 ☐4040 ☐8008 ☒8080 ☐3000 ☐Other_____ (use additional sheets if necessary)

Program Title

CUT-AND-PAST EDITOR (PL/M)

Function

This program provides cut-and-paste capability to augment the MDS-800 ISIS II editor and is nearly as fast as a disk to disk copy.

Required Hardware

MDS-800 Development System with floppy disk and full 64K RAM. (Can redefine buffer dimension for smaller RAM memory.)

Required Software

ISIS II Operating System (SYSTEM LIB and PLM 80 LIB also required for linking).

Input Parameters

1. Input file name.
2. Input text file with insert and block delineators edited in.

Output Results

1. Output file with blocks of data moved around and/or deleted (has original input file name.)
2. Original input file renamed with .BAK extension.

For further description see detailed comments in source listing which include sample paragraph before and after CUTPAS is executed.

Registers Modified: All	Programmer: Dr. A. J. Spuria
RAM Required: 64K [Can configure for 48K or 32K]	Company: Bendix Corporation Communications Division
ROM Required: None	Address: E. Joppa Road
Maximum Subroutine Nesting Level: N/A	City: Towson
Assembler/Compiler Used: PL/M	State: Maryland 21204

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE CUTPAS
OBJECT MODULE PLACED IN :F1:CUTPAS.OBJ
COMPILER INVOKED BY: PLM80 :F1:CUTPAS.PLM

```
/*
   CUT AND PASTE CAPABILITY TO AUGMENT THE MDS-800 EDITOR.
THE EDITOR IS USED IN CONJUNCTION WITH THIS PROGRAM TO ALLOW BLOCKS
OF ASCII DATA TO BE MOVED FROM ONE LOCATION AND INSERTED AT ONE OR
MORE DIFFERENT LOCATIONS. THE ORIGINAL DATA WILL BE DELETED FROM THE
ORIGINAL LOCATION UNLESS AN INSERT COMMAND IS PLACED THERE. IF NO INSERT
COMMANDS ARE GIVEN FOR A BLOCK, IT IS MERELY DELETED AND NOT INSERTED
ANYWHERE. UP TO 9 LEVELS OF NESTING ARE ALLOWED.

    IN ORDER TO USE CUTPAS IT IS FIRST NECESSARY TO USE THE EDITOR
TO PLACE AN INSERT-HERE DELINEATOR CHARACTER AND A BLOCK DELINEATOR
CHARACTER AS THE FIRST TWO CHARACTERS IN THE FILE TO BE CUT AND PASTED.
IN THE FOLLOWING THESE WILL BE ASSUMED TO BE @ AND \ RESPECTIVELY FOR
PURPOSES OF ILLUSTRATION. AFTER DEFINING THE DELINEATORS, IT IS NECESSARY
TO PUT \N BEFORE AND AFTER EACH BLOCK TO BE MOVED AND @N EACH PLACE IT
IS TO BE INSERTED, WHERE N IS ANY ASCII CHARACTER. AFTER EXITING THE
EDITOR, INVOKE THE CUT AND PASTE CAPABILITY BY:

       CUTPAS :F1:FILNAM.EXT

 NOTE: THERE MUST BE EXACTLY 1 BLANK AFTER CUTPAS. THE :F1: MUST BE OMITTED
IF THE FILE IS ON DRIVE 0. THE FILE EXTENSION EXT CANNOT BE TMP OR BAK. THE FILE
FILNAM.EXT WILL BE RENAMED FILNAM.BAK UPON COMPLETION AND THE CUT-AND-PASTED
FILE WILL BE NAMED FILNAM.EXT. THIS PROGRAM REQUIRES A
65,536 MDS MEMORY COMPLEMENT SINCE THE WHOLE INPUT FILE IS HANDLED IN RAM
TO MINIMIZE DISK ACCESS. THE LARGEST FILE THAT CAN BE CUT AND PASTED IS
48,000 CHARACTERS INCLUDING @N AND \N MARKERS.

   THE FOLLOWING PARAGRAPH ILLUSTRATES THE USE OF THE MARKERS
IN A FILE TO BE CUT AND PASTED. IT ALSO ILLUSTRATES THE USE OF NESTED BLOCKS
ALTHOUGH THIS CAN BECOME QUITE COMPLICATED:

PARAGRAPH BEFORE USING CUTPAS:

   @2\1THIS SENTENCE WILL BE MOVED TO THE MIDDLE OF THE PARAGRAPH.\1\X
THIS SENTENCE WILL BE DELETED.
\X\2PART OF THIS SENTENCE \XWILL BE DELETED, PART \XWILL BE MOVED TO @3
\3THE END OF THE PARAGRAPH\3AND THE WHOLE SENTENCE EXCEPT FOR THE DELETED
PART WILL APPEAR AT THE START OF THE PARAGRAPH. \2
@1
NOTE THAT THE USER MUST KEEP TRACK OF CARRIAGE RETURNS AND LINE FEEDS
AND PLACE HIS BLOCK DELINEATORS RESPECTIVELY.
@3.

PARAGRAPH AFTER USING CUTPAS:

   PART OF THIS SENTENCE WILL BE MOVED TO THE END OF THE PARAGRAPH
AND THE WHOLE SENTENCE EXCEPT FOR THE DELETED
PART WILL APPEAR AT THE START OF THE PARAGRAPH.
THIS SENTENCE WILL BE MOVED TO THE MIDDLE OF THE PARAGRAPH.
NOTE THAT THE USER MUST KEEP TRACK OF CARRIAGE RETURNS AND LINE FEEDS
```

```
            AND PLACE HIS BLOCK DELINEATORS RESPECTIVELY.
            THE END OF THE PARAGRAPH.

            ERROR NUMBERS REPORTED ARE AS FOLLOWS:
                101=ERROR IN READING :CI: TO GET FILE NAME
                102=ERROR IN OPENING INPUT FILE
                103=ERROR IN OPENING OUTPUT FILE
                104=ERROR IN READING INPUT FILE
                105=ERROR IN WRITING OUTPUT FILE
                106=ERROR IN CLOSING INPUT FILE
                107=ERROR IN CLOSING OUTPUT FILE
                108=ERROR IN RENAMING :F1:FILNAM.EXT TO :F1:FILNAM.BAK
                109=ERROR IN RENAMING :F1:FILNAM.TMP TO :F1:FILNAM.EXT
                196=FOUND END OF FILE MARK WHILE SEARCHING FOR SEGMENT TO INSERT
                197=FOUND END OF FILE MARK WHILE SEARCHING FOR END OF A BLOCK
                198=NESTING DEEPER THAN 9 LEVELS
                199=INPUT FILE LARGER THAN 15,560 CHARACTERS
             ERROR NUMBERS 101 TO 109 ARE FOLLOWED BY THE CORRESPONDING ISIS II ERROR NUMBER
             */
1            CUTPAS: DO;
             $INCLUDE(:F1:ABBREV)
2   1  =     DECLARE AS LITERALLY 'LITERALLY';
3   1  =     DECLARE DCL AS 'DECLARE';
4   1  =     DCL PROC AS 'PROCEDURE';
5   1  =     DCL PUB AS 'PUBLIC';
6   1  =     DCL EXT AS 'EXTERNAL';
7   1  =     DCL STR AS 'STRUCTURE';
8   1  =     DCL ADR AS 'ADDRESS';
9   1  =     DCL CSR1 AS '20H';
10  1  =     DCL CSR2 AS '28H';
11  1  =     DCL RELAYS AS '21H';
12  1  =     DCL PASSB AS '22H';
13  1  =     DCL RESULT AS '23H';
14  1  =     DCL T2LOW AS '2CH';
15  1  =     DCL T2HIGH AS '2DH';
16  1  =     DCL IN AS 'INPUT';
17  1  =     DCL OUT AS 'OUTPUT';
18  1  =     DCL RET AS 'RETURN';
19  1  =     DCL EN AS 'ENABLE';
20  1  =     DCL DIS AS 'DISABLE';
21  1  =     DCL POP AS 'STACKPTR=STACKPTR+2';
22  1  =     DCL RTI AS 'DO;ENABLE;RETURN;END;';
              /*BUFLEN MUST BE MODIFIED FOR SMALLER MEMORY CONFIGURATIONS*/
23  1        DCL BUFLEN AS '48010';
24  1        DCL BUFFER(BUFLEN)BYTE;
25  1        DCL (ACTUAL,STATUS,AFT$IN,AFT$OUT,M,BUFPTR,START,COUNT)ADR;
26  1        DCL BUFBYT BASED BUFPTR BYTE;
27  1        DCL (LEVEL,N,AT$SIGN,BACK$SLASH)BYTE;
28  1        DCL I ADR;
29  1        DCL STAK(11)STR(START ADR,N BYTE);
30  1        DCL ORIGNAME(15)BYTE;
31  1        DCL NEWNAME(15)BYTE;
32  1        DCL BACKNAME(15)BYTE;
33  1        DCL BLANK AS '20H';
34  1        DCL CR AS '13';
35  1        DCL PERIOD AS '2EH';
36  1        DCL EOF$MARK AS '4';
```

```
 37  1      DCL CHECK$STATUS AS 'IF STATUS<>0 THEN DO;CALL ERROR(M);CALL ERROR(STATUS);CALL EXIT;END';
 38  1      OPEN: PROC(AFT,F,ACC,MODE,S)EXT;
 39  2          DCL(AFT,F,ACC,MODE,S)ADR;
 40  2      END OPEN;
 41  1      CLOSE: PROC(AFT,S)EXT;
 42  2          DCL (AFT,S)ADR;
 43  2      END CLOSE;
 44  1      READ: PROC(AFT,BUF,CNT,ACT,S)EXT;
 45  2          DCL (AFT,BUF,CNT,ACT,S)ADR;
 46  2      END READ;
 47  1      WRITE: PROC(AFT,BUF,CNT,S)EXT;
 48  2          DCL (AFT,BUF,CNT,S)ADR;
 49  2      END WRITE;
 50  1      RENAME: PROC(OLD,NEW,S)EXT;
 51  2          DCL (OLD,NEW,S)ADR;
 52  2      END RENAME;
 53  1      DELETE: PROC(NAME,S)EXT;
 54  2          DCL(NAME,S)ADR;
 55  2      END DELETE;
 56  1      EXIT: PROC EXT;
 57  2      END EXIT;
 58  1      ERROR: PROC(ERRNUM)EXT;
 59  2          DCL ERRNUM ADR;
 60  2      END ERROR;

          /*READ :CI: TO GET FILE NAME*/

 61  1      CALL READ(1,.BUFFER,128,.ACTUAL,.STATUS);
 62  1      M=101;CHECK$STATUS;

          /*OPEN :F1:FILNAM.EXT FOR INPUT*/

 69  1      CALL OPEN(.AFT$IN,.BUFFER,1,0,.STATUS);
 70  1      M=102;CHECK$STATUS;

          /*SAVE ORIGINAL FILE NAME*/

 77  1      CALL MOVE(15,.BUFFER+1,.ORIGNAME);



          /*SETUP OUTPUT FILE NAME IN NEWNAME AND BACKUP FILE NAME IN BACKNAME*/

 78  1      DO I=2 TO 11;
 79  2          IF (BUFFER(I)=BLANK) OR (BUFFER(I)=CR) THEN BUFFER(I)=PERIOD;
 81  2          IF BUFFER(I)=PERIOD THEN GO TO OPEN$OUT;
 83  2      END;
 84  1      OPEN$OUT: BUFFER(I+1)='T';BUFFER(I+2)='M';BUFFER(I+3)='P';BUFFER(I+4)=BLANK;
 88  1          CALL MOVE(15,.BUFFER+1,.NEWNAME);

          /*SETUP BACKUP FILE NAME IN BACKNAME*/

 89  1          BUFFER(I+1)='B';BUFFER(I+2)='A';BUFFER(I+3)='K';
 92  1          CALL MOVE(15,.BUFFER+1,.BACKNAME);

         /*OPEN FILE :F1:FILNAM.TMP FOR OUTPUT*/
```

```
 93  1          CALL OPEN(.AFT$OUT,.NEWNAME,2,0,.STATUS);
 94  1          M=103;CHECK$STATUS;

            /*READ WHOLE INPUT FILE INTO BUFFER*/

101  1      BUFPTR=.BUFFER;
102  1      READ$IN: CALL READ(AFT$IN,BUFPTR,128,.ACTUAL,.STATUS);
103  1          M=104;CHECK$STATUS;
110  1          IF ACTUAL<128 THEN GO TO START$SEARCH;
112  1          BUFPTR=BUFPTR+128;

            /*CHECK FOR FILE LARGER THAN BUFFER SIZE*/

113  1          IF BUFPTR>.BUFFER(BUFLEN-132)THEN
114  1              DO;
115  2              CALL ERROR(199);
116  2              CALL EXIT;
117  2              END;
118  1          GO TO READ$IN;

            /*SETUP SEARCH PARAMETERS AND PUT END OF FILE MARK AT END OF BUFFER*/

119  1      START$SEARCH: START=.BUFFER+2;
120  1          AT$SIGN=BUFFER(0);
121  1          BACK$SLASH=BUFFER(1);
122  1          BUFPTR=BUFPTR+ACTUAL;
123  1          BUFBYT=EOF$MARK;
124  1          LEVEL=0;

            /*START SEARCH FOR @, \ OR END OF FILE MARK*/

125  1      SEARCH: BUFPTR=START;
126  1          DO I=0 TO BUFLEN;
127  2              IF (BUFBYT=AT$SIGN) OR (BUFBYT=BACK$SLASH) OR (BUFBYT=EOF$MARK)
                        THEN GO TO WRITE$OUT;
129  2              BUFPTR=BUFPTR+1;
130  2          END;

            /*WRITE STUFF OUT UP TO MARKER FOUND*/

13:L 1      WRITE$OUT: COUNT=BUFPTR-START;
132  1          CALL WRITE(AFT$OUT,START,COUNT,.STATUS);
133  1          M=105;CHECK$STATUS;

            /*NOW PROCESS INTERVAL DEPENDING ON MARKER FOUND*/

140  1          IF BUFBYT=EOF$MARK THEN GO TO RENAME$FILES;
142  1          IF BUFBYT=AT$SIGN THEN GO TO INSERT$STUFF;

            /*IF BUFBYT=BACK$SLASH THEN PROCESS AS FOLLOWS*/

144  1          BUFPTR=BUFPTR+1;
            /*IF AT LEVEL 0 THEN SKIP \N...\N BLOCK*/

145  1          IF LEVEL=0 THEN GO TO SKIP$BLOCK;

            /*AT LEVEL>0, SKIP BLOCK \N...\N IF AND ONLY IF N DOES NOT CORRESPOND
```

```
                WITH @N FOR THE PREVIOUS LEVEL*/

147   1         IF BUFBYT<>STAK(LEVEL-1).N THEN GO TO SKIP$BLOCK;

               /*FOUND ENDING \N, SO DECREMENT LEVEL CTR AND START SEARCHING
                  AFTER @N*/

149   1          LEVEL=LEVEL-1;
150   1         START=STAK(LEVEL).START;
151   1         GO TO SEARCH;

              /*TO SKIP BLOCK, POINT TO CHAR FOLLOWING ENDING \N*/

152   1      SKIP$BLOCK: N=BUFBYT;
153   1          DO I=1 TO BUFLEN;
154   2          BUFPTR=BUFPTR+1;
155   2          IF BUFBYT=EOF$MARK THEN
156   2              DO;CALL ERROR(197);CALL EXIT;END;
160   2          IF BUFBYT<>BACK$SLASH THEN GO TO END$ILOOP;
162   2          BUFPTR=BUFPTR+1;
163   2          IF BUFBYT=N THEN GO TO GOT$N;
165   2      END$ILOOP: END;

              /*NOW GO START SEARCHING FOR NEXT MARKER*/

166   1      GOT$N:  START=BUFPTR+1;
167   1          GO TO SEARCH;

              /*INSERT STUFF FROM \N...\N INTO @N POSITION*/
              /*SAVE START AFTER @N AND SAVE N ON STAK AND INCR LEVEL CTR*/

168   1      INSERT$STUFF: BUFPTR=BUFPTR+1;
169   1          STAK(LEVEL).START=BUFPTR+1;
170   1          STAK(LEVEL).N,N=BUFBYT;
171   1          LEVEL=LEVEL+1;
172   1          IF LEVEL>10 THEN DO;CALL ERROR(198);CALL EXIT;END;

              /*FIND SEGMENT \N...\N*/

177   1          BUFPTR=.BUFFER+2;
178   1          DO I=0 TO BUFLEN;
179   2              IF BUFBYT=EOF$MARK THEN
180   2                  DO;CALL ERROR(196);CALL EXIT;END;
184   2              IF BUFBYT<>BACK$SLASH THEN GO TO END$LOOP2;
186   2              BUFPTR=BUFPTR+1;
187   2              IF BUFBYT=N THEN GO TO GOT$SEG;
189   2      ENDLOOP2: BUFPTR=BUFPTR+1;
190   2          END;

              /*FOUND SEGMENT...NOW GO SEARCH FOR NEXT MARKER*/

191   1      GOT$SEG: START=BUFPTR+1;
192   1          GO TO SEARCH;

              /*REACHED END OF FILE...CLOSE OUT FILE AND RENAME THEM*/

193   1      RENAME$FILES: CALL CLOSE(AFT$IN,.STATUS);
```

```
194  1          M=106;CHECK$STATUS;
201  1          CALL CLOSE(AFT$OUT,.STATUS);
202  1          M=107;CHECK$STATUS;
209  1          CALL DELETE(.BACKNAME,.STATUS);
210  1          CALL RENAME(.ORIGNAME,.BACKNAME,.STATUS);
211  1          M=108;CHECK$STATUS;
218  1          CALL RENAME(.NEWNAME,.ORIGNAME,.STATUS);
219  1          M=109;CHECK$STATUS;
226  1      CALL EXIT;
227  1      END CUTPAS;


MODULE INFORMATION:

     CODE AREA SIZE     = 0505H   1285D
     VARIABLE AREA SIZE = BBEEH  48110D
     MAXIMUM STACK SIZE = 0008H      8D
     289 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

Ref.#AB118

insite™ INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/40 ☐8008 ☒8080 ☐8048 ☐8085 ☐3000 ☐Other_____ (use additional sheets if necessary)

Program Title

Hewlett Packard Calculator to MDS800 I/O Control Program-HPIO

Function

This program inputs and outputs data and instructions between Hewlett Packard 9815 programmable calculator and MDS 800 memory.

Required Hardware

RS232 interface for HP I/O.
TTY with current loop interface for program control.

Required Software

MDS Monitor

Input Parameters

To output to the HP calculator, data and instructions must be in memory starting at 7000H (relocatable if desired).

Output Results

When inputting from HP calculator, memory will be load with data and instructions starting at 7000H (relocatable if desired).

Registers Modified: A,C,D,E,H,L	Programmer: John E. Kiesling
RAM Required: 100D - Program + Data Storage Space	Company: Quality Measurement Systems
ROM Required: MDS-Monitor	Address: 2555 Baird Road
Maximum Subroutine Nesting Level: 1	City: Penfield,
Assembler/Compiler Used: 8080/8085 Assembler V1.0	State: New York 14526

```
ASM80 :F1:HPIO

ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0          HPIO      PAGE    1

 LOC  OBJ          SEQ          SOURCE STATEMENT

                     1           NAME    HPIO
 5000                2           ORG     5000H
 00F7                3 CRTC      EQU     0F7H
 F803                4 CI        EQU     0F803H
 F809                5 CO        EQU     0F809H
                     6 ;*************************************************************
                     7 ;
                     8 ;     INPUT DATA FROM HEWLETT PACKARD 9815 CALCULATOR
                     9 ;
                    10 ;*************************************************************
 5000 210070        11 START1: LXI      H,7000H          ;SET H AND L TO
                    12                                   ;THE BEGINNING OF THE
                    13                                   ;PROGRAM STORAGE SPACE
 5003 16FF          14          MVI     D,0FFH           ;REG D IS THE CHECK
                    15                                   ;SUM FLAG
 5005 3E81          16 START:   MVI     A,81H            ;TRANSFER IO CONTROL
                    17                                   ;TO RS232 PORT
 5007 320300        18          STA     03H
 500A 3E05          19          MVI     A,5H             ;SEND READY FOR DATA
 500C D3F7          20          OUT     CRTC             ;SIGNAL
 500E CD03F8        21          CALL    CI               ;INPUT DATA FROM HP
 5011 4F            22          MOV     C,A              ;MOVE NEW DATA IN REG C
 5012 3E07          23          MVI     A,7H
 5014 D3F7          24          OUT     CRTC             ;SEND MDS BUSY SIGNAL
 5016 3E80          25          MVI     A,80H
 5018 320300        26          STA     03H              ;TRANSFER IO CONTROL
                    27                                   ;BACK TO TTY
 501B 71            28          MOV     M,C              ;STORE NEW DATA IN
                    29                                   ;MEMORY SRACE DESIGNATED
                    30                                   ;SPACE DESIGNATED BY
                    31                                   ;BY H AND L
 501C 3EB1          32          MVI     A,0B1H
 501E B9            33          CMP     C                ;IF NEW DATA = B1
                    34                                   ;(CODE FOR END)
                    35                                   ;GO TO ENDF
 501F CA2C50        36          JZ      ENDF
 5022 23            37          INX     H                ;H AND L NOW POINT
                    38                                   ;TO THE NEXT STORAGE
                    39                                   ;SPACE
 5023 3E00          40          MVI     A,00H
 5025 BA            41          CMP     D                ;IF D = 00 GO TO CK
 5026 CA3450        42          JZ      CK
 5029 C30550        43          JMP     START
 502C 1600          44 ENDF:    MVI     D,00H            ;SET REG D = 00 TO
                    45                                   ;INDICATE TWO BYTES OF
                    46                                   ;DATA MUST BE PICKED UP
                    47                                   ;THESE BYTES ARE THE
                    48                                   ;CHECK SUM INFO
 502E 1E02          49          MVI     E,02H            ;SET REG E = 2
                    50                                   ;THIS MAKES SURE THAT
                    51                                   ;THE CHECK SUM IS
                    52                                   ;PICKED UP
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

5030 23             53           INX     H
5031 C30550         54           JMP     START
5034 1D             55 CK:       DCR     E                ;IF REG E = 0 ALL
                    56                                    ;BYTES ARE NOW IN
                    57                                    ;THEREFORE THE ROUTINE
                    58                                    ;IS DONE
5035 C20550         59           JNZ     START
5038 00             60 DONE:     NOP
5039 C7             61           RST     0
                    62 ;*****************************************************
                    63 ;
                    64 ;      OUTPUT DATA TO HEWLETT PACKARD 9815 CALCULATOR
                    65 ;
                    66 ;*****************************************************
6000                67           ORG     6000H
6000 210070         68 START2:   LXI     H,7000H          ;SET H AND L TO THE
                    69                                    ;BEGINNING OF THE
                    70                                    ;DATA TO BE OUTPUT
6003 16FF           71           MVI     D,0FFH           ;SET D = FFH
                    72                                    ;REG D IS THE CHECK SUM
                    73                                    ;FLAG
6005 3E00           74 START3:   MVI     A,00H
6007 320300         75           STA     03H              ;TRANSFER IO CONTROL TO
                    76                                    ;TTY
600A 4E             77           MOV     C,M              ;MOVE DATA INTO REG C
600B 23             78           INX     H                ;H AND L NOW POINT TO
                    79                                    ;THE NEXT PIECE OF DATA
600C 3E01           80           MVI     A,01H
600E 320300         81           STA     03H              ;TRANSFER IO CONTROL TO
                    82                                    ;RS232 INTERFACE
6011 DBF7           83 BCK:      IN      CRTC
6013 E681           84           ANI     81H              ;WAIT FOR DEVICE TO BE
                    85                                    ;NOT BUSY
6015 C21160         86           JNZ     BCK
6018 CD09F8         87           CALL    CO               ;OUTPUT DATA
601B 3EB1           88           MVI     A,0B1H
601D B9             89           CMP     C                ;IF DATA = B1H (CODE
                    90                                    ;FOR END) GO TO
                    91                                    ;ENDF1
601E CA2A60         92           JZ      ENDF1
6021 3E00           93           MVI     A,00H
6023 BA             94           CMP     D                ;IF REG D = 00 GO TO CK1
6024 CA3160         95           JZ      CK1
6027 C30560         96           JMP     START3
602A 1E02           97 ENDF1:    MVI     E,02H            ;SET REG E = 2
                    98                                    ;  THIS COUNTER WILL MAKE
                    99                                    ;SURE THE LAST TWO BYTES
                   100                                    ;(CHECK SUM NUMBERS)
                   101                                    ;AFTER THE END
                   102                                    ;ARE PICK UP
602C 1600          103           MVI     D,00H            ;SET REG D = 00 TO FLAG
                   104                                    ;THAT THE NEXT BYTES ARE
                   105                                    ;CHECK SUM BYTES
602E C30560        106           JMP     START3
6031 1D            107 CK1:      DCR     E                ;IF REG E = 0 ALL
```

```
  LOC  OBJ          SEQ          SOURCE STATEMENT

                     108                                    ;DATA HAS BEEN OUTPUT
  6032 C20560        109          JNZ      START3
  6035 3E01          110 DONE1:   MVI      A,01H
  6037 320300        111          STA      03H
  603A 00            112          NOP
  603B C7            113          RST      0
                     114 ;**************************************************************
                     115 ;
                     116 ;                              END
                     117 ;
                     118 ;**************************************************************
  5000               119          END      5000H
```

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USER SYMBOLS

BCK	A 6011	CI	A F803	CK	A 5034	CK1	A 6031	CO	A F80		
DONE1	A 6035	ENDF	A 502C	ENDF1	A 602A	START	A 5005	START1	A 500		

ASSEMBLY COMPLETE, NO ERRORS

Ref.#AB119

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 ☒8080 ☐8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

HP - Controller for Hewlett-Packard 9871A Printer

Function

To interface a Hewlett-Packard 9871A Printer to an Intel MDS-8ØØ.

Required Hardware

Intel MDS-8ØØ
Intel SBC-5Ø8
Hewlett-Packard 9871A

Required Software

MDS Monitor V2.Ø
MDS Boot V2.Ø
ISIS-II V2.2

Input Parameters

C-Register contains character for output by printer

Printer status on port ØDFH

Output Results

Character to be printed appears on port ØDFH
Printer control on port ØDEH

This program outputs characters to the HP 9871A printer using the handshake routine described in Chapter Two of the 'Hewlett-Packard 9871A Printer Operating and Service Manual'. This manual is Hewlett-Packard Part Number Ø9871-9ØØ3Ø.

Available on diskette only.

Registers Modified: All	Programmer: Carl Owenby, Jr.
RAM Required: ØB3H for Loader; 69H for Driver	Company: Gibson-Owenby, Inc.
ROM Required: None	Address: Post Office Box 973
Maximum Subroutine Nesting Level: Two	City: Quincy,
Assembler/Compiler Used: ISIS-II PL/M 8Ø V3.Ø	State: Florida 32351

Five files are required to compile, link, and locate this program. They are HPLOD.SRC, HPDRV.SRC, HPLIDR.SRC, 9871A.DAT, and HP9871.CSD.

The HP9871.CSD file must be edited according to the amount of memory in the MDS-8ØØ system. For a 16K system, the code parameter of the locate command should be changed to 3EØØH. For a 32K system, the parameter should be 7EØØH; for a 48K system it should be ØBEØØH; and should be left at ØF6ØØH for a system with 64K of RAM.

After making any required modification in the Command Sequence Definition file as described above, the program may be compiled by entering the command 'SUBMIT :F1:HP9871.CSD'. This should be done with the PL/M 8Ø compiler disk in Drive Ø and the program source files on the disk in Drive 1.

After compilation the program may be loaded and executed by entering the command 'HP'.

This interface has been operated successfully operated using a twenty foot length of thirty-four conductor flat cable connected in accordance with the run list in file 9871A.DAT.

```
ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE HP9871ALOADER
OBJECT MODULE PLACED IN :F1:HPLOD.OBJ
COMPILER INVOKED BY:  PLM80 :F1:HPLOD.SRC


            $DATE(30JUL77)
            $TITLE('HPLOD - HP 9871A LOADER')
            $PAGEWIDTH(72)
            $PAGELENGTH(57)

  1         HP$9871A$LOADER:
              DO;


            /*
            THIS PROGRAM EXPECTS THE HEWLETT-PACKARD 9871A PRINTER
            TO BE INTERFACED IN THE FOLLOWING FASHION TO THE MDS-800.
            */

            $INCLUDE(:F1:9871A.DAT)
          =   /*
          =   9871A   DESCRIPTION                SBC-508 DESCRIPTION
          =   PIN #                              PIN #
          =
          =   3       S1 (NOT USED)
          =   4       S0 (NOT USED)
          =   5       IO3/ (COVER ON)            91      I3DAT3/ (DATA BIT
           -   3 TO INPUT PORT 3)
          =   6       IO2/ (ABORT)               92      I3DAT2/ (DATA BIT
           -   2 TO INPUT PORT 3)
          =   7       IO1/ (READY)               89      I3DAT1/ (DATA BIT
           -   1 TO INPUT PORT 3)
          =   8       IO0/ (BUFFER SPACE)        90      I3DAT0/ (DATA BIT
           -   0 TO INPUT PORT 3)
          =   9       CMD/ (CONTROL)             27      O2DAT0/ (DATA BIT
           -   0 FROM OUTPUT PORT 2)
          =   10      HLT/ (STOP)                26      O2DAT1/ (DATA BIT
           -   1 FROM OUTPUT PORT 2)
          =   11      FLG/ (FLAG)                98      I3DAT5/ (DATA BIT
           -   5 TO INPUT PORT 3)
          =   12      CALC7/                     51      O3DAT7/ (DATA BIT
           -   7 FROM OUTPUT PORT 3)
          =   13      CALC6/                     36      O3DAT6/ (DATA BIT
           -   6 FROM OUTPUT PORT 3)
          =   14      CALC5/                     53      O3DAT5/ (DATA BIT
           -   5 FROM OUTPUT PORT 3)
          =   24      POP/ (POWER-ON PRESET)     97      I3DAT4/ (DATA BIT
           -   4 TO INPUT PORT 3)
          =   25      CHASSIS GROUND
          =   26      LOGIC GROUND
          =   27      LOGIC GROUND
          =   28      CALC0/                     39      O3DAT0/ (DATA BIT
           -   0 FROM OUTPUT PORT 3)
```

```
=   29      CALC1/                          38      O3DAT1/ (DATA BIT
 -   1 FROM OUTPUT PORT 3)
=   30      CALC2/                          50      O3DAT2/ (DATA BIT
 -   2 FROM OUTPUT PORT 3)
=   31      CALC3/                          37      O3DAT3/ (DATA BIT
 -   3 FROM OUTPUT PORT 3)
=   32      CALC4/                          52      O3DAT4/ (DATA BIT
 -   4 FROM OUTPUT PORT 3)
=   33      LOGIC GROUND
=   34      LOGIC GROUND
=   35      +5V@500MA
=
=   PORT NUMBER     ADDRESS
=   0               00DCH
=   1               00DDH
=   2               00DEH
=   3               00DFH
=
=
=   FLAT CABLE      9871A           SBC-508         DESCRIPTIO
 -   N
=   WIRE#           PIN#            PIN#
=   1                                               GND
=   2               28              39              CALC0/
=   3                                               GND
=   4               29              38              CALC1/
=   5                                               GND
=   6               30              50              CALC2/
=   7                                               GND
=   8               31              37              CALC3/
=   9                                               GND
=   10              32              52              CALC4/
=   11                                              GND
=   12              14              53              CALC5/
=   13                                              GND
=   14              13              36              CACL6/
=   15                                              GND
=   16              12              51              CALC7/
=   17                                              GND
=   18              9               27              CMD/
=   19                                              GND
=   20              10              26              HLT/
=   21                                              GND
=   22              8               90              IO0/
=   23                                              GND
=   24              7               89              IO1/
=   25                                              GND
=   26              6               92              IO2/
=   27                                              GND
=   28              5               91              IO3/
=   29                                              GND
=   30              11              98              FLG/
=   31                                              GND
=   32              24              97              POP/
```

```
                =   33                                                  GND
                =   34                                                  GND
                =   */

  2   1             DECLARE DCL LITERALLY 'DECLARE';
  3   1             DCL AS LITERALLY 'LITERALLY';
  4   1             DCL PROC AS 'PROCEDURE';

  5   1             DCL MEMTOP BYTE AT (0005H),
                        FILE$NAME (*) BYTE DATA ('HPRES '),
                        ENTRY ADDRESS,
                        STATUS ADDRESS;

  6   1             LOAD:
                      PROC (FILE,BIAS,RETSW,ENTRY,STATUS) EXTERNAL;
  7   2                   DCL (FILE,BIAS,RETSW,ENTRY,STATUS) ADDRESS;
  8   2                 END LOAD;

  9   1             MEMCK:
                      PROC ADDRESS EXTERNAL;
 10   2                 END MEMCK;

 11   1             CALL MOVE (319,MEMCK+1,MEMCK-511);

 12   1             MEMTOP = HIGH (MEMCK - 193);

 13   1             CALL LOAD (.FILENAME,0,1,.ENTRY,.STATUS);

 14   1           END HP$9871A$LOADER;


MODULE INFORMATION:

     CODE AREA SIZE     = 004DH     77D
     VARIABLE AREA SIZE = 0004H      4D
     MAXIMUM STACK SIZE = 0008H      8D
     114 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE HP9871ARESIDENTDRIVER
OBJECT MODULE PLACED IN :F1:HPDRV.OBJ
COMPILER INVOKED BY:  PLM80 :F1:HPDRV.SRC


                $DATE(30JUL77)
                $TITLE('HPDRV - HP 9871A DRIVER')
                $PAGEWIDTH(72)
                $PAGELENGTH(57)

  1             HP$9871A$RESIDENT$DRIVER:
                  DO;

  2    1            DECLARE DCL LITERALLY 'DECLARE';
  3    1            DCL AS LITERALLY 'LITERALLY';
  4    1            DCL PROC AS 'PROCEDURE';

  5    1            DCL PRINTER$CONTROL AS '0DEH',
                        CONTROL$LINE$HIGH AS '0';

  6    1            EXIT:
                      PROC EXTERNAL;
  7    2                END EXIT;

  8    1            IOCHK:
                      PROC BYTE EXTERNAL;
  9    2                END IOCHK;

 10    1            IOSET:
                      PROC (STATUS) EXTERNAL;
 11    2                DCL STATUS BYTE;
 12    2                END IOSET;

 13    1            IODEF:
                      PROC (DEVICE,LOCATION) EXTERNAL;
 14    2                DCL DEVICE BYTE,
                            LOCATION ADDRESS;
 15    2                END IODEF;

 16    1            LIST$DRIVER:
                      PROC (CHARACTER) EXTERNAL;
 17    2                DECLARE CHARACTER BYTE;
 18    2                END LIST$DRIVER;

 19    1            OUTPUT (PRINTER$CONTROL) =
                      CONTROL$LINE$HIGH;

 20    1            CALL IODEF (6,.LIST$DRIVER);

 21    1            CALL IOSET (IOCHK AND 0011$1111B OR 1100$0000B);

 22    1            CALL EXIT;
```

```
  23  1         END HP$9871A$RESIDENT$DRIVER;


MODULE INFORMATION:

     CODE AREA SIZE      = 001FH     31D
     VARIABLE AREA SIZE  = 0000H      0D
     MAXIMUM STACK SIZE  = 0002H      2D
     49 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE HP9871ALISTDRIVER
OBJECT MODULE PLACED IN :F1:HPLIDR.OBJ
COMPILER INVOKED BY:  PLM80 :F1:HPLIDR.SRC


                $DATE(30JUL77)
                $TITLE('HPLIDR - HP 9871A LIST DRIVER')
                $PAGEWIDTH(72)
                $PAGELENGTH(57)

  1             HP$9871A$LIST$DRIVER:
                  DO;

  2   1             DECLARE DCL LITERALLY 'DECLARE';
  3   1             DCL AS LITERALLY 'LITERALLY';
  4   1             DCL PROC AS 'PROCEDURE';

  5   1             DCL PRINTER$STATUS AS '0DFH',
                        CHARACTER$OUTPUT$PORT AS '0DFH',
                        PRINTER$CONTROL AS '0DEH',
                        CONTROL$LINE$LOW AS '01H',
                        CONTROL$LINE$HIGH AS '0',
                        PRINTER$NOT$READY AS 'NOT ROR (INPUT (PRINTER$STAT
            -       US),1)',
                        FLAG$IS$LOW AS 'NOT ROL (INPUT (PRINTER$STATUS),3)
            -       ';


  6   1             LIST$DRIVER:
                      PROC (CHARACTER) PUBLIC;
  7   2                 DCL CHARACTER BYTE;

  8   2                 DO WHILE PRINTER$NOT$READY;
  9   3                 END;

 10   2                 OUTPUT(CHARACTER$OUTPUT$PORT) = CHARACTER;

 11   2                 OUTPUT(PRINTER$CONTROL) = CONTROL$LINE$LOW;

 12   2                 DO WHILE FLAG$IS$LOW;
 13   3                 END;

 14   2                 OUTPUT(PRINTER$CONTROL) = CONTROL$LINE$HIGH;

 15   2               END LIST$DRIVER;

 16   1           END HP9871A$LIST$DRIVER;


MODULE INFORMATION:

     CODE AREA SIZE      = 0028H      40D
```

```
VARIABLE AREA SIZE = 0001H      1D
MAXIMUM STACK SIZE = 0000H      0D
40 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

Ref. #AB120

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/4040 ☐ 8008 ☒ 8080 ☐ 8048 ☐ 8085 ☐ Other _____ (use additional sheets if necessary)

Program Title

Print text for SBC 80/10

Function

To print text and numerical data on a teletype with spacing inserted as desired.

Required Hardware

SBC 8080 (8010)

Required Software

SBC 80P Monitor

Input Parameters

Text (ASCII), Numeric data (FPU format) and output buffer address and length.

Output Results

The text will be output with spacing inserted as desired followed by optional output of numeric data. A check is made to insure that the buffer is not overflowed.

Registers Modified: ALL	Programmer: Doug Heere
RAM Required: 61 bytes	Company: PPG Industries
ROM Required: None	Address: Box 400
Maximum Subroutine Nesting Level: 3	City: Wichita Falls
Assembler/Compiler Used: PDP-11	State: Texas 76307

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                     1 ;
                     2 ; PROGRAM PRDAT
                     3 ;
                     4 ;     THIS PROGRAM IS DESIGNED TO PRINT DATA ON A
                     5 ; TELETYPE.  THE DATA IS A MIXTURE OF TEXT AND NUMERIC
                     6 ; DATA.  VARYING LENGTH MESSAGES ARE LOADED FROM PROM
                     7 ; OR RAM AREAS INTO AN OUTPUT BUFFER IN RAM.   THE
                     8 ; NUMERIC MUST PREVIOUSLY HAVE BEEN LOADED INTO A
                     9 ; BUFFER IMMEDIATELY FOLLOWING THE OUTPUT BUFFER.
                    10 ; THE NUMERIC DATA IS IN FLOATING POINT UNIT FORMAT.
                    11 ;     THE PRINT PROGRAM ALLOWS THE INSERTION OF SPACES
                    12 ; IN THE TEXT AND CONTROL OF PRINTING BOTH TEXT AND
                    13 ; NUMERIC DATA, OR TEXT ONLY.  AS THE TEXT IS LOADED
                    14 ; BY THE PROGRAM, IT IS CHECKED FOR SPACING CHARACTERS
                    15 ; (01H - 0CH) AND PRINT CONTROL CHARACTERS (0FEH, 0FFH)
                    16 ; WHEN A SPACING CHARACTER IS FOUND, THE EQUIVALENT
                    17 ; NUMBER OF SPACES IS INSERTED INTO THE OUTPUT BUFFER.
                    18 ; THE PRINT CONTROL CHARACTER IS THE LAST CHARACTER IN
                    19 ; THE TEXT.  A VALUE OF 0FEH CAUSES ONLY THE TEXT TO BE
                    20 ; PRINTED, WHILE A VALUE OF 0FFH CAUSES OUTPUT OF THE
                    21 ; NUMERIC DATA FOLLOWING THE TEXT.
                    22 ;     THE FOLLOWING EXAMPLE SHOWS THE BUFFER RELATIONSHI
                    23 ;AND TEXT STRING SETUP.  MESS1 WILL OUTPUT ONLY THE
                    24 ; TEXT '    TEXT1   ', WHILE MESS2 WILL CAUSE OUTPUT
                    25 ; OF THE TEXT 'TEXT2     ' FOLLOWED BY NUMERIC
                    26 ; DATA PREVIOUSLY LOADED IN THE ANSWER BUFFER.
                    27 ;
                    28 ;     OUTBF  SET  $           ; OUTPUT BUFFER
                    29 ;            DS   30          ; 30 BYTES
                    30 ;     ANSWR  SET  $           ; NUMERIC DATA
                    31 ;            DS   17          ; 17 BYTES
                    32 ;
                    33 ;     MESS1: DB   0DH         ; CARRIAGE RETURN
                    34 ;            DB   04H         ; 4 SPACES
                    35 ;            DB   'TEXT1'     ; TEXT
                    36 ;            DB   03H         ; 3 SPACES
                    37 ;            DB   0FEH        ; PRINT CONTROL, TEXT ONLY
                    38 ;     MESS2: DB   'TEXT2'     ; TEXT
                    39 ;            DB   05H         ; 5 SPACES
                    40 ;            DB   0FFH        ; PRINT CONTROL, TEXT AND
                    41 ;
                    42 ;     THE LENGTH OF THE BUFFER IS INPUT TO THE ROUTINE F
                    43 ; OVERFLOW CHECKING.  IF THE TEXT AS EXPANDED OVERFLOWS T
                    44 ; BUFFER, IT IS TRUNCATED.
                    45 ;
                    46 ; INPUT ARGUMENTS:
                    47 ;                 REG C    - LENGTH OF OUTPUT BUFFER (NO
                    48 ;                            INCL ANSWER BUFFER)
                    49 ;                 REG D&E  - ADDRESS OF OUTPUT BUFFER
                    50 ;                 REG H&L  - ADDRESS OF TEXT
                    51 ;
                    52 ; REGISTERS DESTROYED: ALL
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                    53 ;
                    54 ; MONITOR ROUTINES USED: ECHO
                    55 ;
4400                56 ORG    4400H
                    57 ;
01F9                58 ECHO   EQU  01F9H          ; MONITOR ROUTINE
                    59 ;
                    60 ; MAIN PROGRAM ENTRY POINT
                    61 ;
4400                62 PRDAT  SET  $
4400 D5             63        PUSH D              ; SAVE OUTPUT BUFFER ADDRESS
4401 CD1C44         64        CALL LDTXT          ; LOAD FIXED TEXT TO BUFFER
4404 E1             65        POP  H              ; GET OUTPUT BUFFER ADDRESS
                    66 ;
                    67 ; LAST CHARACTER IN FIXED TEXT (PRINT CONTROL ) IS RETUR
                    68 ; IN ACCUMULATOR BY LDTXT.  SAVE IT FOR LATER
                    69 ;
4405 47             70        MOV  B,A            ; PUT LAST CHAR IN REG B
4406 C5             71        PUSH B              ; SAVE B AND C
                    72 ;
                    73 ; REGISTER C CONTAINS NUMBER OF SLOTS IN BUFFER NOT FILE
                    74 ; BY FIXED TEXT.  THIS IS USED TO LOCATE ANSWER BUFFER
                    75 ;
                    76 ; OUTPUT ROUTINE SEARCHES FOR ENDING CHARACTER.  IF PRINT
                    77 ; THIS IS A ZERO.  IF PRINTING FPU NUMERIC DATA IT IS A S
                    78 ;
4407 1600           79        MVI  D,00H          ; LOAD TEXT END CHARACTER FOR C
4409 CD5644         80        CALL OUTPT          ; PRINT TEXT
440C C1             81        POP  B              ; RECALL PRINT CONTROL AND EMPT
440D 78             82        MOV  A,B            ; GET PRINT CONTROL IN ACCUM
                    83 ;
                    84 ; IF PRINT CONTROL IS FE, EXIT- IF FF PRINT ANSWER
                    85 ;
440E FEFF           86        CPI  0FFH           ; IS CONTROL FF
4410 C0             87        RNZ                 ; NO, FINISHED
                    88 ;
                    89 ; LOCATE ANSWER BUFFER BY SKIPPING UNFILLED OUTPUT BUFFE
                    90 ; COUNT IS IN REGISTER C
                    91 ;
4411 23             92 ENDBF: INX  H              ; INCREMENT POINTER
4412 0D             93        DCR  C              ; REDUCE COUNT
4413 F21144         94        JP   ENDBF          ; LOOP UNTIL DONE
                    95 ;
                    96 ; SET END CHARACTER TO SPACE FOR PRINTING FPU DATA
                    97 ;
4416 1620           98        MVI  D,20H          ; SET SPACE AS COMPARE CHAR
4418 CD5644         99        CALL OUTPT          ; PRINT NUMERIC DATA
441B C9            100        RET                 ; EXIT
                   101 ;
                   102 ;
                   103 ;*******************************************************
                   104 ;
                   105 ; SUBROUTINE TO LOAD TEXTS FROM FIXED ARE TO OUTPUT BUFF
                   106 ; ALSO XPANDS SPACING CODE
                   107 ;
```

```
 LOC  OBJ          SEQ          SOURCE STATEMENT

                    108 ;  INPUT -
                    109 ;          C - BUFFER SIZE
                    110 ;        D&E - OUTPUT BUFFER ADDRESS
                    111 ;        H&L - TEXT ADDRESS
                    112 ;
                    113 ;  OUTPUT-
                    114 ;          C - UNSED PORTION OF OUTPUT BUFFER
                    115 ;
 441C               116 LDTXT    SET  $              ; ENTRY POINT
 441C 0D            117          DCR  C              ; ADJUST BUFFER COUNTER
 441D 7E            118 GETCH:   MOV  A,M            ; GET TEXT CHARACTER
                    119 ;
                    120 ;  TEST FOR SPACING CHARACTER <0DH
                    121 ;
 441E FE0D          122          CPI  0DH            ; IS IT SPACING CHARACTER?
 4420 D23B44        123          JNC  LODCH          ; NO, LOAD IT
                    124 ;
                    125 ;  SPACING CHARACTER FOUND,  INSERT SPACES IN OUTPUT BUFFE
                    126 ;
 4423 47            127          MOV  B,A            ; SET UP COUNTER
 4424 05            128          DCR  B              ; AND ADJUST IT
 4425 EB            129          XCHG                ; POINT H&L TO OUTPUT BUFFER
 4426 3620          130 SPACE:   MVI  M,20H          ; INSERT SPACE
 4428 23            131          INX  H              ; ADJUST BUFFER POINTER
                    132 ;
                    133 ;  CHECK FOR BUFFER OVERFLOW
                    134 ;
 4429 0D            135          DCR  C              ; DECREMENT BUFFER SPACE AVAIL
 442A CA4544        136          JZ   TRUNC          ; IF FULL, TRUNCATE
                    137 ;
                    138 ;  LOOP UNTIL REQUIRED NUMBER OF SPACES INSERTED
                    139 ;
 442D 05            140          DCR  B              ; DECREMENT SPACE COUNTER
 442E F22644        141          JP   SPACE          ; LOOP FOR DESIRED NUMBER
                    142 ;
                    143 ;  ADJUST BUFFER COUNTER
                    144 ;
 4431 0C            145          INR  C              ; ADJUST ROOM IN BUFFER
                    146 ;
                    147 ;  CHECK FOR END OF BUFFER AND LOAD NEXT CHARACTER
                    148 ;
 4432 0D            149 NXTCH:   DCR  C              ; DECREMENT ROOM IN BUFFER
 4433 CA4544        150          JZ   TRUNC          ; IF FULL TRUNCATE
 4436 EB            151          XCHG                ; POINT H&L TO TEXT
 4437 23            152          INX  H              ; NEXT CHARACTER
 4438 C31D44        153          JMP  GETCH          ; LOAD NEXT CHARACTER
                    154 ;
                    155 ;  NON SPACIN CHARACTER FOUND,  CHECK FOR PRINT CONTROL
                    156 ;
 443B FEFD          157 LODCH:   CPI  0FDH           ; IS IT PRINT CONTROL?
 443D DA5044        158          JC   NOTPC          ; NO, GO LOAD IT
                    159 ;
                    160 ;  PRINT CONTROL CHARACTER FOUND,  END LOADING
                    161 ;
                    162 ;
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                   163 ; ENTRY FOR END OF TEXT OR TRUNCATION
                   164 ; PUT ZERO IN LAST POSITION, RETURN REMAINING BUFFER ROO
                   165 ; REG C
                   166 ;
4440 EB            167 ENDCH:  XCHG                ; POINT H&L TO OUTPUT BUFFER
4441 3600          168         MVI  M,00H          ; PUT ZERO IN LAST POSITION
4443 EB            169         XCHG                ; POINT H&L TO TEXT
4444 C9            170         RET                 ; EXIT
                   171 ;
                   172 ; MESSAGE TRUNCATED, FIND PRINT CONTROL CHARACTER
                   173 ;
4445 EB            174 TRUNC:  XCHG                ; POINT TO TEXT
4446 7E            175 LOOPA:  MOV  A,M            ; GET TEXT CHARACTER
4447 FEFD          176         CPI  0FDH           ; IS IT PRINT CNTRL
4449 D24044        177       JNC  ENDCH'           ; YES, FINISH
444C 23            178         INX  H              ; NO, POINT TO NEXT CHARACTER
444D C34644        179         JMP  LOOPA          ; AND CHECK IT
                   180 ;
                   181 ; TEXT CHARACTER, LOAD TO BUFFER
                   182 ;
4450 EB            183 NOTPC:  XCHG                ; POINT TO OUTPUT BUFFER
4451 77            184         MOV  M,A            ; LOAD CHARACTER TO BUFFER
4452 23            185         INX  H              ; INCREMENT BUFFER POINTER
4453 C33244        186         JMP  NXTCH          ; GET NEXT CHARACTER
                   187 ;
                   188 ;
                   189 ; ************************************************************
                   190 ;
                   191 ; SUBROUTINE TO PRINT TEXT OR NUMERIC DATA
                   192 ;
                   193 ; INPUT-
                   194 ;        D- CHARACTER TO COMPARE TO TERMINATE PRINT
                   195 ;      H&L- OUTPUT BUFFER
                   196 ;
4456               197 OUTPT   SET  $
4456 7E            198         MOV  A,M            ; GET CHARACTER
4457 BA            199         CMP  D              ; IS IT END CHARACTER?
4458 C8            200         RZ                  ; YES, FINISHED
                   201 ;
                   202 ; OUTPUT CHARACTER TO TELETYPE USING MONITOR ROUTINE
                   203 ;
4459 4F            204         MOV  C,A            ; PREPARE FOR OUTPUT
445A CDF901        205         CALL ECHO           ; OUTPUT CHARACTER
445D 23            206         INX  H              ; POINT TO NEXT CHARACTER
445E C35644        207         JMP  OUTPT          ; GET NEXT CHARACTER
                   208 ;
                   209 ;
                   210 ; ************************************************************
                   211 ;
0ADF               212 CNASI   EQU  0ADFH
15E5               213 BINBC   EQU  15E5H
                   214 ;
                   215 ; TEST NORMAL MESSAGE WITH ANSWER
                   216 ;
4461               217 START   SET  $
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

4461 3E1E          218          MVI   A,30              ; SET BUFFER SIZE
4463 11D144        219          LXI   D,OUTB1           ; POINT TO BUFFER
4466 213845        220          LXI   H,HEAD1           ; HEADER MESSAGE
4469 CD0044        221          CALL  PRDAT             ; PRINT IT
446C 3E1E          222          MVI   A,30              ; RESET BUFFER SIZE
446E 11D144        223          LXI   D,OUTB1           ; POINT TO BUFFER
4471 215045        224          LXI   H,STEP1           ; STEP MESSAGE
4474 CD0044        225          CALL  PRDAT             ; PRINT IT
4477 3E1E          226          MVI   A,30              ; SET BUFFER SIZE AT 30 BYTE
4479 CD9E45        227          CALL  DOIT              ; LOAD COUNT IN ANSWER BUFFE
447C 11D144        228          LXI   D,OUTB1           ; POINT TO BUFFER
447F 210045        229          LXI   H,MESS1           ; POINT TO MESSAGE
4482 CD0044        230          CALL  PRDAT             ; PRINT MESSAGE AND ANSWER
                   231 ;
                   232 ; TEST TRUNCATION OF MESSAGE
                   233 ;
4485 3E1E          234          MVI   A,30              ; SET BUFFER SIZE
4487 11D144        235          LXI   D,OUTB1           ; POINT TO BUFFER
448A 216345        236          LXI   H,STEP2           ; POINT TO MESSAGE
448D CD0044        237          CALL  PRDAT             ; PRINT IT
4490 3E0A          238          MVI   A,10              ; SET BUFFER LENGTH AT 10 BY
4492 CD9E45        239          CALL  DOIT              ; PUT COUNT IN ANSWER
4495 11E544        240          LXI   D,OUTB2           ; POINT TO SHORT BUFFER
4498 210045        241          LXI   H,MESS1           ; POINT TO MESSAGE
449B CD0044        242          CALL  PRDAT             ; PRINT MESSAGE
                   243 ;
                   244 ; TEST MESSAGE WITHOUT ANSWER
                   245 ;
449E 3E1E          246          MVI   A,30              ; SET BUFFER SIZE
44A0 11D144        247          LXI   D,OUTB1           ; POINT TO BUFFER
44A3 217445        248          LXI   H,STEP3           ; POINT TO MESSAGE
44A6 CD0044        249          CALL  PRDAT             ; PRINT IT
44A9 3E1E          250          MVI   A,30              ; SET BUFFER AT 30 BYTES
44AB CD9E45        251          CALL  DOIT              ; LOAD COUNT IN ANSWER
44AE 11D144        252          LXI   D,OUTB1           ; POINT TO LONG BUFFER
44B1 211345        253          LXI   H,MESS2           ; POINT TO MESSAGE
44B4 CD0044        254          CALL  PRDAT             ; PRINT IT
                   255 ;
                   256 ; TEST TRUNCATING EXPANSION
                   257 ;
44B7 3E1E          258          MVI   A,30              ; SET BUFFER SIZE
44B9 11D144        259          LXI   D,OUTB1           ; POINT TO BUFFER
44BC 218A45        260          LXI   H,STEP4           ; POINT TO MESSAGE
44BF CD0044        261          CALL  PRDAT             ; PRINT IT
44C2 3E1E          262          MVI   A,30              ; SET BUFFER TO 30 BYTES
44C4 CD9E45        263          CALL  DOIT              ; LOAD COUNT IN ANSWER
44C7 11D144        264          LXI   D,OUTB1           ; POINT TO LONG BUFFER
44CA 212445        265          LXI   H,MESS3           ; POINT TO MESSAGE
44CD CD0044        266          CALL  PRDAT             ; PRINT IT
                   267 ;
                   268 ; END OF TEST
                   269 ;
44D0 CF            270          RST   1
                   271 ;
                   272 ; BUFFER AREAS
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                   273 ;
44D1               274 OUTB1  SET  $
0014               275        DS   20
44E5               276 OUTB2  SET  $
000A               277        DS   10
44EF               278 ANSWR  SET  $
0011               279        DS   17
                   280 ;
                   281 ;  TEXTS
                   282 ;
4500 0D            283 MESS1: DB   0DH       ; CARRIAGE RETURN
4501 02            284        DB   02H       ; 2 SPACES
4502 42554646      285        DB   'BUFFER LENGTH ='
4506 4552204C
450A 454E4754
450E 48203D
4511 02            286        DB   02H       ; 2 SPACES
4512 FF            287        DB   0FFH      ; PRINT TEXT AND ANSWER
4513 0D            288 MESS2: DB   0DH       ; CARRIAGE RETURN
4514 04            289        DB   04H       ; 4 SPACES
4515 57495448      290        DB   'WITHOUT ANSWER'
4519 4F555420
451D 414E5357
4521 4552
4523 FE            291        DB   0FEH      ; TEXT ONLY
4524 0D            292 MESS3: DB   0DH       ; CARRIAGE RETURN
4525 02            293        DB   02H       ; 2 SPACES
4526 5452554E      294        DB   'TRUNCATE SPACES'
452A 43415445
452E 20535041
4532 434553
4535 0A            295        DB   0AH       ; 10 SPACES
4536 0C            296        DB   0CH       ; 12 SPACES
4537 FF            297        DB   0FFH      ; TEXT AND ANSWER
4538 0D            298 HEAD1: DB   0DH       ; CARRIAGE RETURN
4539 06            299        DB   06H       ; 6 SPACES
453A 54455354      300        DB   'TEST OF PRDAT ROUTINE'
453E 204F4620
4542 50524441
4546 5420524F
454A 5554494E
454E 45
454F FE            301        DB   0FEH      ; TEXT ONLY
4550 0D            302 STEP1: DB   0DH       ; CARRIAGE RETURN
4551 20544558      303        DB   ' TEXT PLUS ANSWER'
4555 5420504C
4559 55532041
455D 4E535745
4561 52
4562 FE            304        DB   0FEH      ; TEXT ONLY
4563 0D            305 STEP2: DB   0DH       ; CARRIAGE RETURN
4564 20545255      306        DB   ' TRUNCATED TEXT'
4568 4E434154
456C 45442054
4570 455854
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

4573 FE            307          DB  0FEH      ; TEXT ONLY
4574 0D            308 STEP3:   DB  0DH       ; CARRIAGE RETURN
4575 20544558      309          DB  ' TEXT WITHOUT ANSWER'
4579 54205749
457D 54484F55
4581 5420414E
4585 53574552
4589 FE            310          DB  0FEH      ; TEXT ONLY
458A 0D            311 STEP4:   DB  0DH       ; CARRIAGE RETURN
458B 20545255      312          DB  ' TRUNCATED SPACING'
458F 4E434154
4593 45442053
4597 50414349
459B 4E47
459D FE            313          DB  0FEH      ; TEXT ONLY
                   314 ;
                   315 ; ROUTINE TO LOAD COUNT INTO ANSWER BUFFER
                   316 ;
459E F5            317 DOIT:    PUSH PSW       ; SAVE ACCUM
459F 5F            318          MOV  E,A       ; BINARY COUNT
45A0 1600          319          MVI  D,00H     ; SET FOR CONVERSION
45A2 CDE515        320          CALL BINBC     ; CONVERT TO BCD
45A5 7D            321          MOV  A,L       ; GET BCD COUNT
45A6 21EF44        322          LXI  H,ANSWR   ; POINT TO ANSWER BUFFER
45A9 CDDF0A        323          CALL CNASI     ; CONVERT COUNT TO ASCII IN B&C
45AC 70            324          MOV  M,B       ; STORE FIRST DIGIT
45AD 23            325          INX  H         ; MOVE POINTER
45AE 71            326          MOV  M,C       ; STORE SECOND DIGIT
45AF 23            327          INX  H         ; MOVE POINTER
45B0 3620          328          MVI  M,20H     ; SET SPACE FOR END CHARACTER
45B2 F1            329          POP  PSW       ; RETRIEVE COUNT
45B3 4F            330          MOV  C,A       ; PUT COUNT INTO REG C
45B4 C9            331          RET
                   332 ;
                   333 ; END OF PROGRAM
                   334 ;
                   335          END
```

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USER SYMBOLS

ANSWR	A 44EF	BINBC	A 15E5	CNASI	A 0ADF	DOIT	A 459E	ECHO	A 01F
GETCH	A 441D	HEAD1	A 4538	LDTXT	A 441C	LODCH	A 443B	LOOPA	A 444
MESS3	A 4524	NOTPC	A 4450	NXTCH	A 4432	OUTB1	A 44D1	OUTB2	A 44E
SPACE	A 4426	START	A 4461	STEP1	A 4550	STEP2	A 4563	STEP3	A 457

ASSEMBLY COMPLETE, NO ERRORS

Ref.#AB121

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 ☐8080 ☐8048 ☒8085 ☐Other_____ (use additional sheets if necessary)

Program Title

MON805 -- Monitor for iSBC 80/05 or 80/04

Function

2K byte debug monitor for Intel 80/05 and 80/04. Provides simple memory and register display and modification commands as well as program execution with breakpoints. In addition, the monitor supports paper tape I/O if a TTY unit is connected as the console device.

Required Hardware

Intel iSBC 80/05 or 80/04 with console CRT or TTY (TTY required for paper tape I/O).

Required Software

None

Input Parameters

Monitor includes rudimentary command line interpreter for operator interface. See SBC 80P05 User's Guide (Manual #9800508A) for further details.

Output Results

--

Available on non-system diskette only for $35.00. (source & object code included)

Registers Modified: All	Programmer: This monitor is supplied in ROM form to customers purchasing
RAM Required: 31 bytes & stack	Company: the iSBC 80P05.
ROM Required: 1714 bytes	Address:
Maximum Subroutine Nesting Level: ∿8	City:
Assembler/Compiler Used: 8080 macroassembler	State:

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 ☐8080 ☐8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

MON810 -- Monitor for iSBC 80/10 or 80/10A

Function

2K byte debug monitor for Intel iSBC 80/10 and 80/10A. Provides simple memory and register display and modification commands as well as program execution with breakpoints. In addition, the monitor supports paper tape I/O if a TTY unit is connected as the console device.

Required Hardware

Intel iSBC 80/10 or 80/10A with console CRT or TTY (TTY required for paper tape I/O).

Required Software

none

Input Parameters

Monitor includes rudimentary command line interpreter for operator interface. See SBC 80P and 80P10 Prototyping Package User's Guide (manual order #98002230) for further details.

Output Results

--

Available on non-system diskette only for $35.00. (source & object code included)

Registers Modified: A11	Programmer: This monitor is supplied in ROM form to customers purchasing
RAM Required: 16 + stack usage	Company: the iSBC 80P10 Prototype Package and System 80/10.
ROM Required: 1374 bytes	Address:
Maximum Subroutine Nesting Level: ~8	City:
Assembler/Compiler Used: 8080 macroassembler	State:

Ref. #AB123

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 ☐8080 ☐8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

MON820 -- Monitor for iSBC 80/20 or 80/20-4

Function

2K byte debug monitor for Intel iSBC 80/20 and 80/20-4. Provides simple memory and register display and modification commands as well as program execution with breakpoints and single stepping. In addition, the monitor supports paper tape I/O if a TTY unit is connected as the console device.

Required Hardware

Intel iSBC 80/20 or 80/20-4 with console CRT or TTY (TTY required for paper tape I/O)

Required Software

none

Input Parameters

Monitor includes rudimentary command line interpreter for operator interface. See SBC 80P20 User's Guide (Manual #98-338C) for further details.

Output Results

--

Available on non-system diskette only for $35.00. (source & object code included)

Registers Modified: All	Programmer: This monitor is provided in ROM form to customers purchasing
RAM Required: 45 + stack	Company: the iSBC 80P20 prototype package and system 80/20-4.
ROM Required: 1708	Address:
Maximum Subroutine Nesting Level: ~ 8	City:
Assembler/Compiler Used: 8080 macroassembler	State:

Ref.# AA20

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/4040 ☐ 8008 ☒ 8080 ☐ 8048 ☐ 8085 ☐ Other _____ (use additional sheets if necessary)

Program Title

INPUT/OUTPUT DIAGNOSTIC 8080

Function

Allows interactive testing of any MDS I/O port(s). Also allows saving and reloading the test program.

Required Hardware

MDS System with Disc

Required Software

MDS ISIS System

Input Parameters

See Command Summary

Output Results

Program available on diskette only.

Registers Modified: ALL	Programmer: S. G. Thompson
RAM Required: 2340 Decimal	Company: Harris Controls
ROM Required:	Address: P.O. Box 430
Maximum Subroutine Nesting Level: 16	City: Melbourne
Assembler/Compiler Used: Intel V1.1	State: Florida 32901

COMMAND SUMMARY

Input/Output Diagnostic 8080

The following commands are valid when the diagnostic is loaded and has typed the sign-on message: 'I/O test V1.1' and prompted the operator with 'M:'. Abbreviations for each command are in parentheses.

CLR (CL) - Clears the Input/Output buffer of all commands.

EDIT N (ED) - Edits the Input/Output buffer starting at line number 'N'. Valid inputs are:

IN YY (I)	input data from port 'YY'
OUT XX, YY (O)	output data 'XX' to port 'YY'
DELT (D)	delete line entry
X	exit from editor
CR	(carriage return) steps to next line, the present line is not altered

EXIT (EX) - Exit to the MDS-800 monitor

INIT - Initialize the one bit flag variables and 16 bit variables as follows:

FS = 1	format switch on
RS = 0	repeat switch off
HS = 0	home cursor switch off
IC = 32_{16}	inter character time of 50_{10} ms
IM = 0	inter message time of 0_{10} ms

IN YY - Inputs and displays the data received from port 'YY'.

LIST XX (LI) - List the Input/Output buffer starting with line 'XX'. If the line # is missing then the listing starts at line number 0.

LOAD NAME (LO) - Load the previously saved diagnostic Input/Output buffer saved under the disk name 'NAME'. 'NAME' may have the following formats:

:F1:STEVE.TST	as	:drive#:Filename.extension
STEVE	as	Filename (assumes drive 0)
DIAG.OOX	as	Filename.extension
DIAG	as	Filename (assumes drive 0)

OUT XX, YY - The data specified by 'XX' is output to port 'YY'.

SAVE NAME - Save the present diagnostic input/output buffer to disk under the name 'NAME'. 'NAME' may have the formats previously listed for the load command.

SV NAME ZZZZ - Set the Variables named 'NAME' to the value 'ZZZZ'.

SEND XX, YY (SE) - Send the I/O commands in the I/O buffer starting at line number XX through line number YY.

TA - Type out All the values of the variables.

The following variables are used in conjunction with the 'SEND' command:

FS - Format Switch if 'one' causes the data received from any IN commands to be printed on the operator interaction device.

HS - Home Switch if 'one' causes the control codes for clear screen and home to be executed by the output formatter so that one CRT page of data will be repetitively displayed.

RS - Repeat Switch if 'one' causes the I/O buffer to be sent repeatedly until a break character (STX = control B) is entered by the operator.

IC - Inter Character timing value. The time value is used between each I/O command. The units are 1/1000 of a second.

IM - Inter Message timing value. The time value used between each repetition of the I/O buffer (assumes RS=1). The units are 1/1000 of a second.

Ref.# AB124

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 ☒8080 ☐8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

EXTRCT

Function

Extracts selected lines from an ISIS source or print file

Required Hardware

MDS, Floppy Disk

Required Software

ISIS

Input Parameters

1) Command Line:
   EXTRCT source file TO destination file

2) Range of Lines:
   XXXX,YYYY

3) End of Operation:
   END

Output Results

Registers Modified: N/A	Programmer: Ken Norris
RAM Required: 37AH	Company: Daniel Systems
ROM Required:	Address: 9525 Katy Frwy.
Maximum Subroutine Nesting Level: N/A	City: Houston
Assembler/Compiler Used: 8080/8085 Macro Assembler V2.0	State: Texas 77024

```
ASM80 :F1:EXTRCT.SRC MACROFILE NOGEN


ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0        EXTRCT   PAGE    1
EXTRCT.SRC - SELECTIVE LINE EXTRACT

 LOC  OBJ        SEQ        SOURCE STATEMENT

                   1 $TITLE('EXTRCT.SRC - SELECTIVE LINE EXTRACT')
                   2         NAME    EXTRCT
                   3         STKLN   40H
                   4 ;
                   5 ;THIS PROGRAM EXTRACTS LINES FROM A SOURCE FILE
                   6 ;
                   7 ;COMMAND SEQUENCE:
                   8 ;EXTRCT SOURCE.FILE TO DESTINATION.FILE
                   9 ;
                  10 ;PROGRAM WILL PROMPT CONSOLE WITH '*'.
                  11 ;RESPOND WITH: LINEA,LINEB(CR).
                  12 ;LINES FROM LINEA TO LINEB WILL BE MOVED TO DESTINATION FILE
                  13 ;REPEAT UNTIL ALL DESIRED SEGMENTS HAVE BEEN MOVED, THEN
                  14 ;RESPOND WITH: END(CR).
                  15 ;(DO NOT ATTEMPT TO BACK UP, RESULTS ARE UNDEFINED.)
                  16 ;PROGRAM WILL RETURN TO ISIS.
                  17
0000              18 OPEN    EQU     0
0003              19 READ    EQU     3
0004              20 WRITE   EQU     4
0009              21 EXIT    EQU     9
000C              22 ERROR   EQU     12
                  23
                  24         EXTRN   ISIS
                  25
                  26 ;MACRO DEFS
                  27 SYSTEM  MACRO   @TYPE,@BLOCK
                  28         MVI     C,@TYPE
                  29         LXI     D,@BLOCK
                  30         CALL    ISIS
                  31         LDA     STATUS
                  32         ORA     A
                  33         JNZ     ERR
                  34         ENDM
                  35
                  36 MOVE    MACRO   @TO
                  37         LXI     D,@TO
                  38         CALL    MOVEW
                  39         ENDM
                  40
                  41 TEST    MACRO   @W1,@W2
                  42         LXI     H,@W1
                  43         LXI     D,@W2
                  44         CALL    TESTW
                  45         ENDM
                  46
                  47 CHECK   MACRO   @DEST
                  48         LXI     H,@DEST
                  49         CALL    CONVD
                  50         ENDM
                  51
                  52 COMPR   MACRO   @1,@2
```

```
 LOC  OBJ          SEQ          SOURCE STATEMENT

                    53          LHLD    @2
                    54          XCHG
                    55          LHLD    @1
                    56          CALL    HILO
                    57          ENDM
                    58
                    59          CSEG    PAGE
                    60
 0000 310000   S    61 BEGIN:   LXI     SP,STACK        ;INITIALIZE PTRS
 0003 210000        62          LXI     H,0
 0006 220200   D    63          SHLD    IPTR
                    64          SYSTEM  READ,CBLK       ;GET COMMAND STRING
 0018 216800   D    71          LXI     H,CBUFF
                    72          MOVE    IFILE           ;GET INPUT FILE NAME
                    75          MOVE    DUMMY           ;GET 'TO' (WE HOPE)
                    78          MOVE    OFILE           ;GET OUTPUT FILE NAME
                    81          TEST    DUMMY,@TO@      ;MIDDLE WORD 'TO'?
 0036 C2E000   C    85          JNZ     SERR            ;NO
                    86          SYSTEM  OPEN,BBLK       ;OK, OPEN :BB:
                    93          SYSTEM  OPEN,OIBLK      ;OPEN INPUT (ECHO TO :BB:)
                   100          SYSTEM  OPEN,OOBLK      ;OPEN OUTPUT
                   107
                   108 LOOP:    SYSTEM  WRITE,SBLK      ;PROMPT
                   115          SYSTEM  READ,CBLK       ;GET RANGE
 0084 3A0000   D   122          LDA     ACTUAL          ;EOF?
 0087 B7           123          ORA     A
 0088 CAD100   C   124          JZ      DONE
                   125          TEST    CBUFF,@END@     ;'END'?
 0094 DAD100   C   129          JC      DONE            ;YES
 0097 116800   D   130          LXI     D,CBUFF         ;NO,TEST & CONVERT TO DECIMAL
                   131          CHECK   PARM1           ;START OF RANGE
 00A0 D2E000   C   134          JNC     SERR            ;BAD
                   135          CHECK   PARM2           ;END OF RANGE
 00A9 D2E000   C   138          JNC     SERR
                   139          COMPR   PARM2,PARM1     ;TEST PARM1<=PARM2
 00B6 FAE000   C   144          JM      SERR            ;NO
 00B9 CD6801   C   145          CALL    SKIP            ;YES, SKIP TO START OF RANGE ON INPUT FILE
 00BC CD8C01   C   146          CALL    SAVE            ;COPY FROM HERE TO END OF RANGE
 00BF C36600   C   147          JMP     LOOP            ;CONTINUE
                   148 ERR:     SYSTEM  ERROR,EBLK      ;SYSTEM ERROR
                   155 DONE:    SYSTEM  EXIT,XBLK       ;RETURN TO SYSTEM
 00E0 3ECA         162 SERR:    MVI     A,202           ;SYNTAX ERROR
 00E2 320102   C   163          STA     STATUS
 00E5 C3C200   C   164          JMP     ERR
                   165
 00E8 3E0D         166 MOVEW:   MVI     A,0DH           ;MOVES WORD POINTED TO BY H&L TO BUFFER
 00EA 12           167          STAX    D               ;POINTED TO BY D&E
 00EB 7E           168 MOV05:   MOV     A,M             ;TERMINATES ON TRAILING COMMA, SPACE
 00EC 23           169          INX     H               ;SLASH OR CR.  IGNORES LEADING COMMA,
 00ED FE0D         170          CPI     0DH             ;SPACE OR SLASH.  TERMINATOR RETURNED
 00EF C2F600   C   171          JNZ     MOV10           ;IN A-REG.
 00F2 3EFF         172          MVI     A,0FFH
 00F4 B7           173          ORA     A               ;A-REG MINUS ON RET=> NO PARM.
 00F5 C9           174          RET
 00F6 CD0901   C   175 MOV10:   CALL    VALDL           ;LEADING TERM.?
```

```
 LOC  OBJ          SEQ          SOURCE STATEMENT

 0145 09            231          DAD     B               ;*10
 0147 4F            232          MOV     C,A
 0148 0600          233          MVI     B,0
 014A 09            234          DAD     B               ;+ NEW DIGIT
 014B C33201    C   235          JMP     CON05
 014E C1            236 CON10.   POP     B
 014F 7D            237          MOV     A,L
 0150 02            238          STAX    B
 0151 03            239          INX     B
 0152 7C            240          MOV     A,H
 0153 02            241          STAX    B
 0154 C32501    C   242          JMP     GOOD            ;SUCCESS
 0157 E1            243 CON15:   POP     H               ;ERROR
 0158 C32701    C   244          JMP     BAD
                    245
 015B D630          246 VALDG:   SUI     '0'             ;VALID DIGIT TEST(CONVERT TO BIN)
 015D FA2701    C   247          JM      BAD
 0160 FE0A          248          CPI     10
 0162 F22701    C   249          JP      BAD             ;ERROR
 0165 C32501    C   250          JMP     GOOD            ;OK NOW
                    251
                    252 SKIP:    COMPR   IPTR,PARM1      ;READ TO START OF RANGE
 0172 D8            257          RC                      ;OK
                    258          SYSTEM  READ,IBLK       ;NOT YET
 0182 2A0200    D   265          LHLD    IPTR
 0185 23            266          INX     H
 0186 220200    D   267          SHLD    IPTR
 0189 C36801    C   268          JMP     SKIP
                    269
                    270 SAVE:    SYSTEM  WRITE,OBLK      ;SAVE THIS RECORD
                    277          COMPR   IPTR,PARM2      ;SAVED TO END OF RANGE YET?
 01A5 D8            282          RC                      ;YES
                    283          SYSTEM  READ,IBLK       ;NO, GET NEXT
 01B5 2A0200    D   290          LHLD    IPTR
 01B8 23            291          INX     H
 01B9 220200    D   292          SHLD    IPTR
 01BC C38C01    C   293          JMP     SAVE
                    294
                    295 ;CONTROL BLOCKS
 01BF 0100          296 CBLK:    DW      1,CBUFF,128,ACTUAL,STATUS       ;CONSOLE INPUT BLOCK
 01C1 6800      D
 01C3 8000
 01C5 0000      D
 01C7 0102      C
 01C9 F101      C   297 OOBLK:   DW      OAFT,OFILE,2,0,STATUS           ;OPEN OUTPUT BLOCK
 01CB 4800      D
 01CD 0200
 01CF 0000
 01D1 0102      C
 01D3 E701      C   298 OIBLK:   DW      IAFT,IFILE,1                    ;OPEN INPUT BLOCK
 01D5 0800      D
 01D7 0100
 01D9 0000          299 BAFT:    DW      0,STATUS
 01DB 0102      C
 01DD D901      C   300 BBLK:    DW      BAFT,@BB0,2,0,STATUS            ;OPEN :BB: BLOCK
```

EXTRCT.SRC - SELECTIVE LINE EXTRACT

```
MOV15  C 00FE    MOVE   + 0001    MOVEW  C 00E8    OAFT   C 01F1    OBLK   C 01F1    OFILE  D 0048    OIBLK  C 01D3
OOBLK  C 01C9    OPEN   A 0000    PARM1  D 0004    PARM2  D 0006    READ   A 0003    SAVE   C 018C    SBLK   C 01F9
SERR   C 00E0    SKIP   C 0168    STATUS C 0201    SYSTEM + 0000    TEST   + 0002    TESTW  C 0115    VALDG  C 015B
VALDL  C 0109    WRITE  A 0004    XBLK   C 0203
```

ASSEMBLY COMPLETE, NO ERRORS

Ref. #AB125

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 4040 ☐ 8008 ☒ 8080 ☐ 3000 ☐ Other_____ (use additional sheets if necessary)

Program Title: BINARY PUNCH TAPE FOR PROM PROGRAMMER - PUNCH

Function:

Required Hardware: MDS-800, INTELLEC DISKETTE OPERATING SYSTEM, TTY
OPTIONAL: HIGH SPEED PUNCH

Required Software: ISIS-II

Input Parameters: SEE ATTACHMENT

Output Results: SEE ATTACHMENT

Registers Modified: ALL	Programmer: JOHN P. NAGEL
RAM Required: 32K	Company: SINGER-KEARFOTT
ROM Required: MDS MONITOR	Address: 150 TOTOWA ROAD
Maximum Subroutine Nesting Level: 3 LEVELS	City: WAYNE
Assembler/Compiler Used: ASM-80	State: NEW JERSEY, 07470

BINARY PUNCH PROGRAM

Example of Use:

An 8K program starting at location 0000H has been written and debugged and resides on the non-system disk, and is named SAMPLE.OBJ. A User wants to punch 8, 1K paper tapes to be used for programming 8, 1K PROMS. The Binary Punch Program is named PUNCH and resides on the system disk.

The paper tape is punched as follows:

1) In ISIS-II mode, the User types the following command:

   - PUNCH :F1: SAMPLE. OBJ (CR)

2) The Program will respond with the following question:

   Data Length in HEX 100, 200, 400, 800?

   The User should type in 400 (CR)

3) The Program will respond with the following question:

   Start Address in HEX?

   The User should type in Ø (CR)

4) The Program will respond with the following question:

   Number of tapes?

   The User should type in 8 (CR)

5) The Paper Tape Punch will punch 8 tapes with a leader and a trailer and with 256 nulls inbetween each tape. Then the ISIS-II prompt will appear on the console.

Ref. #AB126

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 ☒8080 ☐8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

HEXSYM

Function

Converts absolute object code in Hex-file-format to an object module in OBJ-format with symbol table.

Required Hardware

MDS-800 (with DOS)

Required Software

ISIS-II, Console commands HEXOBJ, COPY, DELETE

Input Parameters

The program HEXSYM must be used with the following sequence of commands:

HEXOBJ $filename_1$ TO HHH START(addr)
HEXSYM $filename_1$
COPY H1,H2,H3 TO $filename_2$
DELETE H?,HHH

Output Results

$filename_1$ - file with object code in hex-file format

$filename_2$ - converted file in absolute OBJ-file-format with a symbol table

H1,H2,H3 and HHH are auxiliary files

Registers Modified: ALL	Programmer: Macha Erich (E52 E32)
RAM Required: ISIS-dependent	Company: AEG Telefunken
ROM Required: --	Address: Steinheimer Str. 117
Maximum Subroutine Nesting Level: --	City: D-6453 Seligenstadt
Assembler/Compiler Used: Intellec MDS Macro Assembler V. 2.0	State: WESTERN GERMANY

```
ASM80 :F1:RETRVE.SRC

ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0        MODULE   PAGE    1

 LOC  OBJ         SEQ         SOURCE STATEMENT

                    1 ;TITLE     'RETRIEVE'
                    2 ;
                    3 ;A.D. SHORT     5TH JANUARY 1978
                    4 ;
                    5 ;WHEN USING THE MDS TEXT EDITOR (NO DOS) IT IS POSSIBLE FOR
                    6 ;TEXT TO BE LOST IF
                    7 ;A)     THE INTERRUPT 0 SWITCH IS OPERATED INADVERTANTLY OR
                    8 ;B)     AN 'E' IS ENTERED IN A COMMAND WHEN THE DESIGNATED
                    9 ;       SYSTEM PUNCH IS NOT AT THE TIME AVAILABLE IN WHICH
                   10 ;       CASE THE INTERRUPT 0 SWITCH MUST BE OPERATED
                   11 ;       DELIBERATELY TO ESCAPE.
                   12 ;TEXT IS NORMALLY LOST FOR GOOD.
                   13 ;THIS PROGRAM EXTRACTS TEXT FROM THE TEXT BUFFER MEMORY AREA
                   14 ;AND OUTPUTS IT VIA THE DESIGNATED SYSTEM PUNCH IN THE CORRECT
                   15 ;FORMAT FOR RE-ENTERING IT TO THE TEXT EDITOR USING THE 'A'
                   16 ;COMMAND.
                   17 ;IF TEXT HAS BEEN OVERWRITTEN THE PROGRAM WILL EXIT WITH
                   18 ;'ALL LOST' DISPLAYED ON THE CONSOLE DEVICE.  IF OK THEN
                   19 ;'ALL RECORDED' WILL BE DISPLAYED.  A CHECK IS NEVERTHELESS
                   20 ;ADVISABLE SINCE PART OF THE RECORDED TEXT MAY BE CORRUPT
                   21 ;IF OTHER PROCEDURES ARE TRIED BEFORE THIS PROGRAM IS RUN.
                   22 ;
00FF               23         BEGWD   EQU     0FFH    ;START WORD OF TEXT BUFFER
0D96               24         BUFLS   EQU     0D96H   ;BUFFER LENGTH STORE
F809               25         CO      EQU     0F809H  ;MONITOR CONSOLE O/P ROUTINE
000D               26         CR      EQU     0DH     ;ASCII CARRIAGE RETURN CHAR
001A               27         EOF     EQU     1AH     ;END OF FILE MARK/CONTROL Z
000A               28         LF      EQU     0AH     ;ASCII LINE FEED CHAR
F80C               29         PO      EQU     0F80CH  ;MONITOR PUNCH O/P ROUTINE
0E00               30         START   EQU     0E00H   ;BEGINNING OF TEXT BUFFER
                   31         ;
0020               32         ORG     20H
                   33         ;
0020 318400        34         LXI     SP,STAK ;LOAD STACK POINTER
0023 21960D        35         LXI     H,BUFLS ;LOAD ADDRESS OF BUFFER LENGTH STORE
0026 5E            36         MOV     E,M     ;LOAD LS BYTE OF BUFFER LENGTH WORD
0027 23            37         INX     H
0028 56            38         MOV     D,M     ;LOAD MS BYTE OF BUFFER LENGTH WORD
0029 21000E        39         LXI     H,START ;LOAD ADDR OF BEGINNING OF TEXT BUFFER
002C 7E            40         MOV     A,M     ;GET FIRST WORD
002D FEFF          41         CPI     BEGWD   ;IS IT THE START WORD?
002F C24700        42         JNZ     ERROR   ;NO-ALL LOST SO BRANCH FOR FAILURE
0032 23            43 AGAIN:  INX     H       ;ELSE ADDRESS NEXT WORD
0033 1B            44         DCX     D       ;AND DECR BUFFER LENGTH
0034 7A            45         MOV     A,D
0035 FE00          46         CPI     0
0037 C24000        47         JNZ     READ    ;IF MS BYTE >0 BRANCH FOR READ
003A 7B            48         MOV     A,E
003B FE00          49         CPI     0
003D CA5000        50         JZ      FINIS   ;IF LS BYTE=0 THEN BRANCH TO END
0040 4E            51 READ:   MOV     C,M     ;GET NEXT CHARACTER
0041 CD0CF8        52         CALL    PO      ;OUTPUT IT TO SYSTEM PUNCH
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

0044 C33200      53         JMP     AGAIN   ;AND SEE ABOUT ANOTHER CHARACTER
0047 216800      54 ERROR:  LXI     H,MES1  ;LOAD ADDRESS OF ERROR MESSAGE
004A 060A        55         MVI     B,LMES1 ;LOAD MESSAGE LENGTH
004C CD5E00      56         CALL    MSGL    ;OUTPUT MESSAGE
004F C7          57         RST     0       ;AND RETURN TO MONITOR
0050 0E1A        58 FINIS:  MVI     C,EOF   ;LOAD END OF FILE CHARACTER
0052 CD0CF8      59         CALL    PO      ;AND OUTPUT IT TO SYSTEM PUNCH
0055 217200      60         LXI     H,MES2  ;LOAD ADDRESS OF SUCCESS MESSAGE
0058 060E        61         MVI     B,LMES2 ;LOAD MESSAGE LENGTH
005A CD5E00      62         CALL    MSGL    ;OUTPUT MESSAGE
005D C7          63         RST     0       ;AND RETURN TO MONITOR
005E 4E          64 MSGL:   MOV     C,M     ;LOAD MESSAGE CHARACTER
005F CD09F8      65         CALL    CO      ;AND OUTPUT IT TO CONSOLE
0062 05          66         DCR     B       ;DECREMENT CHARACTER COUNT
0063 C8          67         RZ              ;RETURN IF ZERO=ALL DONE
0064 23          68         INX     H       ;ELSE SET FOR NEXT CHARACTER
0065 C35E00      69         JMP     MSGL    ;AND LOOP
0068 414C4C20    70 MES1:   DB      'ALL LOST',CR,LF
006C 4C4F5354
0070 0D
0071 0A
000A             71         LMES1   EQU     $-MES1
0072 414C4C20    72 MES2:   DB      'ALL RECORDED',CR,LF
0076 5245434F
007A 52444544
007E 0D
007F 0A
000E             73         LMES2   EQU     $-MES2
                 74         ;
0004             75 BLANK:  DS      4       ;STACK AREA
0084             76         STAK    EQU     $
                 77 ;
                 78 ;HAVING ASSEMBLED THIS PROGRAM RELOAD THE TEXT EDITOR AND
                 79 ;ENTER KNOWN TEXT.  MODIFY THIS TEXT AT WILL.  WHEN
                 80 ;SATISFIED OPERATE THE INTERRUPT 0 SWITCH TO RETURN TO THE
                 81 ;MONITOR.  CONTENTS OF THE TEXT BUFFER ARE NOW NOMINALLY LOST.
                 82 ;LOAD ASSEMBLED 'RETRIEVE'.  ENSURE THAT THE SYSTEM PUNCH IS
                 83 ;ON LINE AND THE 'G20'.
                 84 ;MESSAGE 'ALL RECORDED' WILL BE DISPLAYED ON THE CONSOLE WHEN
                 85 ;PUNCHING IS FINISHED.
                 86 ;MODIFY MEMORY LOCATION 0E00H FROM 0FFH TO ANY OTHER VALUE.
                 87 ;RELOAD 'RETRIEVE' AND 'G20'.
                 88 ;MESSAGE 'ALL LOST' WILL BE DISPLAYED INDICATING CORRUPTION
                 89 ;OF THE BEGINNING OF THE TEXT BUFFER AREA.
                 90 ;RELOAD TEXT EDITOR AND ENTER THE TAPE PRODUCED BY THE 'ALL
                 91 ;RECORDED' RUN USING THE 'A' COMMAND.  THE TEXT ENTERED WILL
                 92 ;BE FOUND TO BE IDENTICAL TO THAT LOST WHEN THE INTERRUPT 0
                 93 ;SWITCH WAS PRESSED.
                 94 ;
                 95         END

PUBLIC SYMBOLS

EXTERNAL SYMBOLS
```

```
USER SYMBOLS


ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0        MODULE   PAGE   3


AGAIN  A 0032    BEGWD  A 00FF    BLANK  A 0080    BUFLS  A 0096    CO     A F809    CR     A 000D    EOF    A 001A
ERROR  A 0047    FINIS  A 0050    LF     A 000A    LMES1  A 000A    LMES2  A 000E    MES1   A 0068    MES2   A 0072
MSGL   A 005E    PO     A F80C    READ   A 0040    STAK   A 0084    START  A 0E00

ASSEMBLY COMPLETE,  NO ERRORS
```

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☒8008 ☐8080 ☐8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

CALIBRATION OF AN EXTERNAL INPUT DEVICE (EID) FOR MAPPING WITH THE SURFACE OF A CRT TERMINAL

Function

This program has two independent parts. The first part initializes the CRT screen for the calibration procedure by displaying the active EID-area and marks where to calibrate the screen (EIDCAL-IN). The second part (EID-CALMA) of the calibration procedure collects the EID data of calibration points. The received data are transformed to the screen coordinate system and converted from BCD to binary. Now each coordinate pair is checked to determine if its X-coordinate is in the approximate area of the calibration spots. After the valid data of the last spot is accepted, the following definements and computations are done: (1) left and right boundary of area (2) bottom and top boundary of area (3) area increments in X and Y axis (4) screen character per TSD digit in X and Y axis.

Required Software

Fixed point package.
BCD - binary conversion

Input Parameters

EID - digits from the EID-service-routine.

Output Results

Computations: Left and right boundary of device area.
Bottom and top boundary of device area.
Screen characters per EID-digit.

Flags: Device calibrated.

AB128A is offered as one program with AB128B.

Registers Modified:	Programmer: Dr. Theodor Luetzeler
RAM Required: 1A H	Company: Siemens Corporation
ROM Required: C29 H	Address: 3 Computer Drive
Maximum Subroutine Nesting Level: 6 H	City: Cherry Hill
Assembler/Compiler Used: PLM 80	State: New Jersey

Ref. #AB128B

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/4040 ☐ 8008 ☒ 8080 ☐ 8048 ☐ 8085 ☐ Other _____ (use additional sheets if necessary)

Program Title

TRANSFORMATION ROUTINE FOR COORDINATE DIGITS FROM AN EXTERNAL INPUT DEVICE (EID) INTO X- Y- COORDINATES OF A CRT TERMINAL

Function

This program supports a hardware arrangement of an external input device, which produces coordinate digits. The coordinates are displayed on a CRT terminal. The significant steps are: Coordinate origin transformation. BCD to binary conversion. Limit checking. Computations. Preparation for cursor positioning.

Required Hardware

Required Software

Fixed point package
BCD to binary conversion

Input Parameters

X and Y digits from the service routine of the EID.
Output of the calibration procedure (see separate program).

Output Results

X and Y coordinates of the CRT terminal screen
Flag, which indicates that the input device was activated.
Flag, which requests to position the cursor at the calculated spot.
Data to position the cursor.

AB128B is offered as one program with AB128A.

Registers Modified:	Programmer: Dr. Theodor Luetzeler
RAM Required: 5A H	Company: Siemens Corporation
ROM Required: 364 H	Address: 3 Computer Drive
Maximum Subroutine Nesting Level: 6 H	City: Cherry Hill
Assembler/Compiler Used: PLM 80	State: New Jersey

Ref. #AB129

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 XX8080 ☐8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

CHECK SUM

Function

This program calculates two verification digits for data string until 1K bytes, and types them out on the console output device. It uses the ISIS-II system calls and it's invoked by the command CHKSUM.ABS <FILENAME>, where <FILENAME> is the name of the hexadecimal file that has the data string input. The calculation way is a simple summation of all input bytes, with overflow from the low checksum to the high checksum. The two output digits could be used for ROM check purposes.

Required Hardware

Intellec MDS with 32K RAM

Required Software

ISIS-II operating system

Input Parameters

Diskette with CHKSUM.ABS program and with data string input file.
CHKSUM.ABS <FILENAME> from the console.

Output Results

Two hexadecimal digits on the console.

Registers Modified:	Programmer: DIEGO SANCHEZ HERNANDEZ
RAM Required:	Company: G.E.E. - Electromedicina
ROM Required:	Address: Doctor Esquerdo, 187
Maximum Subroutine Nesting Level:	City: MADRID.- 7
Assembler/Compiler Used: ISIS-II 8080/8085 Macro Assembler V2.0	State: SPAIN

```
ASM80 :F1:CHKSUM.SRC

ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0        MODULE  PAGE   1

 LOC  OBJ        SEQ        SOURCE STATEMENT

                   1 ; CHECK SUM.
                   2 ; THIS PROGRAM CALCULATES TWO VERIFICATION DIGITS FOR DATA
                   3 ; STRING UNTIL 1K BYTES, AND TYPES OUT THEM ON THE CONSOLE
                   4 ; OUTPUT DEVICE.
                   5 ; IT USES THE ISIS-II SYSTEM CALLS AND IT IS INVOKED BY THE
                   6 ; COMMAND :
                   7 ;             CHKSUM.ABS <FILENAME>
                   8 ; WHERE <FILENAME> IS THE NAME OF THE HEXADECIMAL FILE THAT
                   9 ; HAS THE DATA STRING INPUT.
                  10 ; THE CALCULATION WAY IS A SIMPLE SUMMATION OF ALL INPUT
                  11 ; BYTES, WITH OVERFLOW FROM THE LOW CHECKSUM TO THE HIGH
                  12 ; CHECKSUM.
                  13 ; THE TWO OUTPUT DIGITS COULD BE USED FOR ROM CHECK PURPOSES.
                  14 ; MARCH-1978.  DIEGO SANCHEZ.
                  15 ;
                  16 ;
                  17 ;
                  18 $TITLE('SYMBOLS')
                  19 $EJECT
```

SYMBOLS

```
LOC  OBJ        SEQ        SOURCE STATEMENT

0000              20 OPEN   EQU    0
0001              21 CLOSE  EQU    1
0003              22 READ   EQU    3
0004              23 WRITE  EQU    4
0009              24 EXIT   EQU    9
000C              25 ERROR  EQU    12
                  26 ;
                  27 ;
                  28 ;
                  29 $TITLE('MAIN PROGRAM')
                  30 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                    31        EXTRN   ISIS
                    32 ;
                    33        CSEG                    ; BEGINING OF CODE SEGMENT
                    34 ;
0000 310000   S     35 CHKSUM: LXI    SP,STACK        ; STACK INITIALIZATION
                    36 ;
0003 0E03           37        MVI     C,READ          ; READ FILENAME FROM THE :CO:
0005 110000   D     38        LXI     D,RBLK
0008 CD0000   E     39        CALL    ISIS
000B 3A2200   D     40        LDA     STATUS
000E B7             41        ORA     A
000F C28D00   C     42        JNZ     ERR
                    43 ;
0012 0E00           44        MVI     C,OPEN          ; OPEN THE INPUT FILE
0014 110A00   D     45        LXI     D,OBLK
0017 CD0000   E     46        CALL    ISIS
001A 3A2200   D     47        LDA     STATUS
001D B7             48        ORA     A
001E C28D00   C     49        JNZ     ERR
0021 2A0000   D     50        LHLD    AFT
0024 221C00   D     51        SHLD    CAFT
                    52 ;
0027 0E03           53        MVI     C,READ          ; READ DE INPUT FILE
0029 110000   D     54        LXI     D,RBLK
002C CD0000   E     55        CALL    ISIS
002F 3A2200   D     56        LDA     STATUS
0032 B7             57        ORA     A
0033 C28D00   C     58        JNZ     ERR
                    59 ;
0036 1600           60        MVI     D,0             ; BEGINING OF THE PROCESS
0038 1E00           61        MVI     E,0             ; D,E = HIGH,LOW CHKSUM
003A 212800   D     62        LXI     H,BUFFER
                    63 ;
003D 3E3A           64 LOOP1:  MVI     A,3AH           ; A = :
                    65 ;
003F BE             66 LOOP2:  CMP     M
0040 23             67        INX     H
0041 C23F00   C     68        JNZ     LOOP2
                    69 ;
0044 CD9D00   C     70        CALL    PACKET
0047 CA6200   C     71        JZ      DONE
004A 010700         72        LXI     B,7
004D 09             73        DAD     B               ; JUMP 6 BYTES
004E 47             74        MOV     B,A             ; B = COUNTER
                    75 ;
004F 3E3A           76 SUM:    MVI     A,3AH           ; A = :
0051 CD9D00   C     77        CALL    PACKET
0054 83             78        ADD     E               ; A = BYTE TO ADD
0055 D25900   C     79        JNC     SUM1
0058 14             80        INR     D
0059 5F             81 SUM1:   MOV     E,A
005A 23             82        INX     H
005B 05             83        DCR     B
005C C24F00   C     84        JNZ     SUM
005F C33D00   C     85        JMP     LOOP1
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                    86 ;
0062 21280C   D     87 DONE:  LXI     H,CHECKN        ; STORE CHECK SUM
0065 72             88        MOV     M,D             ; HIGH CHECK SUM
0066 CDB900   C     89        CALL    UNPACK
0069 23             90        INX     H
006A 73             91        MOV     M,E             ; LOW CHECK SUM
006B CDB900   C     92        CALL    UNPACK
                    93 ;
006E 0E04           94        MVI     C,WRITE         ; WRITE TO THE CONSOLE
0070 111400   D     95        LXI     D,WBLK
0073 CD0000   E     96        CALL    ISIS
0076 3A2200   D     97        LDA     STATUS
0079 B7             98        ORA     A
007A C28000   C     99        JNZ     ERR
                   100
007D 0E01          101        MVI     C,CLOSE         ; CLOSE THE INPUT FILE
007F 111C00   D    102        LXI     D,CBLK
0082 CD0000   E    103        CALL    ISIS
0085 0E09          104        MVI     C,EXIT          ; NORMAL EXIT
0087 112000   D    105        LXI     D,XBLK
008A CD0000   E    106        CALL    ISIS
                   107 ;
008D 0E0C          108 ERR:   MVI     C,ERROR
008F 112200   D    109        LXI     D,EBLK
0092 CD0000   E    110        CALL    ISIS
0095 0E09          111        MVI     C,EXIT          ; ERROR EXIT
0097 112000   D    112        LXI     D,XBLK
009A CD0000   E    113        CALL    ISIS
                   114 ;
                   115 ;
                   116 ;
                   117 $TITLE('PACK')
                   118 $EJECT
```

```
 LOC  OBJ        SEQ        SOURCE STATEMENT

                  119 ; SUBROUTINE THAT PACKS TWO HEXADECIMAL DIGITS INTO ONE
                  120 ; BYTE AND PUTS IT IN THE ACCUMULATOR.
                  121 ; LIKE INPUT PARAMETERS H&L REGISTERS MUST POINT AT THE
                  122 ; TWO DIGITS, AND THE ACCUMULATOR MUST BE EQUAL TO 3AH.
                  123 ;
 009D BE          124 PACKET: CMP     M              ; A = :
 009E 7E          125         MOV     A,M
 009F D2A400   C  126         JNC     NUMBER
 00A2 D637        127         SUI     37H            ; LETTER
 00A4 E60F        128 NUMBER: ANI     0FH
 00A6 07          129         RLC
 00A7 07          130         RLC
 00A8 07          131         RLC
 00A9 07          132         RLC
 00AA 4F          133         MOV     C,A            ; C = 4 MSB
                  134 ;
 00AB 3E3A        135         MVI     A,3AH          ; A = :
 00AD 23          136         INX     H
 00AE BE          137         CMP     M
 00AF 7E          138         MOV     A,M
 00B0 D2B500   C  139         JNC     NUMBR2
 00B3 D637        140         SUI     37H            ; LETTER
 00B5 E60F        141 NUMBR2: ANI     0FH
 00B7 B1          142         ORA     C              ; A = PACKET NUMBER
 00B8 C9          143         RET
                  144 ;
                  145 ;
                  146 ;
                  147 $TITLE('UNPACK')
                  148 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                  149 ; SUBROUTINE THAT UNPACKS ONE BYTE INTO TWO HEXADECIMAL
                  150 ; DIGITS.
                  151 ; H&L REGISTERS MUST POINT AT THE INPUT BYTE, AND THE
                  152 ; TWO OUTPUT DIGITS WILL BE IN THE SAME MEMORY POSITION
                  153 ; AND IN THE NEXT.
                  154 ;
00B9 3EF0         155 UNPACK: MVI     A,0F0H
00BB A6           156         ANA     M
00BC 0F           157         RRC
00BD 0F           158         RRC
00BE 0F           159         RRC
00BF 0F           160         RRC
00C0 FE0A         161         CPI     0AH
00C2 DAC700   C   162         JC      NMBR
00C5 C607         163         ADI     7               ; LETTER
00C7 C630         164 NMBR:   ADI     30H
00C9 47           165         MOV     B,A
00CA 3E0F         166         MVI     A,0FH
00CC A6           167         ANA     M
00CD FE0A         168         CPI     0AH
00CF DAD400   C   169         JC      NMBR1
00D2 C607         170         ADI     7               ; LETTER
00D4 C630         171 NMBR1:  ADI     30H
00D6 70           172         MOV     M,B
00D7 23           173         INX     H
00D8 77           174         MOV     M,A
00D9 C9           175         RET
                  176 ;
                  177 ;
                  178 ;
                  179 $TITLE('BUFFERS AND DATA')
                  180 $EJECT
```

```
 LOC  OBJ          SEQ          SOURCE STATEMENT

                   181          DSEG                    ; BEGINING OF DATA SEGMENT
                   182 ;
                   183 RBLK:
 0000 0100         184 AFT:     DW      1
 0002 2800      D  185          DW      BUFFER
 0004 000C         186          DW      3072
 0006 2600      D  187          DW      ACTUAL
 0008 2200      D  188          DW      STATUS
                   189 ;
 000A 0000      D  190 OBLK:     DW      AFT
 000C 2800      D  191          DW      BUFFER
 000E 0300         192          DW      3
 0010 0000         193          DW      0
 0012 2200      D  194          DW      STATUS
                   195 ;
 0014 0000         196 WBLK:    DW      0
 0016 280C      D  197          DW      CHECKN
 0018 0400         198          DW      4
 001A 2200      D  199          DW      STATUS
                   200 ;
                   201 CBLK:
 0002              202 CAFT:    DS      2
 001E 2200      D  203          DW      STATUS
                   204 ;
 0020 2200      D  205 XBLK:    DW      STATUS
                   206 ;
                   207 EBLK:
 0002              208 STATUS:  DS      2
 0024 2200      D  209          DW      STATUS
                   210 ;
 0002              211 ACTUAL:  DS      2
                   212 ;
 0C00              213 BUFFER:  DS      3072
                   214 ;
 0004              215 CHECKN:  DS      4
                   216 ;
 0000           C  217          END     CHKSUM
```

PUBLIC SYMBOLS

EXTERNAL SYMBOLS
ISIS E 0000

USER SYMBOLS

ACTUAL	D 0026	AFT	D 0000	BUFFER	D 0028	CAFT	D 001C	CBLK	D 001C	CHECKN	D 0C28	CHKSUM	C 0000
CLOSE	A 0001	DONE	C 0062	EBLK	D 0022	ERR	C 008D	ERROR	A 000C	EXIT	A 0009	ISIS	E 0000
LOOP1	C 003D	LOOP2	C 003F	NMBR	C 00C7	NMBR1	C 00D4	NUMBER	C 00A4	NUMBR2	C 00B5	OBLK	D 000A
OPEN	A 0000	PACKET	C 009D	RBLK	D 0000	READ	A 0003	STATUS	D 0022	SUM	C 004F	SUM1	C 0059
UNPACK	C 00B9	WBLK	D 0014	WRITE	A 0004	XBLK	D 0020						

ASSEMBLY COMPLETE, NO ERRORS

Ref.#AE15

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 ☒8080 ☐8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

WRITEP - Output Procedures for PLM80

Function

Contains several procedures to call by PLM80 programs for formatted output of ADDRESS or BYTE values, or output of characters and strings.

Required Hardware

MDS Intellec 800

Required Software

PLM80
The program uses the ISIS-call WRITE for doing its output, but may easily be changed in order to use another output procedure, and thus become independent of ISIS.

Input Parameters

See Attached.

Output Results

See Attached.

Registers Modified: ALL	Programmer: Karl Pentzlin
RAM Required: 00CEH	Company: Informatik-Forum GmbH
ROM Required: 0AF4H	Address: Fasanenstraße 16
Maximum Subroutine Nesting Level: 4	City: D-8000 München 60
Assembler/Compiler Used: PLM80 Compiler V.3.0	State: WEST GERMANY

WRITEP contains procedures as listed below:

1. Formatted Output Procedures:

   CALL WRITEA (address$value, format)

   CALL WRITEB (byte$value, format)

   where "format" is to be declared as ADDRESS parameter and has to be a string of one or two characters. The first character specifies the wanted format, as listed below. The second character, if given, specifies the number of digits to be printed (not at formats L and Y).

   B Binary format. Default length: 16 for WRITEA, 8 for WRITEB

   O or Q: Octal format. Default length: 6 for WRITEA, 3 for WRITEB

   D Decimal format. Default length: 5 for WRITEA, 3 for WRITEB

   H Hex format. Default length: 4 for WRITEA, 2 for WRITEB

   I Integer format. Decimal digits, leading zeroes are replaced by spaces. Default length: Number of digits to be printed without leading zeroes, depending on specified value.

   S Signed integer format. Like Integer format, but, if most significant bit of value is set, it is interpreted as negative sign.

   L Logical format. The second character of the format string (must not be a format character. Default is "+") is printed, if the specified value is "true". If "false", a space is printed.

   N Name format. Only for WRITEB. The name of the specified value interpreted as an ASCII character is printed (i.e., "LF" for 0AH). Default length: Length of the name. For all names check source file.

   Y Two bytes octal format. Only for WRITEA.
   High and low byte of specified value are printed by 3 octal digits each, and are separated by the second character of the format string (which must not be a format character. Default is "#"). Therefore, the format length is 7.

   Moreover, there is an error format, which is used if the format string does not begin with a format character. It is like format H, with additional printing of the erroneous format string.

2. Other Output procedures:

   CALL WRITEL
   Print the end of a line (that is, CR and LF).

   CALL WRITEC (character, count)
   Write the specified character <count> times. "character" is to be declared as BYTE parameter, "count" as ADDRESS parameter.

   CALL WRITES (stringpointer)
   Write the string based by "stringpointer" (which is to be declared as ADDRESS parameter). The string may contain any ASCII characters and is to be terminated with an ETX (End of Text, 03H) character.

CALL WRITEN (stringpointer, count)
Write the string based by "stringpointer", and the length of which is <count> (both parameters are to be declared ADDRESS).

CALL WRITEV (stringpointer, count)
Like WRITEN, but invisible ASCII characters (like CR, LF) are replaced by visible ones (CR and LF by "/", HT (Horizontal Tabulation) by the inverse slash, all others by "!").

3. Output Control procedure:

CALL WRITED (aftn)
Change output device. "aftn" (which is to be declared as ADDRESS parameter) specifies the output device by its AFTN, as it is given by the ISIS-call OPEN. Console output (:CO:) is specified by zero. At beginning of a program run, console output is output device for all WRITEP procedures.

WRITEP uses an internal buffer for all output done by WRITEP procedures except when output device is console output (console output is always done immediately during each WRITEP procedure call).
Therefore, regard this notice:

Except if you output on console, you have to call WRITED before you want to output on the same device as used by WRITEP by other than WRITEP procedures (e.g. by ISIS-call WRITE), and you have to call WRITED at the end of your program run, due to the fact that WRITED empties the internal buffer before regarding the new "aftn" value (which, in fact, may be the same as the old one).

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/4040 ☐ 8008 ☒ 8080 ☐ 8048 ☐ 8085 ☐ Other _____ (use additional sheets if necessary)

Program Title

ERRORP/MESSGP - Printing Subroutine for ISIS-II Error Messages and User Messages with Filename

Function

ERRORP prints an ISIS-II error message, together with a filename, if one is specified.

MESSGP does the same with a user defined message.

Required Hardware

MDS 800 Intellec

Required Software

ISIS-II
The system calls WRITE and ERROR are used.

Input Parameters

See Attached.

Output Results

See Attached.

Registers Modified: ASSUME ALL	Programmer: Karl Pentzlin
RAM Required: 0020H	Company: Informatik-Forum GmbH
ROM Required: 061BH	Address: Fasanenstraße 16
Maximum Subroutine Nesting Level: 2	City: D-8000 München 60
Assembler/Compiler Used: PL/M-80 Compiler V.3.0	State: WEST GERMANY

ERRORP is to be declared as follows:

```
ERRORP: PROCEDURE (ERRORNUMBER, FILENAME) EXTERNAL;
        DECLARE   (ERRORNUMBER, FILENAME) ADDRESS; END;
```

ERRORNUMBER contains the error number as given by any ISIS-II system call by the STATUS parameter. If ERRORNUMBER = 0, (that is, the ISIS-II system call was done without an error) the call of ERRORP will have no effect. Otherwise, an error message will be output as readable text, e.g. "NO SUCH FILE" if ERRORNUMBER = 13.

FILENAME either contains 0 (then only the error message will be printed), or is a pointer to the beginning of an ISIS-II filename or device specification. Then this name will be printed before the error message, separated by a comma (without leading spaces, if any).

MESSGP is to be declared as follows:

```
MESSGP: PROCEDURE (TEXTPOINTER, FILENAME) EXTERNAL,
        DECLARE   (TEXTPOINTER, FILENAME) ADDRESS; END;
```

TEXTPOINTER is a pointer to the beginning of an ASCII string representing the user defined message. The text has to be concluded by an ETX (End of Text, 03H) byte.

FILENAME has the same function as for ERRORP.

All printing will be done on :CO: (Console Output).
Before printing a message, a Carriage Return/Line Feed will be output.

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☒8008 ☐8080 ☐8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

FORMFD-FORMFEED TO LINEFEED CONVERSION PROGRAM

Function

This program will convert all formfeeds in a program listing, except the first one, to line feeds. The program was created for use with the Intel MDS-770 lineprinter. The program will add a paging feature to the 770 printer.

Required Hardware

MDS-800 or SERIES-II Development System

Required Software

ISIS-II Operating System
MDS System Monitor

Input Parameters

Syntax as follows:

FORMFD filename.LST To :LP: (To Lineprinter)
To :TO: (To Teletype List)
To filename.EXT (To Store On Disk)

Output Results

The output device will list your program with the appropriate number of Linefeeds replacing all but the first formfeed.

Registers Modified:	Programmer: Mary Jane Elmore
RAM Required: 32K	Company: Intel Corp.
ROM Required:	Address: 3065 Bowers Ave.
Maximum Subroutine Nesting Level:	City: Santa Clara
Assembler/Compiler Used: PL/M 80	State: Calif. 95051

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE ERRBLOCK
OBJECT MODULE PLACED IN :F4:UTILS.OBJ
COMPILER INVOKED BY:  PLM80 :F4:UTILS.PLM  WORKFILES(:F4:,:F4:)


               $DEBUG PAGEWIDTH(80)

                /*PROCEDURE TO SEND ERROR MESSAGE TO CONSOLE */


  1            ERRBLOCK:    DO;



  2   1        EXIT:
                   PROCEDURE EXTERNAL;
  3   2        END EXIT;


  4   1        ERROR:
                   PROCEDURE(ERRNUM) EXTERNAL;
  5   2          DECLARE ERRNUM ADDRESS;
  6   2        END ERROR;


  7   1        ERRPRT:  PROCEDURE(ERRNO) PUBLIC;
  8   2            DECLARE ERRNO ADDRESS;

  9   2        CALL ERROR(ERRNO);

 10   2        CALL EXIT;

 11   2        END ERRPRT;

 12   1        END ERRBLOCK;


MODULE INFORMATION:

     CODE AREA SIZE     = 0012H     18D
     VARIABLE AREA SIZE = 0002H      2D
     MAXIMUM STACK SIZE = 0002H      2D
     30 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE FORMFD
OBJECT MODULE PLACED IN :F1:FORMFD.OBJ
COMPILER INVOKED BY:  :F1:PLM80 :F1:FORMFD.PLM


             $DEBUG PAGEWIDTH(80)

               /* THIS PROGRAM WILL CONVERT ALL BUT THE FIRST FORMFEED IN &
             YOUR PROGRAM AND LISTING TO LINEFEEDS. */


  1          FORMFD: DO;

  2    1       DECLARE BUFFER(128) BYTE;
  3    1       DECLARE OUTBUF(2000) BYTE;
  4    1       DECLARE ACTUAL$COUNT ADDRESS;
  5    1       DECLARE STATUS ADDRESS;
  6    1       DECLARE AFT$IN ADDRESS;
  7    1       DECLARE AFT$OUT ADDRESS;
  8    1       DECLARE READ$ACCESS LITERALLY '1';
  9    1       DECLARE WRITE$ACCESS LITERALLY '2';
 10    1       DECLARE NOT$END$OF$FILE LITERALLY 'ACTUAL$COUNT <> 0' ;
 11    1       DECLARE ERRMSG(*) BYTE DATA ('ERROR IN COMMAND TAIL',0DH,0AH);
 12    1       DECLARE OUT$BUF$INDEX BYTE;
 13    1       DECLARE READ$BUF$INDEX ADDRESS;
 14    1       DECLARE WRITE$BUF$INDEX ADDRESS;
 15    1       DECLARE LINE$COUNT BYTE;
 16    1       DECLARE I BYTE;
 17    1       DECLARE LINE$FEED LITERALLY '0AH';
 18    1       DECLARE FORM$FEED LITERALLY '0CH';
 19    1       DECLARE FIRST$READ BYTE;
 20    1       DECLARE TRUE LITERALLY '1';
 21    1       DECLARE FALSE LITERALLY '0';


 22    1       OPEN:
                 PROCEDURE(AFT,FILE,ACCESS,MODE,STATUS) EXTERNAL;
 23    2           DECLARE(AFT,FILE,ACCESS,MODE,STATUS) ADDRESS;
 24    2       END OPEN;


 25    1       CLOSE:
                 PROCEDURE(AFT,STATUS) EXTERNAL;
 26    2           DECLARE(AFT,STATUS) ADDRESS;
 27    2       END CLOSE;

 28    1       READ:
                 PROCEDURE(AFT,BUFFER,COUNT,ACTUAL,STATUS) EXTERNAL;
 29    2           DECLARE(AFT,BUFFER,COUNT,ACTUAL,STATUS) ADDRESS;
 30    2       END READ;

 31    1       WRITE:
                 PROCEDURE(AFT,BUFFER,COUNT,STATUS) EXTERNAL;
 32    2           DECLARE(AFT,BUFFER,COUNT,STATUS) ADDRESS;
 33    2       END WRITE;
```

```
 34   1          EXIT:
                   PROCEDURE EXTERNAL;
 35   2          END EXIT;

 36   1          ERROR:
                   PROCEDURE (ERRNUM) EXTERNAL;
 37   2             DECLARE ERRNUM ADDRESS;
 38   2          END ERROR;

 39   1          ERRPRT:
                   PROCEDURE (ERRNO) EXTERNAL;
 40   2             DECLARE ERRNO ADDRESS;
 41   2          END ERRPRT;



              /*READ THE CONSOLE FILE TO GET THE PARAMETER STRING*/



 42   1       CALL READ(1,.BUFFER,128,.ACTUAL$COUNT,.STATUS);
 43   1            IF STATUS >0 THEN CALL ERRPRT(STATUS);
 45   1       CALL OPEN(.AFT$IN,.BUFFER,READ$ACCESS,0,.STATUS);
 46   1            IF STATUS >0 THEN CALL ERRPRT(STATUS);



              /* OUTPUT BUFFER INDEX LOOP*/



 48   1       OUT$BUF$INDEX = 0;
 49   1       I = 0;

 50   1       DO WHILE OUT$BUF$INDEX = 0 AND I <= ACTUAL$COUNT;
 51   2          IF BUFFER(I) = ' ' THEN
 52   2             IF BUFFER(I+1) = 'T' THEN
 53   2                IF BUFFER(I+2) = '0' THEN
 54   2                   IF BUFFER(I+3) = ' ' THEN

 55   2                      OUT$BUF$INDEX = I+4;
 56   2       I=I+1;

 57   2       END;


 58   1       IF OUT$BUF$INDEX = 0 THEN

 59   1         DO;
 60   2           CALL WRITE(0,.ERRMSG,23,.STATUS);
 61   2           CALL EXIT;
 62   2         END;

 63   1       CALL OPEN(.AFT$OUT,.BUFFER(OUT$BUF$INDEX),WRITE$ACCESS,0,.STATUS);
 64   1            IF STATUS >0 THEN CALL ERRPRT(STATUS);
```

```
                /* CHANGE FORMFEEDS TO LINEFEEDS */


 66   1        FIRST$READ = TRUE;
 67   1        WRITE$BUF$INDEX = 0;
 68   1        READ$BUF$INDEX = 0;
 69   1        LINE$COUNT = 66;
 70   1        ACTUAL$COUNT = 1;

 71   1           CALL READ(AFT$IN,.BUFFER,128,.ACTUAL$COUNT,.STATUS);


 72   1        DO WHILE NOT$END$OF$FILE;
 73   2          IF STATUS > 0 THEN CALL ERRPRT(STATUS);

 75   2          IF FIRST$READ = TRUE THEN

 76   2            DO;

 77   3              OUTBUF(0) = BUFFER(0);
 78   3              WRITE$BUF$INDEX = WRITE$BUF$INDEX + 1;

 79   3                DO READ$BUF$INDEX = 1 TO ACTUAL$COUNT - 1;

 80   4                  IF BUFFER(READ$BUF$INDEX) = FORM$FEED THEN
 81   4                    DO;

 82   5                      DO I = 0 TO LINE$COUNT - 1;
 83   6                        OUTBUF(WRITE$BUF$INDEX) = LINE$FEED;
 84   6                        WRITE$BUF$INDEX = WRITE$BUF$INDEX + 1;
 85   6                      END;

 86   5                      LINE$COUNT = 66;

 87   5                    END;

 88   4                    ELSE DO;
 89   5                      OUTBUF(WRITE$BUF$INDEX) = BUFFER(READ$BUF$INDEX);
 90   5                    WRITE$BUF$INDEX = WRITE$BUF$INDEX + 1;
 91   5                    END;

 92   4                  IF BUFFER(READ$BUF$INDEX) = LINE$FEED THEN
 93   4                  LINE$COUNT = LINE$COUNT -1;

 94   4                  IF LINE$COUNT = 0 THEN
 95   4                  LINE$COUNT = 66;


 96   4                END;

 97   3              FIRST$READ = FALSE;

 98   3            END;
```

```
 99   2          ELSE DO;


100   3            DO READ$BUF$INDEX = 0 TO ACTUAL$COUNT - 1;
101   4              IF BUFFER(READ$BUF$INDEX) =FORM$FEED THEN
102   4                DO;

103   5                  DO I = 0 TO LINE$COUNT - 1;
104   6                    OUTBUF(WRITE$BUF$INDEX) = LINE$FEED;
105   6                    WRITE$BUF$INDEX = WRITE$BUF$INDEX + 1;
106   6                  END;

107   5                  LINE$COUNT = 66;

108   5                END;

109   4                ELSE DO;
110   5                  OUTBUF(WRITE$BUF$INDEX) = BUFFER(READ$BUF$INDEX);
111   5                  WRITE$BUF$INDEX = WRITE$BUF$INDEX + 1;
112   5                END;


113   4              IF BUFFER(READ$BUF$INDEX) = LINE$FEED THEN
114   4              LINE$COUNT = LINE$COUNT -1;

115   4              IF LINE$COUNT = 0 THEN

116   4              LINE$COUNT = 66;




117   4              END;
118   3            END;

119   2          CALL WRITE(AFT$OUT,.OUTBUF,WRITE$BUF$INDEX,.STATUS);
120   2          IF STATUS > 0 THEN CALL ERRPRT(STATUS);
122   2          WRITE$BUF$INDEX = 0;

123   2          CALL READ(AFT$IN,.BUFFER,128,.ACTUAL$COUNT,.STATUS);
124   2        END;


125   1        CALL CLOSE(AFT$IN,.STATUS);
126   1            IF STATUS >0 THEN CALL ERRPRT(STATUS);

128   1        CALL CLOSE(AFT$OUT,.STATUS);
129   1            IF STATUS >0 THEN CALL ERRPRT(STATUS);


131   1        CALL EXIT;


132   1        END FORMFD;
```

```
MODULE INFORMATION:

     CODE AREA SIZE     = 033BH    827D
     VARIABLE AREA SIZE = 0860H   2144D
     MAXIMUM STACK SIZE = 0008H      8D
     220 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

Ref.#AB131

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☒8008 ☐8080 ☐8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

MONITOR ROUTINES FOR A 3M DCD1 CASSETTE TAPE DRIVE

Function

To record and play-back records from a SBC system to a 3M DCD1 tape cassette. Intended as an extension of the SBC-monitor.

Required Hardware

TTY or equiv. serial terminal. One 24 bits parrallel port (8255)
One 3M DCD1 tape cassette drive incl. electronics.
One 74123 TTL circuit or equiv. Six 220Ωpull up resistors.

Required Software

SBC 80P monitor or equiv. routines. The tape commands can easily be incorporated into the SBC monitor.

Input Parameters

1) C↵ on the TTY to print the catalogue of the tape.

2) O↵ on the TTY to dump the content of a memory area onto the tape and get the catalogue updated.
The start address, end address and record number will be asked for by the routine and must be given by the user.

3) P↵ on the TTY to play back a record from the tape into the memory. The start address, address limit and record number will be asked for by the routine and must be given by the user.

Output Results

The program will continuously check the status of the tape drive and give a message if anything is wrong. The output routine will checkread the written record and then update the catalogue of the tape. In case of read error the program pushes the addresses of the incorrect bytes during the read operation, and then lists these addresses on the TTY.

For further information, see the demonstration run or contact the programmer.

Registers Modified: All	Programmer: Per Nylen
RAM Required: 1 k bytes	Company: University of Stockholm
ROM Required: 1075D bytes	Address: Vanadisvagen 9
Maximum Subroutine Nesting Level:	City: S-113 46 Stockholm
Assembler/Compiler Used: ASM/20	State: Sweden

Ref.#AB132

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/4040 ☐ 8008 ☒ 8080 ☐ 8048 ☐ 8085 ☐ Other_____ (use additional sheets if necessary)

Program Title

ALPHA: AN ALPHABETIZED LISTING OF THE DISK DIRECTORY OUTPUT TO THE LINE PRINTER

Function

1) Stack pointer set, 2) system list device set to line printer, 3) file :F0: ALPHA DIR opened and read, 4) first two lines of directory printer, 5) table built of addresses of entries in the directory in alphabetical order, 6) the lines of the directory printed as they are listed in the table, 7) the last two lines of the directory printed, 8) the system list device reset, 9) program exited and control returned to ISIS-II.

Required Hardware

MDS
disk drive: F0:
line printer
console input (CRT or TTY)

Required Software

ISIS-II (Tested on V2.2D, V3.4D)
Intellect Monitor

Input Parameters

:F0: ALPHA. DIR Input to buffer in RAM (IBUF)

Output Results

Directory is output character by character from Register C by calling the LO routine.

I/O operations are performed by the ISIS-II subroutine calls and driver routines.

Registers Modified: A,F,B,C,D,E,H,I,SP,PC	Programmer: S. Bann
RAM Required: Module is located 34ØØ to 56BAH 229BH + BUFFERS	Company: Xerox
ROM Required: MDS Monitor loaded in ROM	Address: 701 S. Aviation Blvd. A2-43
Maximum Subroutine Nesting Level: One level	City: El Sigundo
Assembler/Compiler Used: ISIS-II 8080/8085 Macro Assembler V2.0	State: CA 90245

```
ASM80 :F1:ALPHA.SRC

ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0        MODULE   PAGE   1
PRINT ALPHABETIZED DIRECTORY

 LOC  OBJ         SEQ        SOURCE STATEMENT

                    1 $       TITLE('PRINT ALPHABETIZED DIRECTORY')
                    2
                    3
                    4 ;PROGRAM NAME: ALPHA
                    5 ;WRITTEN BY: SHERRI ANN BANN
                    6 ;DATE: JUNE 30, 1978
                    7 ;REVISION: A.00
                    8
                    9 ;ALPHA IS EXECUTED UNDER THE CONTROL OF ISIS-2
                   10
                   11 ;THIS PROGRAM TAKES AS INPUT :F0:ALPHA.DIR, A FILE CONTAINING THE
                   12 ;UNALPHABETIZED DIRECTORY LISTING CREATED WITH THE ISIS-2 DIR COMMAND.
                   13
                   14 ;IT CREATES AS AN OUTPUT BUFFER A TABLE OF POINTERS(ADDRESSES) WHICH
                   15 ;REFER TO THE FIRST BYTES OF FILENAMES IN THE INPUT FILE ALPHABETICALLY,
                   16 ;(I.E. THE FIRST POINTER AT THE TOP OF THE OUTPUT FILE REFERS TO THE FIRST
                   17 ;FILENAME TO BE OUTPUT IN ALPHABETICAL ORDER
                   18
                   19 ;IT OUTPUTS THE ALPHABETIZED DIRECTORY TO THE LINE PRINTER.
                   20
                   21 ;ALPHA ASSUMES:  1) THE DIRECTORY LISTING CONSISTS OF 2 LINES OF HEADER,
                   22 ;A VARIABLE NUMBER OF LINES OF FILENAMES AND THEIR LENGTHS AND ATTRIBUTES,
                   23 ;AND 2 LINES OF ENDING;  2) EACH LINE IS SEPARATED BY A LINE FEED CHARACTER;
                   24 ;3) THE SECOND TO THE LAST LINE IS UNIQUE BECAUSE IT STARTS WITH A SPACE;
                   25 ;4) EACH FILENAME IS UNIQUE.
                   26
                   27 ;THESE ARE JUSTIFIED ASSUMPTIONS UNDER V2.2D AND V3.4D ISIS-2 SOFTWARE
                   28
                   29 ;SYSTEM CALLS
                   30                EXTRN   IOSET                ;SET I/O CONFIGURATION
                   31                EXTRN   IOCHK                ;GET I/O CONFIGURATION
                   32                EXTRN   LO                   ;LIST OUTPUT
                   33                EXTRN   ISIS
                   34
                   35
                   36                DSEG
                   37
0000               38        OPEN    EQU     0
0003               39        READ    EQU     3
0009               40        EXIT    EQU     9
                   41
                   42 ;OPEN CALL PARAMETERS
0000 1A00      D   43        OBLK:   DW      RBLK
0002 0C00      D   44                DW      OFILE
0004 0100          45                DW      1
0006 0000          46                DW      0
0008 0A00      D   47                DW      OSTAT
0002               48        OSTAT:  DS      2
000C 3A46303A      49        OFILE:  DB      ':F0:ALPHA.DIR '
0010 414C5048
0014 412E4449
0018 5220
```

```
LOC  OBJ          SEQ        SOURCE STATEMENT

                   50
                   51 ;READ CALL PARAMETERS
0002               52         RBLK:   DS      2
001C 2D00     D    53                 DW      IBUF
001E 8019          54                 DW      1980H
0020 2400     D    55                 DW      ACTUAL
0022 2600     D    56                 DW      RSTAT
0002               57         ACTUAL: DS      2
0002               58         RSTAT:  DS      2
                   59
                   60 ;EXIT CALL PARAMETERS
0028 2A00     D    61         EBLK:   DW      ESTAT
0002               62         ESTAT:  DS      2
                   63
                   64 ;SYSTEM CONFIGURATION STATUS BYTE STORAGE
0001               65         IOSTOR: DS      1
                   66
                   67 ;INPUT BUFFER FOR ALPHA.DIR
2000               68         IBUF:   DS      2000H
                   69
                   70 ;OUTPUT BUFFER VARIABLES AND ADDRESSES
                   71 ;THE OUTPUT BUFFER IS BUILT UPWARDS WITH NEW ENTRIES PUSHING THE TOP
                   72 ;OF THE BUFFER UPWARDS (TOWARDS LOWER MEMORY ADDRESSES) WHILE THE BOTTOM
                   73 ;END OF THE BUFFER REMAINS FIXED.
                   74 ;EACH ENTRY TAKES TWO BYTES OF STORAGE BECAUSE THEY ARE ADDRESSES
                   75 ;THE LOW BYTE PRECEEDS THE HIGH BYTE FOR EACH ADDRESS
                   76
202D FF            77         XX:     DB      0FFH                ;EOF DELIMETER FOR BUFFER
0190               78         BUFFER: DS      400D                ;LIST OF ADDRESSES OF ALPHABETIZED
                   79                                             ;FILE NAMES
21BE 2D20     D    80         ENDBUF: DW      XX                  ;ADDRESS OF EOF BUFFER DELIMETER
21C0 BE21     D    81         TPBUFF: DW      ENDBUF
                   82
                   83
                   84                 CSEG
                   85
                   86 ;SET STACK POINTER
                   87         START:  STKLN   20D                 ;SET THE MAXIMUM STACK LENGTH
0000 310000   S    88                 LXI     SP,STACK            ;LOAD STACK POINTER
                   89                                             ;WITH VALUE DETERMINED BY
                   90                                             ;THE LOCATE COMMAND
                   91
                   92 ;SET LIST DEVICE TO LINE PRINTER
0003 CD0000   E    93                 CALL    IOCHK               ;GET I/O DEVICE CONFIGURATION
0006 322C00   D    94                 STA     IOSTOR              ;STORE STATUS WORD
0009 E63F          95                 ANI     3FH                 ;CLEAR LIST DEVICE
000B F680          96                 ORI     80H                 ;SET LINE PRINTER TO LIST DEVICE
000D 4F            97                 MOV     C,A
000E CD0000   E    98                 CALL    IOSET               ;REPLACE STATUS BYTE
                   99
                  100 ;OPEN DIRECTORY FILE
                  101
0011 0E00         102                 MVI     C,OPEN
0013 110000   D   103                 LXI     D,OBLK
0016 CD0000   E   104                 CALL    ISIS
```

```
 LOC  OBJ          SEQ        SOURCE STATEMENT

                    105
                    106 ;READ DIRECTORY FILE
                    107
 0019 0E03          108              MVI     C,READ
 001B 111A00   D    109              LXI     D,RBLK
 001E CD0000   E    110              CALL    ISIS
                    111
                    112
                    113 ;PRINT OUT 2 LINE HEADER
 0021 0E0C          114              MVI     C,0CH          ;FORM FEED
 0023 CD0000   E    115              CALL    LO             ;PRINT FORM FEED
 0026 0E0A          116              MVI     C,0AH          ;LINE FEED
 0028 CD0000   E    117              CALL    LO             ;PRINT LINE FEED
 002B 212D00   D    118              LXI     H,IBUF         ;SET H&L TO TOP OF INPUT BUFFER
                    119
                    120 ;FIRST LINE
 002E 4E            121      PRNTHD: MOV     C,M            ;GET CHARACTER FROM INPUT BUFFER
 002F CD0000   E    122              CALL    LO             ;PRINT CHARACTER
 0032 7E            123              MOV     A,M            ;GET CHARACTER AGAIN
 0033 EE0A          124              XRI     0AH            ;IS CHARACTER A LINE FEED?
 0035 23            125              INX     H              ;POINT H&L TO NEXT CHARACTER
 0036 C22E00   C    126              JNZ     PRNTHD         ;JUMP IF CHAR NOT LINE FEED
                    127
                    128 ;SECOND LINE
 0039 4E            129      PRNTH2: MOV     C,M            ;GET CHARACTER FROM INPUT BUFFER
 003A CD0000   E    130              CALL    LO             ;PRINT CHARACTER
 003D 7E            131              MOV     A,M            ;GET CHARACTER AGAIN
 003E EE0A          132              XRI     0AH            ;IS CHARACTER A LINE FEED?
 0040 23            133              INX     H              ;POINT H&L TO NEXT CHARACTER
 0041 C23900   C    134              JNZ     PRNTH2         ;JUMP IF CHAR NOT LINE FEED
                    135
                    136
                    137 ;CHECK FOR END OF DIRECTORY
                    138
 0044 7E            139              MOV     A,M            ;GET CHAR FROM INPUT BUFFER
 0045 EE20          140              XRI     20H            ;IS CHARACTER A SPACE?
 0047 CAB400   C    141              JZ      LLINES         ;JUMP IF CHAR IS A SPACE
                    142
                    143 ;ALPHABETIZE ROUTINE
                    144
                    145
                    146 ;RESET TOP OF BUFFER
 004A 44            147      LOOK:   MOV     B,H            ;POINT B&C TO INPUT CHARACTER
 004B 4D            148              MOV     C,L
 004C 2AC021   D    149              LHLD    TPBUFF         ;POINT H&L TO TOP OF OUT BUFFER
 004F 2B            150              DCX     H              ;MOVE TOP OF OUT BUFFER
 0050 2B            151              DCX     H              ;BACK TWO SPACES
 0051 22C021   D    152              SHLD    TPBUFF         ;STORE NEW TOP OF OUT BUFFER
 0054 23            153              INX     H              ;SET H&L BACK TO OLD
 0055 23            154              INX     H              ;TOP OF BUFFER
                    155
                    156 ;NEW WORD REFERS TO FILENAME TO BE ALPHABETIZED
                    157 ;OLD WORD REFERS TO THE FILENAME ALREADY ALPHABETIZED TO WHICH THE NEW WORD
                    158 ;IS CURRENTLY BEING COMPARED
                    159 ;COMPARE FIRST LETTER OF NEW FILENAME TO ALREADY ALPHABETIZED FILENAMES
```

```
LOC  OBJ          SEQ         SOURCE STATEMENT

0056 E5           160       SEARCH: PUSH    H                   ;STORE POINTER OF ADDRESS OF
                  161                                           ;OLD WORD IN STACK
0057 5E           162               MOV     E,M                 ;POINT D&E TO FIRST CHARACTER
0058 23           163               INX     H                   ;OF FILENAME IN OUT BUFFER
0059 56           164               MOV     D,M
005A 60           165               MOV     H,B                 ;POINT H&L TO FIRST CHARACTER
                  166                                           ;OF FILENAME IN INPUT BUFFER
005B 69           167               MOV     L,C
005C 7E           168               MOV     A,M                 ;GET CHARACTER FROM IN BUFFER
005D EB           169               XCHG                        ;POINT H&L TO OLD WORD
                  170                                           ;AND POINT D&E TO NEW WORD
005E BE           171               CMP     M                   ;COMPARE NEW WORD WITH OLD WORD
005F CA7C00   C   172               JZ      EQUAL               ;FIRST LETTERS ARE SAME
0062 D28600   C   173               JNC     SORT                ;NEW WORD > OLD WORD
                  174 ;ADD ADDRESS OF NEW FILENAME TO TOP OF ALPHABETIZED LIST
                  175
0065 E1           176       FOUND:  POP     H                   ;POINT H&L TO ADDRESS OF OLD WORD
                  177                                           ;IN OUTPUT BUFFER
0066 2B           178               DCX     H                   ;SET H&L TWO SPACES BACK
0067 70           179               MOV     M,B                 ;SET ADDRESS OF NEW WORD
0068 2B           180               DCX     H                   ;IN OUTPUT BUFFER
0069 71           181               MOV     M,C
                  182
                  183
                  184 ;GET THE NEXT WORD TO ALPHABETIZE
                  185
006A 60           186       NXTWD:  MOV     H,B                 ;POINT H&L TO IN BUFFER AT START
006B 69           187               MOV     L,C                 ;OF FILENAME THAT WAS JUST ALPHABETIZED
006C 7E           188       NXTLTR: MOV     A,M                 ;GET CHARACTER FROM IN BUFFER
006D EE0A         189               XRI     0AH                 ;IS CHARACTER A LINE FEED?
006F 23           190               INX     H                   ;POINT H&L TO NEXT CHAR IN IN BUFFER
0070 C26C00   C   191               JNZ     NXTLTR              ;JUMP IF CHAR NOT LINE FEED
0073 7E           192               MOV     A,M                 ;GET CHAR FROM IN BUFFER
0074 EE20         193               XRI     20H                 ;IS CHAR A SPACE?
0076 CA9500   C   194               JZ      PTDIR               ;JUMP IF CHAR IS SPACE TO OUTPUT
                  195                                           ;THE DIRECTORY
0079 C34A00   C   196               JMP     LOOK                ;ELSE JUMP TO ALPHABETIZE THIS
                  197                                           ;NEW FILENAME
                  198
                  199 ;FIRST LETTERS ARE SAME,COMPARE NEXT LETTERS OF SAME WORDS
007C 23           200       EQUAL:  INX     H                   ;POINT H&L TO NEXT CHAR OF OLD WORD
007D 13           201               INX     D                   ;POINT D TO NEXT CHAR OF NEW WORD
007E 1A           202               LDAX    D                   ;MOVE CHARACTER OF NEW WORD
                  203                                           ;INTO ACCUMULATOR
007F BE           204               CMP     M                   ;COMPARE NEW WORD TO OLD WORD
0080 CA7C00   C   205               JZ      EQUAL               ;JUMP IF LETTERS ARE THE SAME
0083 DA6500   C   206               JC      FOUND               ;JUMP IF NEW WORD<OLD WORD
                  207
                  208
                  209 ;MOVES OLD WORD BACK IN BUFFER
0086 E1           210       SORT:   POP     H                   ;H&L POINT TO THE ADDRESS OF THE OLD WORD
0087 54           211               MOV     D,H
0088 5D           212               MOV     E,L
0089 1B           213               DCX     D                   ;D&E POINT TO THE DESTINATION
008A 1B           214               DCX     D                   ;OF THE ADDRESS OF THE OLD WORD
```

```
 LOC  OBJ          SEQ        SOURCE STATEMENT

 008B 7E           215                MOV     A,M                ;MOVE BACK LOW BYTE
 008C 12           216                STAX    D
 008D 23           217                INX     H
 008E 13           218                INX     D
 008F 7E           219                MOV     A,M                ;MOVE BACK HIGH BYTE
 0090 12           220                STAX    D
 0091 23           221                INX     H                  ;THE NEXT ENTRY IN THE
                   222                                           ;OUTPUT BUFFER NOW BECOMES
                   223                                           ;THE OLD WORD
 0092 C35600   C   224                JMP     SEARCH
                   225
                   226
                   227
                   228
                   229 ;PRINT OUT DIRECTORY
 0095 E5           230        PTDIR:  PUSH    H                  ;SET STACK TO POINT TO
                   231                                           ;CHARACTER IN INPUT BUFFER
 0096 2AC021   D   232                LHLD    TPBUFF             ;POINT H&L TO THE TOP OF THE
                   233                                           ;OUTPUT BUFFER
 0099 5E           234        PRINT:  MOV     E,M                ;POINT D&E TO THE FILENAME
 009A 23           235                INX     H
 009B 56           236                MOV     D,M
 009C 23           237                INX     H                  ;POINT H&L TO THE NEXT ENTRY
                   238                                           ;IN THE OUT BUFFER
 009D EB           239                XCHG                       ;POINT H&L TO FILENAME
                   240                                           ;POINT D&E TO NEXT ENTRY OF OUT BUFFER
 009E 7E           241                MOV     A,M                ;GET CHARACTER TO BE OUTPUT
 009F EEFF         242                XRI     0FFH               ;IS CHAR END OF BUFFER?
 00A1 CAB300   C   243                JZ      ENDING             ;JUMP IF CHAR IS END OF BUFFER
                   244                                           ;DELIMETER
 00A4 4E           245        PLTTR:  MOV     C,M                ;GET CHARACTER TO BE OUTPUT
 00A5 CD0000   E   246                CALL    LO                 ;PRINT CHARACTER
 00A8 7E           247                MOV     A,M                ;GET SAME CHARACTER
 00A9 EE0A         248                XRI     0AH                ;IS CHAR A LINE FEED?
 00AB 23           249                INX     H                  ;POINT H&L TO NEXT CHAR
 00AC C2A400   C   250                JNZ     PLTTR              ;JUMP IF CHAR IS NOT A LINE FEED
 00AF EB           251                XCHG                       ;POINT H&L TO NEXT ENTRY IN
                   252                                           ;OUT BUFFER
 00B0 C39900   C   253                JMP     PRINT
                   254
                   255 ;PRINT OUT LAST 2 LINES
 00B3 E1           256        ENDING: POP     H                  ;POINT H&L TO 2ND TO THE LAST LINE
 00B4 4E           257        LLINES: MOV     C,M                ;GET CHARACTER TO BE OUTPUT
 00B5 CD0000   E   258                CALL    LO                 ;PRINT CHARACTER
 00B8 7E           259                MOV     A,M                ;GET CHARACTER AGAIN
 00B9 EE0A         260                XRI     0AH                ;IS CHAR A LINE FEED?
 00BB 23           261                INX     H                  ;POINT H&L TO NEXT CHARACTER
 00BC C2B400   C   262                JNZ     LLINES             ;JUMP IF CHARACTER IS NOT
                   263                                           ;A LINE FEED
                   264
 00BF 4E           265        LLINE:  MOV     C,M                ;GET CHARACTER TO BE OUTPUT
 00C0 CD0000   E   266                CALL    LO                 ;PRINT CHARACTER
 00C3 7E           267                MOV     A,M                ;GET SAME CHARACTER
 00C4 EE0A         268                XRI     0AH                ;IS CHARACTER A LINE FEED?
 00C6 23           269                INX     H                  ;POINT H&L TO NEXT CHARACTER
```

```
 LOC  OBJ          SEQ         SOURCE STATEMENT

 00C7 C2BF00   C   270                 JNZ     LLINE                 ;JUMP IF CHARACTER IS NOT
                   271                                               ;A LINE FEED
                   272
                   273 ;SET LIST DEVICE BACK AND EXIT TO ISIS
                   274 ;RESET STATUS
 00CA 3A2C00   D   275         XIT:    LDA     IOSTOR                ;STORED SYSTEM I/O STATUS
 00CD 4F           276                 MOV     C,A
 00CE CD0000   E   277                 CALL    IOSET
                   278 ;EXIT BACK TO ISIS-II
 00D1 0E09         279                 MVI     C,EXIT                ;ALL OPEN FILES ARE CLOSED
 00D3 112800   D   280                 LXI     D,EBLK                ;EXCEPT :CI: AND :CO: AND
 00D6 CD0000   E   281                 CALL    ISIS                  ;CONTROL PASSES BACK TO ISIS
 0000          C   282                 END     START
```

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

```
IOCHK  E 0000    IOSET  E 0000    ISIS   E 0000    LO     E 0000
```

USER SYMBOLS

```
ACTUAL D 0024    BUFFER D 202E    EBLK   D 0028    ENDBUF D 21BE    ENDING C 00B3    EQUAL  C 007C    ESTAT  D 002A
EXIT   A 0009    FOUND  C 0065    IBUF   D 002D    IOCHK  E 0000    IOSET  E 0000    IOSTOR D 002C    ISIS   E 0000
LLINE  C 00BF    LLINES C 00B4    LO     E 0000    LOOK   C 004A    NXTLTR C 006C    NXTWD  C 006A    OBLK   D 0000
OFILE  D 000C    OPEN   A 0000    OSTAT  D 000A    PLTTR  C 00A4    PRINT  C 0099    PRNTH2 C 0039    PRNTHD C 002E
PTDIR  C 0095    RBLK   D 001A    READ   A 0003    RSTAT  D 0026    SEARCH C 0056    SORT   C 0086    START  C 0000
TPBUFF D 21C0    XIT    C 00CA    XX     D 202D
```

ASSEMBLY COMPLETE, NO ERRORS

Ref.#AB133

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 ☒8080 ☐8048 ☐8085 ☒Other UPP102 (use additional sheets if necessary)

Program Title

BLOCK: LARGE HEX FILE INTO PROM LENGTH BLOCKS CONVERTER

Function

This program is used to prevent a "system error" which could be caused by a hex file larger than the usable memory slot available while operating the UPM. This program facilitates programming of the 2708 I Prom by breaking up a large hex file into prom length blocks.

Required Hardware

MDS 800 with 32K Ram, MDS 2DS, Console Device, UPP 102

Required Software

32K ISIS-II

Input Parameters

This program should be located at absolute memory address of 3A00H. The Dseg portion should be located at 3B00H. I placed this program on the system floppy. The program to be worked on should be a hex file and can be in either drive 0 or 1. To invoke this program type "Block (:F1:) hex File."

Output Results

The output results will be newly created files bearing the name of the hex file, with the extension of numerical sequence starting at 00. The extension is derived from the first address of each block.

After the hex file is blocked, then the UPM program my be called, and each block file may be read in in two's. After programming two proms, then use the offset function to offset the starting address of the next two blocks.

Registers Modified:	Programmer: Steven L. Mulkey
RAM Required:	Company: Zenith Electronics Corp. of MO.
ROM Required:	Address: 2500 East Kearney
Maximum Subroutine Nesting Level:	City: Springfield
Assembler/Compiler Used: Intellec MDS Macro Assembler	State: Missouri 65803

```
ASM80 :F1:BLCK.SRC

ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0        MODULE   PAGE    1

 LOC  OBJ          SEQ          SOURCE STATEMENT

                     1          ;BLOCK PROGRAM WRITTEN BY
                     2          ;STEVEN MULKEY
                     3          ;ZENITH RADIO CORP OF MO
                     4          ;2500 EAST KEARNEY
                     5          ;SPRINGFIELD, MO
 FE0E                6          CONV    EQU     0FE0EH
 0000                7          OPEN    EQU     0
 0001                8          CLOSE   EQU     1
 0003                9          READ    EQU     3
 0004               10          WRITE   EQU     4
 0009               11          EXIT    EQU     9
 000C               12          ERROR   EQU     12
                    13          EXTRN   ISIS
                    14
                    15          CSEG
 0000 310800    S   16 BEGIN:   LXI     SP,STACK+8      ;SET STACK
 0003 3EFC          17          MVI     A,0FCH
 0005 32A20B    D   18          STA     PROM            ;STORE FILE EXTENSION NUMBER
 0008 0E03          19          MVI     C,READ
 000A 111600    D   20          LXI     D,RBLK
 000D CD0000    E   21          CALL    ISIS            ;READ KEYBOARD FOR FILE TO BE BLOCKED
 0010 3A2A00    D   22          LDA     STATUS
 0013 B7            23          ORA     A
 0014 C24900    C   24          JNZ     ERR             ;CHECK FOR ERROR
 0017 CDCA00    C   25          CALL    STORE           ;STORE FILE NAME OF FILE TO BE OPENED
 001A 0E00          26          MVI     C,OPEN
 001C 112000    D   27          LXI     D,PBLK
 001F CD0000    E   28          CALL    ISIS            ;OPEN HEX FILE
 0022 3A2A00    D   29          LDA     STATUS
 0025 B7            30          ORA     A
 0026 C24900    C   31          JNZ     ERR             ;CHECK FOR ERROR
 0029 CD5900    C   32 AA:      CALL    REED            ;READ 2880H BYTES FROM HEX FILE
 002C 2A0E00    D   33          LHLD    ACTUAL          ;LOAD HL WITH ACTUAL BYTES READ
 002F 7C            34          MOV     A,H
 0030 B5            35          ORA     L
 0031 CA5100    C   36          JZ      EXI             ;IF H=L THEN HEX FILE HAS BEEN READ
 0034 CD6D00    C   37          CALL    LOAD            ;LOAD STORED FILE NAME AND EXTENSION
 0037 CD9A00    C   38          CALL    OPN             ;OPEN NEW FILE
 003A 2A0A00    D   39          LHLD    OAFT
 003D 221200    D   40          SHLD    CBLK            ;STORE NEW FILE AFTN IN CLOSE BLOCK
 0040 CDAA00    C   41          CALL    WRIT            ;WRITE 2880H BYTE IN NEW FILE
 0043 CDBA00    C   42          CALL    CLOS            ;CLOSE NEW FILE
 0046 C32900    C   43          JMP     AA
 0049 0E0C          44 ERR:     MVI     C,ERROR
 004B 112A00    D   45          LXI     D,EBLK
 004E CD0000    E   46          CALL    ISIS            ;PRINT WHAT TYPE OF ERROR
 0051 0E09          47 EXI:     MVI     C,EXIT
 0053 112E00    D   48          LXI     D,XBLK
 0056 CD0000    E   49          CALL    ISIS            ;EXIT BACK TO ISIS
                    50
                    51          ;SUBROUTINES
                    52
```

```
LOC  OBJ          SEQ         SOURCE STATEMENT

0059 0E03          53 REED:   MVI     C,READ
005B 111600   D    54         LXI     D,RBLK
005E CD0000   E    55         CALL    ISIS          ;READ HEX FILE OR KEYBOARD
0061 3A2A00   D    56         LDA     STATUS
0064 B7            57         ORA     A
0065 C26900   C    58         JNZ     RERR          ;JUMP IF ERROR OCCURED
0068 C9            59         RET
                   60
0069 D1            61 RERR:   POP     D             ;POP RETURN ADDRESS OFF OF STACK
006A C34900   C    62         JMP     ERR           ;AND JUMP TO ERROR ROUTINE
                   63
006D 21890B   D    64 LOAD:   LXI     H,OPA         ;FILE NAME STORAGE
0070 11700B   D    65         LXI     D,OFILE       ;BUFFER FOR NEW FILE TO BE OPENED
0073 7E            66 CCC:    MOV     A,M
0074 12            67         STAX    D
0075 23            68         INX     H
0076 13            69         INX     D
0077 FE2E          70         CPI     '.'           ;TRANSFER FROM OPA TO OFILE
                   71                               ;UNTIL '.'
0079 C27300   C    72         JNZ     CCC
007C 3AA20B   D    73         LDA     PROM          ;LOAD A WITH STORED EXTENSION
007F C604          74         ADI     04            ;ADD 4 TO A
0081 32A20B   D    75         STA     PROM          ;STORE RESULTS
0084 F5            76         PUSH    PSW           ;SAVE A
0085 1F            77         RAR
0086 1F            78         RAR
0087 1F            79         RAR
0088 1F            80         RAR                   ;ROTATE A RIGHT 4 TIMES
0089 CD0EFE        81         CALL    CONV          ;CONVER A RIGHT 4 BITS TO ASCII
008C 79            82         MOV     A,C
008D 12            83         STAX    D             ;STORE IN OFILE
008E 13            84         INX     D
008F F1            85         POP     PSW           ;RETRIEVE A
0090 CD0EFE        86         CALL    CONV          ;CONVERT A RIGHT 4 BITS TO ASCII
0093 79            87         MOV     A,C
0094 12            88         STAX    D             ;STORE IN OFILE
0095 13            89         INX     D
0096 3E20          90         MVI     A,' '
0098 12            91         STAX    D             ;STORE TERMINATING CHAR
0099 C9            92         RET
                   93
009A 0E00          94 OPN:    MVI     C,OPEN
009C 110000   D    95         LXI     D,OBLK
009F CD0000   E    96         CALL    ISIS          ;OPEN FILE
00A2 3A2A00   D    97         LDA     STATUS
00A5 B7            98         ORA     A
00A6 C26900   C    99         JNZ     RERR          ;JUMP IF ERROR OCCURED
00A9 C9           100         RET
                  101
00AA 0E04         102 WRIT:   MVI     C,WRITE
00AC 110A00   D   103         LXI     D,WBLK
00AF CD0000   E   104         CALL    ISIS          ;WRITE INTO NEW FILE
00B2 3A2A00   D   105         LDA     STATUS
00B5 B7           106         ORA     A
00B6 C26900   C   107         JNZ     RERR          ;JUMP IF ERROR OCCURED
```

```
LOC  OBJ          SEQ         SOURCE STATEMENT

00B9 C9           108         RET
                  109
00BA 0E01         110 CLOS:   MVI     C,CLOSE
00BC 111200   D   111         LXI     D,CBLK
00BF CD0000   E   112         CALL    ISIS            ;CLOSE FILE
00C2 3A2A00   D   113         LDA     STATUS
00C5 B7           114         ORA     A
00C6 C26900   C   115         JNZ     RERR            ;JUMP IF ERROR OCCURED
00C9 C9           116         RET
                  117
00CA 213000   D   118 STORE:  LXI     H,BUFFER        ;POINT HL TO INPUT FROM KEYBO.
00CD 11890B   D   119         LXI     D,OPA           ;SET D AT PERM. FILE NAME STO.
00D0 23           120         INX     H
00D1 7E           121 BB:     MOV     A,M             ;TRANSFERS FROM M INTO A
00D2 FE2E         122         CPI     '.'             ;AND CHECK FOR '.'
00D4 CAE200   C   123         JZ      BBB
00D7 FE20         124         CPI     ' '             ;AND SPACE, IF NEITHER OCCURED
00D9 CAE200   C   125         JZ      BBB
00DC 12           126         STAX    D               ;STO CHAR AND JMP TO BB
00DD 23           127         INX     H
00DE 13           128         INX     D
00DF C3D100   C   129         JMP     BB
00E2 3E2E         130 BBB:    MVI     A,'.'           ;STORE '.' IN PERM STOR
00E4 12           131         STAX    D
00E5 C9           132         RET
                  133
                  134         DSEG
0000 0A00     D   135 OBLK:   DW      OAFT            ;POINTER FOR OPEN AFTN
0002 700B     D   136         DW      OFILE           ;POINTER FOR FILE TO BE OPENED
0004 0200         137         DW      2               ;OPEN FILE FOR WRITE
0006 0000         138         DW      0               ;NO ECHO
0008 2A00     D   139         DW      STATUS          ;POINTER FOR STATUS
                  140 OAFT:
0002              141 WBLK:   DS      2               ;STORAGE FOR OPENED FILE AFTN
000C 3000     D   142         DW      BUFFER          ;POINTER FOR STORAGE AREA
0002              143 ACTUAL: DS      2               ;STORAGE FOR THE NO. OF BYTE TO BE WRITTEN
0010 2A00     D   144         DW      STATUS          ;POINTER FOR STATUS
0002              145 CBLK:   DS      2               ;AFTN OF FILE TO BE CLOSED
0014 2A00     D   146         DW      STATUS          ;POINTER FOR STATUS
                  147 RAFT:
0016 0100         148 RBLK:   DW      1               ;READ KEYBOARD
0018 3000     D   149         DW      BUFFER          ;POINTER TO STORAGE AREA
001A 400B         150         DW      2880            ;BYTES TO BE READ
001C 0E00     D   151         DW      ACTUAL          ;HOW MANY WERE ACTUALLY READ
001E 2A00     D   152         DW      STATUS          ;POINTER FOR STATUS
0020 1600     D   153 PBLK:   DW      RAFT            ;AFTN OF HEX FILE
0022 3000     D   154         DW      BUFFER          ;POINTER TO STORAGE AREA
0024 0100         155         DW      1               ;OPEN FOR READ OPERATION
0026 0000         156         DW      0               ;NO ECHO
0028 2A00     D   157         DW      STATUS          ;POINTER TO STATUS
                  158 EBLK:
0002              159 STATUS: DS      2               ;STORAGE FOR STATUS
002C 2A00     D   160         DW      STATUS          ;POINTER FOR STATUS
002E 2A00     D   161 XBLK:   DW      STATUS          ;POINTER FOR STATUS
0B40              162 BUFFER: DS      2880            ;DEFINED STORAGE OF 2880H BYTES
```

```
 LOC  OBJ        SEQ        SOURCE STATEMENT

 0019             163 OFILE: DS     25            ;DEFINED STORAGE OF 25H BYTES
 0019             164 OPA:   DS     25            ;DEFINED STORAGE OF 25H BYTES
 0002             165 PROM:  DS     2             ;EXTENSION STORAGE
 0000         C   166        END    BEGIN

PUBLIC SYMBOLS


EXTERNAL SYMBOLS
ISIS   E 0000

USER SYMBOLS
AA     C 0029   ACTUAL D 000E   BB     C 00D1   BBB    C 00E2   BEGIN  C 0000   BUFFER D 0030   CBLK   D 0012
CCC    C 0073   CLOS   C 00BA   CLOSE  A 0001   CONV   A FE0E   EBLK   D 002A   ERR    C 0049   ERROR  A 000C
EXI    C 0051   EXIT   A 0009   ISIS   E 0000   LOAD   C 006D   OAFT   D 000A   OBLK   D 0000   OFILE  D 0B70
OPA    D 0B89   OPEN   A 0000   OPN    C 009A   PBLK   D 0020   PROM   D 0BA2   RAFT   D 0016   RBLK   D 0016
READ   A 0003   REED   C 0059   RERR   C 0069   STATUS D 002A   STORE  C 00CA   WBLK   D 000A   WRIT   C 00AA
WRITE  A 0004   XBLK   D 002E

ASSEMBLY COMPLETE,  NO ERRORS
```

Ref.#AB134

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 ☒8080 ☐8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

PROM BUS MAPPER

Function

To Relocate Hex data Bits, prior to programming a PROM which is to be used in a system where the address and data lines between μP and PROM are not directly related.

Required Hardware

MDS 800 System

Required Software

Program + console input and console output routines

Input Parameters

Hex data at address 1ØØØH upwards
Address and data interconnect information input on console

Output Results

Relocated Hex data at address 2ØØØH upwards

Registers Modified: All	Programmer: R. A. Stevenson
RAM Required: 16K	Company: G.E.C. Telecoms. Ltd.
ROM Required:	Address: Wiminbank Road
Maximum Subroutine Nesting Level: 2	City: Aycliffe Ind. Estate
Assembler/Compiler Used: 8080 Macro Ass. V.2.2	State: Durham England

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/4040 ☐ 8008 ☒ 8080 ☐ 8048 ☒ 8085 ☐ Other _____ (use additional sheets if necessary)

Program Title

FLEXIBLE NAME LIST MANAGER

Function

To store and retrieve variable length names in a symbol table, together with associated attributes.

Required Hardware

8080/8085 + memory

Required Software

None

Input Parameters

HL = (BLOCK)

```
BLOCK:  DB  INDEX
        DB  ATTRIBUTE
        DB  NO, CHARS
        DB  'STRING'
```

Output Results

```
INDEX & ATTRIBUTE  given  'STRING' & NO. CHARS  name present
INDEX              given  'STRING' & NO. CHARS  name not present
'STRING' & ATTRIBUTE  given INDEX               name present
*NAME NOT FOUND*  ERROR
*TABLE FULL*      ERROR
```

Registers Modified: A & PSW	Programmer: M. G. Dineley
RAM Required: Dependant on No. of names & size	Company: Dept. E. E. & E. UMIST
ROM Required: 285_{10}	Address: Sackville St.
Maximum Subroutine Nesting Level: 2	City: Manchester England
Assembler/Compiler Used: ISIS-II Macro Assembler	State:

Ref.#AB136

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/4040 ☐ 8008 ☒ 8080 ☐ 8048 ☐ 8085 ☐ Other _____ (use additional sheets if necessary)

Program Title

IDENT1 - FRONT PAGE IDENTIFIER PRINT PROGRAM

Function

The program prompts the operator to furnish the date, disk identification, filename and programmer's name. The program then composes and prints an identification front page as a cover page for the program listing which follows it.

Required Hardware

INTELLEC MDS with floppy disk (dual) drive and line printer.

Required Software

ISIS-II

Input Parameters

The program does not interact with other programs.

Output Results

Registers Modified: All	Programmer: Phil Greenberg
RAM Required: 3950 = 0F7EH	Company: Conrac Corp. - Plant 3
ROM Required: None	Address: 32 Fairfield Place
Maximum Subroutine Nesting Level: N/A	City: West Caldwell
Assembler/Compiler Used: INTELLEC MDS Macro Assembler	State. New Jersey 07006

Ref.#AE17

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 ☒8080 ☐8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

CERROR - PL/M-80 COMPILER ERROR DISPLAY PROGRAM

Function

Program displays PL/M-80 Compiler Errors
Assembled program must be linked with System.LIB and located above ISIS-II. Program resides on system diskette.

Required Hardware

MDS 800, DOS-2DS, TTY or CRT

Required Software

ISIS-II

Input Parameters

Command CERROR :FX:Name.LST

Output Results

All PL/M Compiler errors are copied on :CO:

Registers Modified: All registers	Programmer: Prof. Ing. Dalibor Nemec
RAM Required: 13B H	Company: VSE- Dept. of Technology
ROM Required: None	Address: Pelhrimovska 9, Praha 4,
Maximum Subroutine Nesting Level: 3	City: Czechoslov
Assembler/Compiler Used: 8080/8085 Macro Assembler	State:

```
ASM80 :F1:CERROR.SRC

ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0        CERROR   PAGE    1


 LOC  OBJ         SEQ        SOURCE STATEMENT

                    1 ;
                    2 $       TITLE('PL/M-80 COMPILER ERRORS')
                    3         NAME CERROR
                    4 ;
 0000               5         OPEN EQU 0
 0003               6         READ EQU 3
 0009               7         EXIT EQU 9
 000C               8         ERROR EQU 12
 000A               9         LF EQU 0AH
 000D              10         CR EQU 0DH
 F809              11         CO EQU 0F809H   ; MONITOR ROUTINE ADDRESS
                   12 ;
                   13         EXTRN ISIS
                   14 ;
                   15         CSEG
 0000 310000    S  16 BEGIN:  LXI SP,STACK    ; ADJUST STACK POINTER
 0003 0E03         17         MVI C,READ      ; READ NAME OF LIST
 0005 110000    D  18         LXI D,RBLK      ; FILE FROM :CI:
 0008 CD4900    C  19         CALL SYSTEM     ; INTO BUFFER
 000B 0E00         20         MVI C,OPEN      ; OPEN LIST FILE
 000D 110A00    D  21         LXI D,OBLK      ; FOR READING
 0010 CD4900    C  22         CALL SYSTEM
 0013 CD5F00    C  23         CALL READBL     ; READ BLOK INTO BUFFER
 0016 CD4200    C  24 LOOP:   CALL CHAR       ; PICK UP CHAR. FROM BUFFER
 0019 79           25         MOV A,C
 001A FE0A         26         CPI LF          ; <LF> LAST CHAR. ON LINE,SEQUENCE <CR><LF>
 001C C21600    C  27         JNZ LOOP        ; LOOP UNTIL LAST CHAR.
 001F CD4200    C  28 LOOP1:  CALL CHAR       ; PICK UP NEXT CHAR.
 0022 79           29         MOV A,C
 0023 FE2A         30         CPI '*'         ; BEGINNING OF COMPILER ERROR MESSAGE
 0025 CA3000    C  31         JZ DISPCH       ; DISPLAY ERR. MESSAGE
 0028 FE0D         32         CPI CR          ; LINE ENDS <CR>,SEQUENCE <LF><CR>
 002A C21600    C  33         JNZ LOOP        ; LOOP UNTIL LAST CHAR.
 002D C31F00    C  34         JMP LOOP1       ; TEST NEXT LINE
 0030 CD09F8       35 DISPCH: CALL CO         ; DISPLAY FIRST CHAR. OF MESSAGE
 0033 CD4200    C  36         CALL CHAR       ; PICK UP SECOND CHAR.
 0036 79           37         MOV A,C
 0037 FE0A         38         CPI LF          ; TEST IF END OF LINE
 0039 C23000    C  39         JNZ DISPCH      ; LOOP UNTIL END OF COMPILER MESSAGE
 003C CD09F8       40         CALL CO         ; DISPLAY LAST CHAR ON LINE
 003F C31F00    C  41         JMP LOOP1       ; TEST NEW LINE
                   42 ;
 0042 23           43 CHAR:   INX H           ; ADDRESS NEXT CHAR
 0043 4E           44         MOV C,M         ; PICK UP CHAR FROM BUFFER
 0044 05           45         DCR B           ; IF BUFFER EMPTY READ
 0045 CC5F00    C  46         CZ READBL       ; NEW BLOCK
 0048 C9           47         RET
                   48 ;
 0049 CD0000    E  49 SYSTEM: CALL ISIS
 004C 3A2000    D  50         LDA STATUS      ; TEST ERROR STATUS
 004F B7           51         ORA A
 0050 C27600    C  52         JNZ ERR         ; BRANCH TO ERROR ROUTINE
```

PL/M-80 COMPILER ERRORS

```
LOC  OBJ          SEQ         SOURCE STATEMENT

0053 C9            53          RET
                   54 ;
0054 0E0D          55 NLINE:   MVI C,CR        ; INSERT <CR> INTO DISPLAY BUFF
0056 CD09F8        56          CALL CO         ; DISPLAY CHAR
0059 0E0A          57          MVI C,LF        ; INSERT <LF> INTO DISPLAY BUFF
005B CD09F8        58          CALL CO         ; DISPLAY CHAR
005E C9            59          RET
                   60 ;
005F C5            61 READBL:  PUSH B          ; SAVE LAST CHAR FROM LAST BLOCK IN C REG
0060 0E03          62          MVI C,READ      ; READ NEW BLOCK FROM
0062 111400    D   63          LXI D,RBLK1     ; LIST FILE INTO BUFFER
0065 CD4900    C   64          CALL SYSTEM
0068 2A9C00    D   65          LHLD ACTUAL     ; IF ALL BLOCKS OF LIST
006B 7C            66          MOV A,H         ; FILE ALREADY READ
006C B5            67          ORA L           ; THEN ACTUAL=0
006D CA7E00    C   68          JZ DONE
0070 C1            69          POP B           ; RESTORE C REG
0071 47            70          MOV B,A         ; NUMBER OF CHARS READ INTO BUFFER
0072 212100    D   71          LXI H,BUFFER-1  ; BUFFER ADDERSSING
0075 C9            72          RET
                   73 ;
0076 0E0C          74 ERR:     MVI C,ERROR     ; DISPLAY SYSTEM ERROR NUMBER
0078 111E00    D   75          LXI D,EBLK
007B CD0000    E   76          CALL ISIS
                   77 ;
007E CD5400    C   78 DONE:    CALL NLINE      ; SEPARATE WITH ONE LINE
0081 0E09          79          MVI C,EXIT      ; CLOSE LIST FILE
0083 111E00    D   80          LXI D,DBLK      ; AND RETURN TO SUPERVISOR
0086 CD0000    E   81          CALL ISIS
                   82 ;
                   83          DSEG
0000 0100          84 RBLK:    DW 1            ; READ CONSOLE
0002 2200      D   85          DW BUFFER
0004 7A00          86          DW 122
0006 9C00      D   87          DW ACTUAL
0008 2000      D   88          DW STATUS
                   89 ;
000A 1400      D   90 OBLK:    DW AFTN
000C 2200      D   91          DW BUFFER       ; FILE TO BE OPEN
000E 0100          92          DW 1            ; FOR READING
0010 0000          93          DW 0
0012 2000      D   94          DW STATUS
                   95 ;
                   96 RBLK1:
0002               97 AFTN:    DS 2            ; READ BLOCK FROM FILE
0016 2200      D   98          DW BUFFER
0018 7A00          99          DW 122
001A 9C00      D  100          DW ACTUAL
001C 2000      D  101          DW STATUS
                  102 ;
                  103 DBLK:
001E 2000      D  104 EBLK:    DW STATUS
0002              105 STATUS:  DS 2
007A              106 BUFFER:  DS 122          ; LENGTH OF BUFFER
0002              107 ACTUAL:  DS 2
```

```
 LOC  OBJ        SEQ        SOURCE STATEMENT

                  108 STKLN  10            ; DEPTH OF STACK
                  109 ;
 0000         C   110        END BEGIN

PUBLIC SYMBOLS


EXTERNAL SYMBOLS
ISIS   E 0000

USER SYMBOLS
ACTUAL D 009C   AFTN   D 0014   BEGIN  C 0000   BUFFER D 0022   CHAR   C 0042   CO     A F809   CR     A 000D
DBLK   D 001E   DISPCH C 0030   DONE   C 007E   EBLK   D 001E   ERR    C 0076   ERROR  A 000C   EXIT   A 0009
ISIS   E 0000   LF     A 000A   LOOP   C 0016   LOOP1  C 001F   NLINE  C 0054   OBLK   D 000A   OPEN   A 0000
RBLK   D 0000   RBLK1  D 0014   READ   A 0003   READBL C 005F   STATUS D 0020   SYSTEM C 0049

ASSEMBLY COMPLETE,  NO ERRORS
```

Ref.#AB136

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 ☒8080 ☐8048 ☐8085 ☐Other_____ (use additional sheets if necessary)

Program Title

IDENT1 - FRONT PAGE IDENTIFIER PRINT PROGRAM

Function

The program prompts the operator to furnish the date, disk identification, filename and programmer's name. The program then composes and prints an identification front page as a cover page for the program listing which follows it.

Required Hardware

INTELLEC MDS with floppy disk (dual) drive and line printer.

Required Software

ISIS-II

Input Parameters

The program does not interact with other programs.

Output Results

Registers Modified: ALL	Programmer: Phil Greenberg
RAM Required: 3950 = 0F7EH	Company: Conrac Corp. - Plant 3
ROM Required: None	Address: 32 Fairfield Place
Maximum Subroutine Nesting Level: N/A	City: West Caldwell
Assembler/Compiler Used: INTELLEC MDS Macro Assembler	State: New Jersey 07006

Ref.#AB137

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☒ Other iSBC 80/30 (use additional sheets if necessary)

Program Title

MON830 - Monitor for iSBC 80/30

Function

2K Debug monitor for iSBC 80/30. Provides simple memory and register display and modification commands as well as program execution with breakpoints. The monitor also provides the user with routines for performing console I/O and paper tape I/O.

Required Hardware

iSBC 80/30 plus console device

Required Software

None

Input Parameters

None. At system power up, monitor will wait for a string of ASCII 'u's to be typed on the console device, to determine the baud rate of the terminal. After this, a banner is output and command mode is entered. Commands are implemented for memory/register display/modification, execution with and without single stepping and breakpoints, and paper tape program load/save.

Output Results

Available on non-system diskette only for $35.00 (source & object code included)

Registers Modified: All	Programmer:
RAM Required: 96	Company:
ROM Required: 2040	Address:
Maximum Subroutine Nesting Level: 8	City:
Assembler/Compiler Used: MDS Macro Assembler	State:

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐4004/4040 ☐8008 ☐8080 ☒8048 ☐8085 ☒Other UPI-41 (use additional sheets if necessary)

Program Title

UPI-41 8-DIGIT LED DISPLAY CONTROLLER

Function

This program uses the UPI-41 as a LED Display Controller which scans and refreshes eight multiplexed seven-segment LED Displays. The characters are defined by input from the master microprocessor in the form of an eight bit word per digit-character selection. A total of 32 different alpha-numeric characters are available for display. Applications; clock or temperature readout, various message displays.

Required Hardware

All information pertaining to required hardware, software, input parameters and output results are fully documented in the Intel application note AP 41 ("INTRODUCTION TO THE UPI-41A") and the listing for this program.

Required Software

Input Parameters

Output Results

Registers Modified: A,RB1 RØ, R2,R3,R7	Programmer: Robin J. Jigour
RAM Required: (within UPI-41) EH(see AP 41)	Company: Intel
ROM Required: (within UPI-41) 73H(see AP 41)	Address:
Maximum Subroutine Nesting Level: 1	City:
Assembler/Compiler Used: ASM48 MOD41	State:

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, X107                    PAGE    1
AUGUST 30 1978

 LOC  OBJ        SEQ        SOURCE STATEMENT

                   1 ;       ***********************************************
                   2 ;       *    UPI-41 8-DIGIT LED DISPLAY CONTROLLER    *
                   3 ;       ***********************************************
                   4 ;
                   5 ;
                   6 ;
                   7 ;THIS PROGRAM USES THE UPI-41 AS A LED DISPLAY CONTROLLER
                   8 ;WHICH SCANS AND REFRESHES EIGHT SEVEN-SEGMENT LED DISPLAYS.
                   9 ;THE CHARACTERS ARE DEFINED BY INPUT FROM A MASTER CPU IN THE
                  10 ;FORM OF ONE EIGHT BIT WORD PER DIGIT-CHARACTER SELECTION.
                  11 ;
                  12 ;
                  13 ;
                  14 ;*************************************************************************
                  15 ;
                  16 ;REGISTER DEFINITIONS:
                  17 ;       REGISTER              RB1                     RB0
                  18 ;       --------              ---                     ---
                  19 ;       R0              DISPLAY MAP POINTER           NOT USED
                  20 ;       R1              NOT USED                      NOT USED
                  21 ;       R2              DATA WORD AND CHARACTER STORAGE NOT USED
                  22 ;       R3              DIGIT COUNTER                 NOT USED
                  23 ;       R4              NOT USED                      NOT USED
                  24 ;       R5              NOT USED                      NOT USED
                  25 ;       R6              NOT USED                      NOT USED
                  26 ;       R7              ACCUMULATOR STORAGE           NOT USED
                  27 ;*************************************************************************
                  28 ;
                  29 ;PORT PIN DEFINITIONS:
                  30 ;       PIN             PORT 1 FUNCTION         PORT 2 FUNCTION
                  31 ;       ---             ---------------         ---------------
                  32 ;       P0-7            SEGMENT DRIVER CONTROL  DIGIT DRIVER CONTROL
                  33 ;
                  34 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                    35 ;*************************************************************************
                    36 ;DISPLAY DATA WORD BIT DEFINITION:
                    37 ;       BIT             FUNCTION
                    38 ;       ---             --------
                    39 ;       0-4             CHARACTER SELECT
                    40 ;       5-7             DIGIT SELECT
                    41 ;
                    42 ;CHARACTER SELECT:
                    43 ;               D4  D3  D2  D1  D0  CHARACTER
                    44 ;               0   0   0   0   0      0
                    45 ;               0   0   0   0   1      1
                    46 ;               0   0   0   1   0      2
                    47 ;               0   0   0   1   1      3
                    48 ;               0   0   1   0   0      4
                    49 ;               0   0   1   0   1      5
                    50 ;               0   0   1   1   0      6
                    51 ;               0   0   1   1   1      7
                    52 ;               0   1   0   0   0      8
                    53 ;               0   1   0   0   1      9
                    54 ;               0   1   0   1   0      A
                    55 ;               0   1   0   1   1      B
                    56 ;               0   1   1   0   0      C
                    57 ;               0   1   1   0   1      D
                    58 ;               0   1   1   1   0      E
                    59 ;               0   1   1   1   1      F
                    60 ;               1   0   0   0   0
                    61 ;               1   0   0   0   1      G
                    62 ;               1   0   0   1   0      H
                    63 ;               1   0   0   1   1      I
                    64 ;               1   0   1   0   0      J
                    65 ;               1   0   1   0   1      L
                    66 ;               1   0   1   1   0      N
                    67 ;               1   0   1   1   1      O
                    68 ;               1   1   0   0   0      P
                    69 ;               1   1   0   0   1      R
                    70 ;               1   1   0   1   0      T
                    71 ;               1   1   0   1   1      U
                    72 ;               1   1   1   0   0      Y
                    73 ;               1   1   1   0   1      -
                    74 ;               1   1   1   1   0      /
                    75 ;               1   1   1   1   1      "BLANK"
                    76
                    77 ;
                    78 ;DIGIT SELECT:
                    79 ;               D7  D6  D5   DIGIT NUMBER
                    80 ;               0   0   0        1
                    81 ;               0   0   1        2
                    82 ;               0   1   0        3
                    83 ;               0   1   1        4
                    84 ;               1   0   0        5
                    85 ;               1   0   1        6
                    86 ;               1   1   0        7
                    87 ;               1   1   1        8
                    88 ;*************************************************************************
                    89 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                  90 ;*******************************************************************************
                  91 ;                              EQUATES
                  92 ;THE FOLLOWING CODE DESIGNATES "TIME" AS A VARIABLE. THIS
                  93 ;ADJUSTS THE AMOUNT OF CYCLES THE TIMER COUNTS BEFORE
                  94 ;A TIMER INTERRUPT OCCURS AND REFRESHES THE DISPLAY. APPROXIMATELY
                  95 ;50 TIMES PER SECOND.
FFF1              96 TIME    EQU     -0FH    ;TIMER VALUE 2.5MSEC
                  97 ;*******************************************************************************
                  98 ;                          INTERRUPT BRANCHING
                  99 ;THIS PORTION OF MEMORY IS DEDICATED FOR USE OF RESET AND
                 100 ;INTERRUPT BRANCHING. WHEN THE INTERRUPTS ARE ENABLED THE
                 101 ;CODE AT THE FOLLOWING DESIGNATED SPOTS ARE EXECUTED WHEN A
                 102 ;RESET OR A INTERRUPT OCCURS.
0000             103        ORG     0               ;
0000 0409        104        JMP     START           ;RESET
0002 00          105        NOP                     ;
0003 0438        106        JMP     INPUT           ;IBF INTERRUPT
0005 00          107        NOP                     ;
0006 00          108        NOP                     ;
0007 041F        109        JMP     DISPLA          ;TIMER INTERRUPT
                 110 ;*******************************************************************************
                 111 ;                          INITIALIZATION
                 112 ;THE FOLLOWING CODE SETS UP THE UPI-41 AND DISPLAY HARDWARE
                 113 ;INTO OPERATIONAL FORMAT.  THE DISPLAY IS TURNED OFF, THE DISPLAY
                 114 ;MAP IS FILLED WITH "BLANK" CHARACTERS, THE TIMER SET AND THE
                 115 ;INTERRUPTS ARE ENABLED.
                 116 ;
0009 D5          117 START:  SEL     RB1             ;
000A 8A08        118        ORL     P2,#08H         ;TURN DIGIT DRIVERS OFF
000C B838        119        MOV     R0,#38H         ;DISPLAY MAP POINTER,BOTTOM OF DISPLAY MAP
000E 23FF        120 BLKMAP: MOV     A,#0FFH         ;FF="BLANK"
0010 A0          121        MOV     @R0,A           ;BLANK TO DISPLAY MAP
0011 18          122        INC     R0              ;INCREMENT DISPLAY MAP POINTER
0012 F8          123        MOV     A,R0            ;DISPLAY MAP POINTER TO ACCUMULATOR
0013 B20E        124        JB5     BLKMAP          ;BLANK DISPLAY MAP TILL FILLED
0015 BB00        125        MOV     R3,#00H         ;SET DIGIT COUNTER TO 0
0017 23F1        126        MOV     A,#TIME         ;TIMER VALUE
0019 62          127        MOV     T,A             ;LOAD TIMER
001A 55          128        STRT    T               ;START TIMER
001B 25          129        EN      TCNTI           ;ENABLE TIMER INTERRUPT
001C 05          130        EN      I               ;ENABLE IBF INTERRUPT
                 131 ;*******************************************************************************
                 132 ;                          USER PROGRAM
                 133 ;A USERS PROGRAM WOULD INITIALIZE AT THIS POINT.  THE FOLLOWING
                 134 ;CODE IS USED TO TAKE THE PLACE OF A POSSIBLE USER PROGRAM.
                 135 ;
                 136 ;
001D 041D        137 LOOP    JMP     LOOP            ;WAIT FOR INTERRUPT
                 138 ;*******************************************************************************
                 139 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                140 ;*************************************************************************
                141 ;                          DISPLAY ROUTINE
                142 ; THIS PORTION OF THIS PROGRAM IS AN INTERRUPT ROUTINE WHICH IS
                143 ;ACTED UPON WHEN THE TIMER COUNT IS COMPLETED.  THE ROUTINE UPDATES
                144 ;ONE DISPLAY DIGIT FROM THE DISPLAY MAP PER INTERRUPT SEQUENTIALLY,
                145 ;THUS EIGHT TIMER INTERRUPTS WILL HAVE REFRESHED THE ENTIRE DISPLAY.
                146 ;REGISTER BANK 1 IS SELECTED AND THE ACCUMULATOR IS SAVED UPON
                147 ;ENTERING THE ROUTINE.  ONCE THE DISPLAY HAS BEEN REFRESHED THE TIMER
                148 ;IS RESET AND THE ACCUMULATOR AND PRE-INTERRUPT REGISTER BANK IS RESTORED.
                149 ;
001F D5         150 DISPLA: SEL     RB1             ;REGISTER BANK 1
0020 AF         151         MOV     R7,A            ;SAVE ACCUMULATOR
0021 8A08       152         ORL     P2,#08H         ;TURN DIGIT DRIVERS OFF
0023 FB         153         MOV     A,R3            ;DIGIT COUNTER TO ACCUMULATOR
0024 4338       154         ORL     A,#38H          ;"OR" TO GET DISPLAY MAP ADDRESS
0026 A8         155         MOV     R0,A            ;DISPLAY MAP POINTER
0027 F0         156         MOV     A,@R0           ;GET CHARACTER FROM DISPLAY MAP
0028 39         157         OUTL    P1,A            ;OUTPUT CHARACTER TO SEGMENT DRIVERS
0029 FB         158         MOV     A,R3            ;DIGIT COUNTER VALUE TO ACCUMULATOR
002A 3A         159         OUTL    P2,A            ;OUTPUT TO DIGIT DRIVERS
002B 1B         160         INC     R3              ;INCREMENT DIGIT COUNTER
002C D307       161         XRL     A,#07H          ;CHECK IF AT LAST DIGIT
002E 9632       162         JNZ     SETIME          ;RESET TIMER IN NOT LAST DIGIT
0030 BB00       163         MOV     R3,#00H         ;RESET DIGIT COUNTER
0032 23F1       164 SETIME: MOV     A,#TIME         ;TIMER VALUE
0034 62         165         MOV     T,A             ;LOAD TIMER
0035 55         166         STRT    T               ;START TIMER
0036 FF         167         MOV     A,R7            ;RESTORE ACCUMULATOR
0037 93         168         RETR                    ;RETURN
                169 ;*************************************************************************
                170 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                171 ;
                172 ;*****************************************************************************
                173 ;                INPUT CHARACTER AND DIGIT ROUTINE
                174 ; THIS PORTION OF THE PROGRAM IS AN INTERRUPT ROUTINE WHICH
                175 ;IS ACTED UPON WHEN THE IBF BIT IS SET.  THE ROUTINE GETS THE
                176 ;DISPLAY DATA WORD FROM THE DBB AND DEFINES BOTH THE DIGIT AND
                177 ;THE CHARACTER TO BE DISPLAYED.  THIS IS DONE BY MEANS OF A
                178 ;CHARACTER LOOP-UP TABLE AND A DISPLAY MAP FOR DIGIT AND CHARACTER
                179 ;LOCATION.  SPECIAL CONSIDERATION IS TAKEN FOR A DECIMAL POINT WHICH IS
                180 ;SIMPLY ADDED TO THE EXISTING CHARACTER IN THE DISPLAY MAP.  REGISTER
                181 ;BANK 1 IS SELECTED AND THE ACCUMULATOR IS SAVED UPON ENTERING
                182 ;THE ROUTINE.  ONCE THE DATA WORD HAS BEEN FULLY DEFINED THE ACCUMULATOR
                183 ;AND THE PRE-INTERRUPT REGISTER BANK IS RESTORED.
                184 ;
0038 D5         185 INPUT:  SEL     RB1             ;REGISTER BANK 1
0039 AF         186         MOV     R7,A            ;SAVE ACCUMULATOR
003A 22         187         IN      A,DBB           ;GET DATA
003B AA         188         MOV     R2,A            ;SAVE DATA WORD
003C 47         189         SWAP    A               ;DEFINE DIGIT LOCATION
003D 77         190         RR      A               ;
003E 5307       191         ANL     A,#07H          ;
0040 4338       192         ORL     A,#38H          ;
0042 A8         193         MOV     R0,A            ;DIGIT LOCATION IN DIGIT POINTER
0043 FA         194         MOV     A,R2            ;SAVED DATA WORD TO ACCUMULATOR
0044 531F       195         ANL     A,#1FH          ;DEFINE CHARACTER LOOK-UP-TABLE LOC.
0046 E3         196         MOVP3   A,@A            ;GET CHARACTER
0047 AA         197         MOV     R2,A            ;SAVE CHARACTER
0048 D37F       198         XRL     A,#7FH          ;IS CHARACTER DECIMAL POINT
004A C650       199         JZ      DPOINT          ;
004C FA         200         MOV     A,R2            ;SAVED CHARACTER TO ACCUMULATOR
004D A0         201         MOV     @R0,A           ;CHARACTER TO DISPLAY MAP
004E 0453       202         JMP     RETURN          ;
0050 FA         203 DPOINT: MOV     A,R2            ;SAVED CHARACTER TO ACCUMULATOR
0051 50         204         ANL     A,@R0           ;"AND" WITH OLD CHARACTER
0052 A0         205         MOV     @R0,A           ;BACK TO DISPLAY MAP
0053 FF         206 RETURN: MOV     A,R7            ;RESTORE ACCUMULATOR
0054 93         207         RETR
                208 ;*****************************************************************************
                209 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                210 ;*****************************************************************
                211 ;                        LOOK-UP TABLE
                212 ; THIS LOOK-UP TABLE ORIGINATES IN PAGE 3 OF THE UPI-41 PROGRAM
                213 ;MEMORY.  IT IS USED TO DEFINE THE CORRECT LEVEL OF EACH SEGMENT
                214 ;AND DECIMAL POINT FOR A SELECTED CHARACTER FROM THE INPUT ROUTINE.
                215 ;INVERSE LOGIC IS USED BECAUSE OF THE SPECIFIC DRIVER CIRCUITRY, THUS
                216 ;A 1 ON A GIVEN SEGMENT MEANS IT IS OFF AND A 0 MEANS IT IS ON.
                217 ;
                218                                ;*******SEGMENTS********
0300            219         ORG     300H           ;DP  G  F  E  D  C  B  A
0300 C0         220 CH0:    DB      0C0H           ;1   1  0  0  0  0  0  0
0301 F9         221 CH1:    DB      0F9H           ;1   1  1  1  1  0  0  1
0302 A4         222 CH2:    DB      0A4H           ;1   0  1  0  0  1  0  0
0303 B0         223 CH3:    DB      0B0H           ;1   0  1  1  0  0  0  0
0304 99         224 CH4:    DB      99H            ;1   0  0  1  1  0  0  1
0305 92         225 CH5:    DB      92H            ;1   0  0  1  0  0  1  0
0306 82         226 CH6:    DB      82H            ;1   0  0  0  0  0  1  0
0307 F8         227 CH7:    DB      0F8H           ;1   1  1  1  1  0  0  0
0308 80         228 CH8:    DB      80H            ;1   0  0  0  0  0  0  0
0309 98         229 CH9:    DB      98H            ;1   0  0  1  1  0  0  0
030A 88         230 CHA:    DB      88H            ;1   0  0  0  1  0  0  0
030B 83         231 CHB:    DB      83H            ;1   0  0  0  0  0  1  1
030C C6         232 CHC:    DB      0C6H           ;1   1  0  0  0  1  1  0
030D A1         233 CHD:    DB      0A1H           ;1   0  1  0  0  0  0  1
030E 86         234 CHE:    DB      86H            ;1   0  0  0  0  1  1  0
030F 8E         235 CHF:    DB      8EH            ;1   0  0  0  1  1  1  0
0310 7F         236 CHDP:   DB      7FH            ;0   1  1  1  1  1  1  1
0311 C2         237 CHG:    DB      0C2H           ;1   1  0  0  0  0  1  0
0312 89         238 CHH:    DB      89H            ;1   0  0  0  1  0  0  1
0313 FB         239 CHI:    DB      0FBH           ;1   1  1  1  1  0  1  1
0314 E1         240 CHJ:    DB      0E1H           ;1   1  1  0  0  0  0  1
0315 C7         241 CHL:    DB      0C7H           ;1   1  0  0  0  1  1  1
0316 AB         242 CHN:    DB      0ABH           ;1   0  1  0  1  0  1  1
0317 A3         243 CHO:    DB      0A3H           ;1   0  1  0  0  0  1  1
0318 8C         244 CHP:    DB      8CH            ;1   0  0  0  1  1  0  0
0319 AF         245 CHR:    DB      0AFH           ;1   0  1  0  1  1  1  1
031A 87         246 CHT:    DB      87H            ;1   0  0  0  0  1  1  1
031B C1         247 CHU:    DB      0C1H           ;1   1  0  0  0  0  0  1
031C 91         248 CHY:    DB      91H            ;1   0  0  1  0  0  0  1
031D BF         249 CHDASH: DB      0BFH           ;1   0  1  1  1  1  1  1
031E FD         250 CHAPOS: DB      0FDH           ;1   1  1  1  1  1  0  1
031F FF         251 BLANK:  DB      0FFH           ;1   1  1  1  1  1  1  1
                252 ;*****************************************************************
                253         END
```

USER SYMBOLS

BLANK	031F	BLKMAP	000E	CH0	0300	CH1	0301	CH2	0302	CH3	0303	CH4	0304	CH5	0305
CH6	0306	CH7	0307	CH8	0308	CH9	0309	CHA	030A	CHAPOS	031E	CHB	030B	CHC	030C
CHD	030D	CHDASH	031D	CHDP	0310	CHE	030E	CHF	030F	CHG	0311	CHH	0312	CHI	0313
CHJ	0314	CHL	0315	CHN	0316	CHO	0317	CHP	0318	CHR	0319	CHT	031A	CHU	031B
CHY	031C	DISPLA	001F	DPOINT	0050	INPUT	0038	LOOP	001D	RETURN	0053	SETIME	0032	START	0009
TIME	FFF1														

ASSEMBLY COMPLETE, NO ERRORS

Ref. #AB139

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 4004/4040 ☐ 8008 ☐ 8080 ☒ 8048 ☐ 8085 ☒ Other UPI-41A (use additional sheets if necessary)

Program Title

UPI-41A SENSOR MATRIX CONTROLLER

Function

This program uses the UPI-41A as a Sensor Matrix Controller. It has monitoring capabilities of up to 128 sensors. The coordinate and sensor status of each detected change is available to the master microprocessor in a single byte. A 4ØX8 FIFO queue is provided for data buffering. Both hardware or polled interrupt methods can be used to notify the master of a detected sensor change. Applications; alarm systems, control panels, various keyboards.

Required Hardware

Required Software

All information pertaining to required hardware, software, input parameters and output results are fully documented in the Intel application note AP41 ("INTRODUCTION TO THE UPI-41A") and the listing for this program.

Input Parameters

Output Results

Registers Modified: A,STS, RBØ RØ-R7	Programmer: Robin J. Jigour
RAM Required: (within UPI-41A) 3FH(see AP41)	Company: Intel
ROM Required: (within UPI-41A) 9EH(see AP41)	Address: *(Internal Use Only)
Maximum Subroutine Nesting Level: None	City:
Assembler/Compiler Used: ASM48 MOD42	State:

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                  1 ;       ********************************************
                  2 ;       *     UPI-41A SENSOR MATRIX CONTROLLER     *
                  3 ;       ********************************************
                  4 ;
                  5 ;       THIS PROGRAM USES THE UPI-41A AS A SENSOR MATRIX CONTROLLER.
                  6 ;IT HAS MONITORING CAPABILITIES OF UP TO 128 SENSORS.  THE COORDINATE
                  7 ;AND SENSOR STATUS OF EACH DETECTED CHANGE IS AVAILABLE TO THE MASTER
                  8 ;MICROPROCESSOR IN A SINGLE BYTE.  A 40X8 FIFO QUEUE IS PROVIDED FOR
                  9 ;DATA BUFFERING.  BOTH HARDWARE OR POLLED INTERRUPT METHODS CAN BE USED
                 10 ;TO NOTIFY THE MASTER OF A DETECTED SENSOR CHANGE.
                 11 ;
                 12 ;*************************************************************************
                 13 ;
                 14 ;REGISTER DEFINITIONS:
                 15 ;       REGISTER            RBQ                 RBI
                 16 ;       --------            ---                 ---
                 17 ;         R0          MATRIX MAP POINTER      NOT USED
                 18 ;         R1          FIFO POINTER            NOT USED
                 19 ;         R2          SCAN ROW SELECT         NOT USED
                 20 ;         R3          COLUMN COUNTER          NOT USED
                 21 ;         R4          FIFO-IN                 NOT USED
                 22 ;         R5          FIFO-OUT                NOT USED
                 23 ;         R6          CHANGE WORD             NOT USED
                 24 ;         R7          COMPARE                 NOT USED
                 25 ;
                 26 ;*************************************************************************
                 27 ;
                 28 ;PORT PIN DEFINITIONS:
                 29 ;
                 30 ;PIN       PORT 1 FUNCTION      PIN       PORT 2 FUNCTION
                 31 ;---       ---------------      ---       ---------------
                 32 ;P0-7      COLUMN LINE INPUTS   P0-3      ROW SELECT OUTPUTS
                 33 ;                               P4        FIFO NOT EMPTY INTERRUPT
                 34 ;                               P5        OBF INTERRUPT
                 35 ;                               P6-7      NOT USED
                 36 ;
                 37 ;*************************************************************************
                 38 ;
                 39 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                 40 ;*************************************************************************
                 41 ;
                 42 ;CHANGE WORD BIT DEFINITION:
                 43 ;
                 44 ;                BIT              FUNCTION
                 45 ;                ---              --------
                 46 ;                D0-6             SENSOR COORDINATE
                 47 ;                D7               SENSOR STATUS
                 48 ;
                 49 ;*************************************************************************
                 50 ;
                 51 ;STATUS REGISTER BIT DEFINITION:
                 52 ;
                 53 ;                BIT              FUNCTION
                 54 ;                ---              --------
                 55 ;                D0               OBF
                 56 ;                D1-3             IBF, F0, F1 (NOT USED)
                 57 ;                D4               FIFO NOT EMPTY
                 58 ;                D5-7             USED DEFINED (NOT USED)
                 59 ;
                 60 ;*************************************************************************
                 61 ;
                 62 ;                         EQUATES
                 63 ;
                 64 ;THE FOLLOWING CODE DESIGNATES THREE VARIABLES; SCANTM,FIFOBA
                 65 ;AND FIFOTA.  SCANTM ADJUSTS THE LENGTH OF A DELAY BETWEEN
                 66 ;SCANNING SWITCH.  THIS SIMULATES DEBOUNCE FUNCTIONS.  FIFOBA
                 67 ;IS THE BOTTOM ADDRESS OF THE FIFO.  FIFOTA IS THE TOP ADDRESS
                 68 ;OF THE FIFO.  THIS MAKES IT POSSIBLE TO HAVE A FIFO 3 TO 40
                 69 ;BYTES IN LENGTH.
                 70 ;
                 71 ;*************************************************************************
                 72 ;
000F             73 SCANTM  EQU     0FH             ;SCAN TIME ADJUST
0008             74 FIFOBA  EQU     08H             ;FIFO BOTTOM ADDRESS
002F             75 FIFOTA  EQU     2FH             ;FIFO TOP ADDRESS
                 76 ;
                 77 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                    78 ;*************************************************************************
                    79 ;
                    80 ;                          INITIALIZATION
                    81 ;
                    82 ;THE PROGRAM STARTS AT THE FOLLOWING CODE UPON RESET.  WITHIN
                    83 ;THIS INITIALIZATION SECTION THE REGISTERS THAT MAINTAIN THE MATRIX
                    84 ;MAP,FIFO AND ROW SCANNING ARE SET UP.  PORT 1 IS SET HIGH FOR USE
                    85 ;AS AN INPUT PORT FOR THE COLUMN STATUS.  BIT 4 OF STATUS REGISTER IS
                    86 ;WRITTEN TO CONVEY A FIFO EMPTY CONDITION.  THE INITIAL COLUMN STATUS
                    87 ;OF ALL THE ROWS IN THE SENSOR MATRIX IS THEN READ INTO THE MATRIX
                    88 ;MAP.  ONCE THE MATRIX MAP IS FILLED THE OBF INTERRUPT (PORT 2-4) IS
                    89 ;ENABLED.
                    90 ;
                    91 ;*************************************************************************
                    92 ;
0000                93         ORG     0
0000 B83F           94 INITMX: MOV     R0,#3FH          ;MATRIX MAP POINTER REGISTER,TOP ADDRESS
0002 BA0F           95         MOV     R2,#0FH          ;SCAN ROW SELECT REGISTER,TOP ROW
0004 BC08           96         MOV     R4,#FIFOBA       ;FIFO INPUT ADDRESS REGISTER,BOTTOM OF FIFO
0006 BD2F           97         MOV     R5,#FIFOTA       ;FIFO OUTPUT ADDRESS REGISTER,TOP OF FIFO
0008 89FF           98         ORL     P1,#0FFH         ;INITIALIZE PORT 1 HIGH FOR INPUTS
000A 2300           99         MOV     A,#00H           ;INITIALIZE STATUS REGISTER, FIFO EMPTY
000C 90            100         MOV     STS,A            ;WRITE TO STATUS REGISTER, BITS 4-7
000D FA            101 FILLMX: MOV     A,R2             ;SCAN ROW SELECT TO ACCUMULATOR
000E 3A            102         OUTL    P2,A             ;OUTPUT SCAN ROW SELECT TO PORT 2
000F 09            103         IN      A,P1             ;INPUT COLUMN STATUS PORT 1
0010 A0            104         MOV     @R0,A            ;LOAD MATRIX MAP WITH COLUMN STATUS
0011 FA            105         MOV     A,R2             ;CHECK SCAN ROW SELECT REGISTER VALUE FOR 0
0012 C618          106         JZ      OBFINT           ;IF 0 ENABLE OBF INTERRUPT
0014 C8            107         DEC     R0               ;DECREMENT TO NEXT MATRIX MAP ADDRESS
0015 CA            108         DEC     R2               ;DECREMENT TO SCAN NEXT ROW
0016 040D          109         JMP     FILLMX           ;FILL NEXT MATRIX MAP ADDRESS
0018 BA10          110 OBFINT: MOV     R2,#10H          ;BIT 4 HIGH IN ROW SCAN SELECT REGISTER
001A FA            111         MOV     A,R2             ;ROW SCAN SELECT VALUE TO ACCUMULATOR
001B 3A            112         OUTL    P2,A             ;INITIALIZE PORT 2, BIT 4 FOR "EN FLAGS"
001C F5            113         EN      FLAGS            ;ENABLE OBF INTERRUPT PORT 2, BIT 4
                   114 ;
                   115 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                116 ;*****************************************************************************
                117 ;
                118 ;                          SCAN AND COMPARE
                119 ;
                120 ;THE FOLLOWING CODE IS THE SCAN AND COMPARE SECTION OF THE PROGRAM.
                121 ;UPON ENTERING THIS SECTION A CHECK IS MADE TO SEE IF THE ENTIRE MATRIX
                122 ;HAS BEEN SCANNED.  IF SO THE REGISTERS THAT MAINTAIN THE MATRIX MAP AND ROW
                123 ;SCANNING ARE RESET TO THE BEGINNING OF THE SENSOR MATRIX.  IF THE ENTIRE
                124 ;MATRIX HASNT BEEN SCANNED THE REGISTERS INCREMENT TO SCAN THE NEXT ROW.
                125 ;FROM THIS POINT ON THE ROW SCAN SELECT REGISTER IS USED FOR TWO FUNCTIONS.
                126 ;BITS 0-3 FOR SCANNING AND BITS 4 AND 5 FOR THE EXTERNAL INTERRUPTS.  THUSLY
                127 ;ALL USAGE OF THE REGISTERS IS DONE BY LOGICALLY MASKING IT SO AS TO ONLY
                128 ;AFFECT THE FUNCTION DESIRED.  ONCE THE REGISTERS ARE RESET, ONE ROW OF THE
                129 ;SENSOR MATRIX IS SCANNED.  A DELAY IS EXECUTED TO ADJUST FOR SCAN TIME
                130 ;(DEBOUNCE).  A BYTE OF COLUMN STATUS IS THEN READ INTO THE MATRIX MAP.
                131 ;AT THE TIME THE NEW COLUMN STATUS IS COMPARED TO THE OLD.  THE RESULT IS
                132 ;STORED IN THE COMPARE REGISTER.  THE PROGRAM IS THEN ROUTED ACCORDING TO
                133 ;WHETHER OR NOT A CHANGE WAS DETECTED.
                134 ;
                135 ;*****************************************************************************
                136 ;
001D FA         137 ADJREG: MOV     A,R2            ;SCAN ROW SELECT TO ACCUMULATOR
001E 530F       138         ANL     A,#0FH          ;CHECK FOR 0 SCAN VALUE ONLY,NOT INTERRUPT
0020 C626       139         JZ      RSETRG          ;IF 0 RESET REGISTERS
0022 C8         140         DEC     R0              ;DECREMENT MATRIX MAP POINTER
0023 CA         141         DEC     R2              ;DECREMENT SCAN ROW SELECT
0024 042C       142         JMP     SCANMX          ;SCAN MATRIX
0026 B83F       143 RSETRG: MOV     R0,#3FH         ;RESET MATRIX MAP POINTER REGISTER,TOP ADDRESS
0028 FA         144         MOV     A,R2            ;SCAN ROW SELECT TO ACCUMULATOR
0029 430F       145         ORL     A,#0FH          ;RESET SCAN ROW SELECT,NO INTERRUPT CHANGE
002B AA         146         MOV     R2,A            ;SCAN ROW SELECT REGISTER
002C FA         147 SCANMX: MOV     A,R2            ;SCAN ROW SELECT TO ACCUMULATOR
002D 3A         148         OUTL    P2,A            ;OUTPUT SCAN ROW SELECT TO PORT 2
002E BB0F       149         MOV     R3,#SCANTM      ;SET DELAY FOR OUTPUT SCAN TIME
0030 EB30       150 DELAY2: DJNZ    R3,DELAY2       ;DELAY
0032 09         151         IN      A,P1            ;INPUT COLUMN STATUS FROM PORT 1 TO ACCUMULATOR
0033 20         152         XCH     A,@R0           ;STORE NEW COLUMN STATUS SAVE OLD IN ACCUMULATOR
0034 D0         153         XRL     A,@R0           ;COMPARE OLD WITH NEW COLUMN STATUS
0035 AF         154         MOV     R7,A            ;SAVE COMPARE RESULT IN COMPARE REGISTER
0036 C669       155         JZ      CHFFUL          ;IF THE SAME, CHECK IF FIFO IS FULL
                156 ;
                157 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                158 ;*************************************************************************
                159 ;
                160 ;                       CHANGE WORD ENCODING
                161 ;
                162 ;THE FOLLOWING CODE IS THE CHANGE WORD ENCODING SECTION.  THIS
                163 ;SECTION IS ONLY EXECUTED IF A CHANGE WAS DETECTED.  THE COLUMN COUNTER
                164 ;IS SET AND DECREMENTED TO DESIGNATE EACH OF THE 8 COLUMNS.  THE COMPARE
                165 ;REGISTER IS LOOKED AT ONE BIT AT A TIME TO FIND THE EXACT LOCATION OF
                166 ;THE CHANGE(S).  WHEN A CHANGE IS FOUND IT IS ENCODED BY GIVING IT A
                167 ;COORDINATE FOR ITS LOCATION.  THIS IS DONE BY COMBINING THE PRESENT VALUE
                168 ;IN THE ROW SCAN SELECT REGISTER AND THE COLUMN COUNTER.  THE ACTUAL STATUS
                169 ;OF THAT SENSOR IS ESTABLISHED BY LOOKING AT THE CORRESPONDING BYTE IN
                170 ;THE MATRIX MAP.  THIS STATUS IS COMBINED WITH THE COORDINATE TO ESTABLISH
                171 ;THE CHANGE WORD.  THE CHANGE WORD IS THEN STORED IN THE CHANGE WORD REGISTER.
                172 ;
                173 ;*************************************************************************
                174 ;
0038 B808       175         MOV     R3,#08H         ;SET COLUMN COUNTER REGISTER TO 8
003A CB         176 RRLOOK: DEC     R3              ;DECREMENT COLUMN COUNTER
003B F0         177         MOV     A,@R0           ;COLUMN STATUS TO ACCUMULATOR
003C 77         178         RR      A               ;ROTATE COLUMN STATUS RIGHT
003D A0         179         MOV     @R0,A           ;ROTATED COLUMN STATUS BACK TO MATRIX MAP
003E FF         180         MOV     A,R7            ;COMPARE REGISTER VALUE TO ACCUMULATOR
003F 77         181         RR      A               ;ROTATE COMPARE VALUE RIGHT
0040 AF         182         MOV     R7,A            ;ROTATED COMPARE VALUE TO COMPARE REGISTER
0041 F245       183         JB7     ENCODE          ;TEST BIT 7 IF CHANGE DETECTED ENCODE CHANGE WORD
0043 0469       184         JMP     CHFFUL          ;IF NO CHANGE IS DETECTED CHECK FOR FIFO FULL
0045 FA         185 ENCODE: MOV     A,R2            ;SCAN ROW SELECT TO ACCUMULATOR  0000XXXX
0046 530F       186         ANL     A,#0FH          ;ROTATE ONLY SCAN VALUE
0048 E7         187         RL      A               ;ROTATE LEFT                     000XXXX0
0049 E7         188         RL      A               ;ROTATE LEFT                     00XXXX00
004A E7         189         RL      A               ;ROTATE LEFT                     0XXXX000
004B 4B         190         ORL     A,R3            ;ESTABLISH MATRIX COORDINANT     0XXXXXXX
                191                                 ;(OR) COLUMN COUNTER VALUE WITH ACCUMULATOR
004C AE         192         MOV     R6,A            ;SAVE COORDINANT IN CHANGE WORD REGISTER
004D F0         193         MOV     A,@R0           ;COLUMN STATUS FROM MATRIX MAP TO ACCUMULATOR
004E 5380       194         ANL     A,#80H          ;0 ALL BITS BUT BIT 7
0050 4E         195         ORL     A,R6            ;(OR) SENSOR STATUS WITH COORDINATE FOR COMPLETED CHANGE WORD
0051 AE         196         MOV     R6,A            ;SAVE CHANGE WORD                XXXXXXXX
                197 ;
                198 $EJECT
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                199 ;*****************************************************************
                200 ;
                201 ;                   FIFO-DBBOUT MANAGEMENT
                202 ;
                203 ;THE FOLLOWING CODE IS THE FIFO-DBBOUT MANAGEMENT SECTION OF THE
                204 ;PROGRAM.  THIS SECTION TAKES AN ENCODED CHANGE WORD AND LOADS IT INTO
                205 ;THE FIFO.  THE FIFO NOT EMPTY INTERRUPT IS THEN SET AND THE FIFO-IN
                206 ;POINTER GETS UPDATED.  A FIFO FULL CONDITION IS THEN CHECKED FOR AND
                207 ;ROUTED ACCORDINGLY.  IF BOTH THE FIFO AND OBF HAVE CHANGE WORDS THE
                208 ;PROGRAM LOCKS UP UNTIL THIS HAS CHANGED.  IF THE FIFO ISNT FULL COLUMN
                209 ;COUNTER= 0, FIFO EMPTY AND OBF CONDITIONS ARE CHECKED.  THE FIFO-OUT
                210 ;POINTER IS SET AND DBBOUT IS LOADED IF THE FIFO ISNT EMPTY AND OBF ISNT
                211 ;SET.  IF THIS ISNT THE SITUATION, PROGRAM FLOW IS ROUTED BACK TO THE
                212 ;THE SCAN AND COMPARE SECTION TO SCAN THE NEXT ROW.
                213 ;
                214 ;*****************************************************************
                215 ;
0052 FC         216 LOADFF: MOV     A,R4          ;FIFO INPUT ADDRESS TO ACCUMULATOR
0053 A9         217         MOV     R1,A          ;FIFO POINTER USED FOR INPUT
0054 FE         218         MOV     A,R6          ;CHANGE WORD TO ACCUMULATOR
0055 A1         219         MOV     @R1,A         ;LOAD FIFO AT FIFO INPUT ADDRESS
0056 2310       220 STATNE: MOV     A,#10H        ;BIT 4 FOR FIFO NOT EMPTY
0058 90         221         MOV     STS,A         ;WRITE TO STATUS REGISTER, FIFO NOT EMPTY
0059 8A20       222 INTRH1: ORL     P2,#20H       ;FIFO NOT EMPTY INTERRUPT PORT 2-5 HIGH
005B FA         223         MOV     A,R2          ;ROW SCAN SELECT TO ACCUMULATOR
005C 4320       224         ORL     A,#20H        ;SAVE INTERRUPT, NO CHANGE TO SCAN VALUE
005E AA         225         MOV     R2,A          ;ROW SCAN SELECT REGISTER
005F 232F       226 ADJFIN: MOV     A,#FIFOTA     ;FIFO TOP ADDRESS TO ACCUMULATOR
0061 DC         227         XRL     A,R4          ;COMPARE WITH CURRENT FIFO INPUT ADDRESS
0062 C667       228         JZ      RSFFIN        ;IF THE SAME RESET FIFO INPUT REGISTER
0064 1C         229         INC     R4            ;NEXT FIFO INPUT ADDRESS
0065 0469       230         JMP     CHFFUL        ;CHECK FIFO FULL
0067 BC08       231 RSFFIN: MOV     R4,#FIFOBA    ;RESET FIFO INPUT REGISTER,BOTTOM OF FIFO
0069 FC         232 CHFFUL: MOV     A,R4          ;FIFO INPUT ADDRESS TO ACCUMULATOR
006A DD         233         XRL     A,R5          ;COMPARE INPUT WITH OUTPUT FIFO ADDRESS
006B 967D       234         JNZ     CHCNTR        ;IF NOT SAME CHECK COLUMN COUNTER VALUE
006D 866D       235 CHOBF1: JOBF    CHOBF1        ;IF OBF IS 1 THEN CHECK OBF
006F 232F       236 ADJFOT: MOV     A,#FIFOTA     ;FIFO TOP ADDRESS TO ACCUMULATOR
0071 DD         237         XRL     A,R5          ;COMPARE TOP TO OUTPUT FIFO ADDRESS
0072 C677       238         JZ      RSFFOT        ;IF THE SAME RESET FIFO OUTPUT REGISTER
0074 1D         239         INC     R5            ;NEXT FIFO OUTPUT ADDRESS
0075 0479       240         JMP     LOADDB        ;LOAD DBBOUT
0077 BD08       241 RSFFOT: MOV     R5,#FIFOBA    ;RESET FIFO OUTPUT ADDRESS TO BOTTOM OF FIFO
0079 FD         242 LOADDB: MOV     A,R5          ;OUTPUT FIFO ADDRESS TO ACCUMULATOR
007A A9         243         MOV     R1,A          ;FIFO POINTER USED FOR OUTPUT
007B F1         244         MOV     A,@R1         ;CHANGE WORD TO ACCUMULATOR
007C 02         245         OUT     DBB,A         ;CHANGE WORD TO DBBOUT
007D FB         246 CHCNTR: MOV     A, R3         ;COLUMN COUNTER TO ACCUMULATOR
007E 963A       247         JNZ     RRLOOK        ;IF NOT 0 FINISH CHANGE WORD ENCODING
0080 2308       248 CHFFEM: MOV     A,#FIFOBA     ;FIFO BOTTOM ADDRESS TO ACCUMULATOR
0082 DC         249         XRL     A,R4          ;COMPARE FIFO INPUT ADDRESS WITH FIFO BOTTOM ADDRESS
0083 C68C       250         JZ      ADJFEM        ;IF THE SAME,ADJUST TO CHECK FOR FIFO EMPTY
0085 FC         251         MOV     A,R4          ;FIFO INPUT ADDRESS TO ACCUMULATOR
0086 07         252         DEC     A             ;DECREMENT FIFO INPUT ADDRESS IN ACCUMULATOR
0087 DD         253         XRL     A,R5          ;COMPARE INPUT TO OUTPUT FIFO ADDRESSES
```

```
LOC  OBJ       SEQ        SOURCE STATEMENT

0088 C691       254        JZ      STATMT        ;IF SAME,WRITE STATUS REGISTER FOR FIFO EMPTY
008A 049C       255        JMP     CHOBF2        ;CHECK OBF
008C 232F       256 ADJFEM: MOV    A,#FIFOTA     ;FIFO TOP ADDRESS TO ACCUMULATOR
008E DD         257        XRL     A,R5          ;COMPARE TOP TO OUTPUT FIFO ADDRESS
008F 969C       258        JNZ     CHOBF2        ;IF NOT SAME THEN FIFO IS NOT EMPTY, CHECK OBF
0091 2300       259 STATMT: MOV    A,#00H        ;CLEAR BIT 0 FOR FIFO EMPTY
0093 90         260        MOV     STS,A         ;WRITE TO STATUS REGISTER
0094 9ADF       261 INTRLO: ANL    P2,#0DFH      ;FIFO EMPTY, INTERRUPT PORT 2-5 LOW
0096 FA         262        MOV     A,R2          ;SCAN ROW SELECT TO ACCUMULATOR
0097 53DF       263        ANL     A,#0DFH       ;SAVE INTERRUPT, NO CHANGE TO SCAN VALUE
0099 AA         264        MOV     R2,A          ;SCAN ROW SELECT REGISTER
009A 041D       265        JMP     ADJREG        ;ADJUST REGISTERS
009C 861D       266 CHOBF2: JOBF   ADJREG        ;IF OBF=1 THEN ADJUST REGISTERS
009E 046F       267        JMP     ADJFOT        ;ADJUST FIFO OUT ADDRESS TO LOAD DBBOUT
                268 ;
                269        END
```

USER SYMBOLS

ADJFEM 008C	ADJFIN 005F	ADJFOT 006F	ADJREG 001D	CHCNTR 007D	CHFFEM 0080	CHFFUL 0069	CHOBF1 006D
CHOBF2 009C	DELAY2 0030	ENCODE 0045	FIFOBA 0008	FIFOTA 002F	FILLMX 000D	INITMX 0000	INTRH1 0059
INTRLO 0094	LOADDB 0079	LOADFF 0052	OBFINT 0018	RRLOOK 003A	RSETRG 0026	RSFFIN 0067	RSFFOT 0077
SCANMX 002C	SCANTM 000F	STATMT 0091	STATNE 0056				

ASSEMBLY COMPLETE, NO ERRORS

Ref.#AB140

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☒ 8048 ☐ 8080/8085 ☐ 8086 ☒ Other UPI-41 (use additional sheets if necessary)

Program Title

AP-27 LRC Printer Controller

Function

This program uses the UPI-41 or UPI-41A as a dot matrix printer controller for the 40-column LRC 7040 dot matrix printer. The UPI buffers up to 40 ASCII characters, formats and prints the buffer whenever 40 characters or a CRLF is received whichever occurs first.

Required Hardware

All information pertaining to required hardware, software, input parameters, and output results are documented in the Intel Application Note AP-27 ("Printer Control with the UPI-41").

Required Software

Input Parameters

Output Results

Registers Modified:	See Listing	**Programmer:**	J. Beaston
RAM Required:	8 bytes	**Company:**	Intel
ROM Required:	766 bytes	**Address:**	
Maximum Subroutine Nesting Level:	--	**City:**	
Assembler/Compiler Used:	ASM48 MOD41	**State:**	

Ref. #AB141

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☒ 8048 ☐ 8080/8085 ☐ 8086 ☒ Other UPI-41A (use additional sheets if necessary)

Program Title

Combination I/O Device (UNIOD)

Function

This program uses the UPI-41A as a combination serial and parallel I/O device. The serial part is full-duplex asynchronous with programmable baud rates of 1200, 600, 300, or 110 baud. The transmitter and receiver are double buffered. The receiver checks for framing and overrun errors.

Required Hardware

The 8-bit parallel part is programmable for input or output.

Required Software

All information pertaining to required hardware, software input and output parameters is fully documented in the Intel Application Note AP-41 ("Introduction to the UPI-41A") and the listing for this program.

Input Parameters

Output Results

Registers Modified:	See listing	**Programmer:**	J. Beaston
RAM Required:	12 bytes	**Company:**	Intel
ROM Required:	363 bytes	**Address:**	
Maximum Subroutine Nesting Level:	1	**City:**	
Assembler/Compiler Used:	ASM48 MOD42	**State:**	

Ref.#AB142

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☐ 8080/8085 ☒ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

SDK-86 Serial Monitor, V1.1

Function

Interactive monitor with commands for examining/modifying registers and memory, controlling program execution using breakpoints or single step, moving memory blocks, inputting from or outputting to I/O ports, and reading and writing HEX/Object files on paper tape.

Required Hardware

SDK-86
ASR-33 teletype or CRT

Required Software

Input Parameters

See SDK-86 MCS-86 System Design Kit User's Guide, Order #98000698

Output Results

See above mentioned User's Guide

Available on non-system diskette only for $35.00 (source & object code included)

Registers Modified:	N/A	**Programmer:**	Janet Takami
RAM Required:	256 bytes	**Company:**	Intel
ROM Required:	4K bytes	**Address:**	
Maximum Subroutine Nesting Level:		**City:**	
Assembler/Compiler Used:	PL/M-86 Compiler V1.0	**State:**	

Ref.#AB143

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☐ 8080/8085 ☒ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

SDK-86 Keypad Monitor, V.1.1

Function

Interactive monitor with commands for examining/modifying registers and memory, controlling program execution using breakpoints or single step, moving memory blocks, and inputting from or outputting to I/O ports.

Required Hardware

SDK-86

Required Software

See SDK-86 MCS-86 System Design Kit User's Guide, Order #98000698

Input Parameters

Output Results

See above User's Guide

Available on non-system diskette only for $35.00 (source & object code included)

Registers Modified:	N/A	Programmer:	Janet Takami
RAM Required:	256 bytes	Company:	Intel
ROM Required:	4K bytes	Address:	
Maximum Subroutine Nesting Level:		City:	
Assembler/Compiler Used:	PL/M-86 Compiler V.1.0	State:	

Ref.#AB144

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☒ Other SDK-80 (use additional sheets if necessary)

Program Title

TAPE - Audio Tape Interface

Function

This routine outputs RAM data to an audio cassette recorder which is paralleled to a CRT terminal. The data can be read back by means of the monitor I-command so that one needs not enter any bootstrap loader.

Required Hardware

Intel SDK-80 Kit or other 8080 computer; CRT terminal (or TTY); cassette interface (see diagram included with source listing)

Required Software

SDK-80 monitor PROM

Input Parameters

Data area start, end; start program: .G13ØØ (CR)
Enter start/end-addresses: XXXX,YYYY (CR)
After the CR turn on cassette recorder immediately.
Leader and trailer of approx. 5 sec are produced by the program itself.
After data recording (visible on the terminal) the cassette recorder should be stopped while the trailer runs to avoid the sign on message being recorded.

Output Results

Data on cassette in 'I-format'

Registers Modified:	N/A	**Programmer:**	Guenter Ruschitzka
RAM Required:	70 bytes	**Company:**	
ROM Required:	SDK-80 monitor PROM	**Address:**	Im Bruehl 1
Maximum Subroutine Nesting Level:	N/A	**City:**	D-6921 Zuzenhausen
Assembler/Compiler Used:	8080	**State:**	Western Germany

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

TEXT PROCESSOR

Function

Processes the textual material into intended format by using the format command language. The command languages are interspersed within the source text. The user can specify margins, case headings and foolings, paragraphs, center text, right justify, page footnote, underline, create tables and more.

Required Hardware

Intel MDS, Disk drive, Video terminal, Printer

Required Software

ISIS-II file system

Input Parameters

TEXT <input file> [<output file>]
where
<input file> is the source textual material
<output file> is the output file mane
:LP: is for line printer
:CO: is for CRT, and :TO: is for teletype
If not specified, the default file name is used (see the user's Manual)

Output Results

The output is generated as specified.
The error message is displayed onto the CRT

Program available on diskette only for $70.00

Registers Modified:	None	Programmer:	Triyono
RAM Required:		Company:	Naval Postgraduate School
ROM Required:	8K	Address:	Code 52Bg
Maximum Subroutine Nesting Level:	2	City:	Monterey
Assembler/Compiler Used:	PL/M-80	State:	CA 93940

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☐ 8080/8085 ☐ 8086 ☒ Other UPI-41A (use additional sheets if necessary)

Program Title

8278 Keyboard/Display Controller

Function

This program is the source code for the UPI-41A based 8278 Keyboard/Display Controller. Features of the 8278 are:

--128 key scanning logic.

--16 digit LED display multiplexing.

Required Hardware

--Interface for either contact or capacitively coupled keyboards.

--N-key rollover.

Required Software

--8-character Keyboard FIFO.

--Right or left entry display.

Input Parameters

All information related to required hardware, software, input parameters, and output results is documented in the 8278 Data sheet and program listing.

Output Results

Registers Modified: See Listing	Programmer: J. Beaston
RAM Required: 64 bytes	Company: Intel
ROM Required: 965 bytes	Address:
Maximum Subroutine Nesting Level: See listing	City:
Assembler/Compiler Used: ASM48 MOD42	State:

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☐ 8080/8085 ☐ 8086 ☒ Other UPI-41A (use additional sheets if necessary)

Program Title

8295 - Dot Matrix Printer Controller

Function

This program is the source for the UPP-41A based 8295 Dot Matrix Printer Controller. It is intended to interface LRC 7040 series printers. The 8295 features are:

--On-chip 40 character buffer.
--Parallel or serial communication to master processor.
--Programmable DMA interface.
--10/12 chr/in print density.
--Single or double width printing.
--Programmable print intensity.
--Programmable line feeds.
--3 programmable tabulations.
--2 general-purpose output pins.

Required Hardware

Required Software

Input Parameters

All information related to required hardware, software, input parameters, and output results is documented in the 8295 Data Sheet and program listing.

Output Results

Registers Modified:	See Listing	Programmer:	J. Beaston
RAM Required:	48 bytes	Company:	Intel
ROM Required:	1021 bytes	Address:	
Maximum Subroutine Nesting Level:	See Listing	City:	
Assembler/Compiler Used:	ASM48 MOD42	State:	

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☐ 8080/8085 ☐ 8086 ☒ Other UPI-41 (use additional sheets if necessary)

Program Title

OLIVETTI 20-Column Printer Controller

Function

This program is for a UPI-41 controlling an Olivetti 20-column printer. The UPI interface to the master processor may be parallel via the standard peripheral interface or serial (2400 baud). The UPI accepts and buffers up to 20 ASCII characters before printing. Complete control and timing of the printer motor and solenoids is provided by the UPI.

Required Hardware

Required Software

All information pertaining to required hardware, software, input parameters, and output results are included in the attached document and program listing.

Input Parameters

Output Results

Registers Modified:	See Listing	Programmer:	J. Beaston
RAM Required:	See Listing	Company:	Intel
ROM Required:	885 Bytes	Address:	
Maximum Subroutine Nesting Level:	See Listing	City:	
Assembler/Compiler Used:	ASM48 MOD41	State:	

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☐ 8080/8085 ☐ 8086 ☒ Other 8741A (use additional sheets if necessary)

Program Title

8292 (GPIB Controller) implementation on 8741A.

Function

This software implements the IEEE 488 controller function (8292) on the 8741A.

Required Hardware

Refer to 8292 Data Sheet for required hardware.

Required Software

Refer to listing and 8292 Data Sheet for required software, input parameters, output results, and other software details.

Input Parameters

Output Results

Registers Modified:	Programmer: T. Voll
RAM Required: N/A	Company: Intel
ROM Required: 1000 10	Address:
Maximum Subroutine Nesting Level:	City:
Assembler/Compiler Used: MCS-48/UPI-41	State:

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☒ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

SEND48 - Download to PROMPT 48 for Series II.

Function

Allows user to download a HEX file from a Series II Development System (with disk) to the PROMPT 48.

Required Hardware

(1) Series II Development System with Serial Channel 2 available and unmodified.
(2) Prompt 48 strapped for RS232C, 2400 baud.
(3) Interconnecting cables.
(See program listing for details.)

Required Software

ISIS-II in Development System.
Monitor firmware in Prompt 48.

Input Parameters

See program listing for operating instructions.

Output Results

None, except for standard ISIS error messages, if appropriate.

Registers Modified: All	Programmer: P. Bushell
RAM Required: 1206H bytes including a 4K buffer whose size may be altered.	Company: MicroGenics
ROM Required: -	Address: 22, Willows Road,
Maximum Subroutine Nesting Level: (Stack= 2AH)	Bourne End, Bucks.
Assembler/Compiler Used: PLM80 ver. 3.1	SL8 5HG. ENGLAND.

insite T.M.

INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

OUTIN - PROMPT 48 or PROMPT 80 INTERFACE

Function

"OUTIN" sends (OUTBYT) and receives (INBYT) one byte to PROMPT48 or PROMPT80 via PROMPT-SPP-Cable (see source-listing)

Required Hardware

Intellec-MDS, PROMPT48 or PROMPT80, PROMPT-SPP-Cable (see attached PROMPT-SPP-Schematic)

Required Software

ISIS II (for testing: see attached PROTES)

Input Parameters

OUTBYT Register C = transmit-value
INBYT none

Output Results

OUTBYT non (but transmit-value is transmitted to PROMPT)
INBYT Resgister A = received value form PROMPT (or 0, if no startbit was detected during time-out-loop)

----for more details see OUTIN-source-listing----

Registers Modified: A, B, C, D, E	Programmer: Glasmacher
RAM Required: none	Company: Peter Glasmacher
ROM Required: 98 byte	Address: Boxberger Str. 11,D-8000
Maximum Subroutine Nesting Level: 1	City: Munchen 45,
Assembler/Compiler Used: ISIS-II 8080/8085 Macro Ass. V2.0	State: West Germany

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

REMOTE48 - INTERACTIVE CONTROLLER OF PROMPT48

Function

Remote, interactive control of PROMPT48 through MDS and CRT
(for detailed description see additional sheet)

Required Hardware

MDS with 32K-RAM, CRT or TTY, PROMPT48, PROMPT-SPP

Required Software

for use: ISIS-II
for generation: PLM80 V2.0 or V3.0 (for REMOTE.SRC)
ASM80 V2.0 (for OUTIN.SRC)

Input Parameters

type commands on consol

Output Results

Depends on typed commands:
PROMPT48-action (fe.e. signle step, go with break, etc.)
Dump of PROMPT48-data (f.e. registers, program-memory, etc.)
to consol and - if required- to an ISIS-file

The PROMPT48-data on consol are cleaned and tabulated.
an address-header is inserted after every 8 lines, to
get good readability.

The PROMPT48-data, that is sent to an ISIS-file, is in
Intel-Hex-Format.

Registers Modified:	All	Programmer:	Peter Glasmacher
RAM Required:	32K-MDS-RAM	Company:	Ingenie. Glasmacher
ROM Required:	Nothing	Address:	Boxberger Strabe II
Maximum Subroutine Nesting Level:	Less 10	City:	D-8000 Munchen 45
PLM80 V3.0 and ASM80 V2.0 Assembler/Compiler Used:		State:	West-Germany

AB153

APPENDIX to REMOTE48

REMOTE48 accepts the following commands, which correspond to one or more keys of the PROMPT48.

f.e. the single-letter-command "S" performs the following PROMPT48-keystrokes or PROMPT48-commands
(GO) (SINGLE STEP) (.) (D) (REGISTER) (0) (,) (4) (8) (.)
and prints or displays it to the consol in this form

```
ADDR +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
0000 04 67 94 D4 FF 00 00 00 01 30 FE 98 55 A3 6D 3E
0010 00 FF ......and so on.........
```

REMOTE48-Commands	= PROMPT48-Keystrokes or function
EXIT	exits or returns from REMOTE48 to ISIS
PROG	(PROGRAM MEMORY)
DAT	(DATA MEMORY)
REG	(REGISTER)
DUP	(D) (PROGRAMM MEMORY)
DUD	(D) (DATA MEMORY)
DUR	(D) (REGISTER)
C/P	(CLEAR ENTRY/PREVIOUS)
EX	(EXAMINE)
GB	(GO) (BREAK)
GN	(GO) (NO BREAK)
GS	(GO) (SINGLE STEP)
TO␣ <isis-file>	write the content of the input-buffer, f.e. dumped program-memory, to the specified ISIS-file
R	(D) (REGISTERS) (0) (,) (4) (8) (.)
S	(GO) (SINGLE STEP) (.) (D) (REGISTERS) (0) (,) (4) (8) (.)
,	(,) or (NEXT)
.	(.) or (EXECUTE/END)
0	(0)
1	(1)
etc.	etc.
9	(9)
A	(A)
etc.	etc.
E	(E)
F	(F)

(0) … (F): hexadecimal numbers

It is allowed to insert spaces between commands or hexadecimal numbers, but not within commands. It is possible to correct a line with ISIS-line-editing-commands like "rub out" etc. The PROMPT48-monitor-echo-signs like "?" or "-" are displayed on the consol. REMOTE48 asks for new command or commands with TYPE COMMAND:_

Error-reporting:

*****NO CONNECTION BETWEEN MDS AND PROMPT (PRESS (MON INT) OR (SYS RST)!)

This message is printed to the consol, if MDS and PROMPT don't communicate, f.e. PROMPT48 is switched off, the PROMPT48-key-monitor is active or the PROMPT 48 is running after "Go no break" or "Go with break".

^*****HERE OCCURED ERROR OR CONSOL-BREAK

The first sign of this message ("^") points to the first sign of the unrecognized command, that is reprinted first.

Example:

```
TYPE COMMAND: NONSENSE

TYPE COMMAND:          NONSENSE
                       ^*****HERE OCCURED ERROR OR CONSOL-BREAK
TYPE COMMAND:
```

The underlined signs were typed in by the user.

The same message appears, if the PROMPT48 was running when one hit any consol-key.

Using REMOTE48 under ISIS (a short example of a REMOTE48-session)

```
-REMOTE                  (that is the call to REMOTE48 from ISIS-level)

PROMPT48-REMOTE   VERSION 1.0
TYPE COMMAND: GS 0.      (that is one single step beinning at address 0)
???-                     (that is the echo of PROMPT48)
TYPE COMMAND: EXIT       (that means: return to ISIS)
-                        (that is the ISIS-prompt)
```

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

SYMBOL TABLE INSERTER FOR AB22

Function

Inserts labels into a disassembled object tape.

Required Hardware

MDS System

Required Software

Program AB22

Input Parameters

This program accepts:

(1) a symbol table from the console operator
(2) a disassembled object tape of the format produced by program AB22.

Output Results

Source tape with labels inserted, ready for reassembly.

Registers Modified: All	Programmer: B.A. Robinson
RAM Required: 2837	Company: Du Pont of Canada Limited
ROM Required:	Address: P.O. Box 5000
Maximum Subroutine Nesting Level: 3	City: Kingston
Assembler/Compiler Used: 8080 Macro Assem. V2.0	State: Ontario, Canada, K7L 5A5

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

PSEUDO DISASSEMBLER FOR SDK-80

Function

The program lists a machine program one instruction a line. An instruction consists of one, two or three bytes. Each line is preceded by the instruction address.

Required Hardware

SDK-80Kit for other 8080 computer

Required Software

SDK-80 Monitor PROM

Input Parameters

Start the program with G13ØØ (CR), then type in the begin address followed by (CR).
ESC terminates the listing, any other character triggers another line.

Output Results

Pseudo-disassembled machine code on console

Registers Modified:	Programmer: Guenter Ruschitzka
RAM Required: 8AH Bytes	Company:
ROM Required:	Address: Im Bruehl 1
Maximum Subroutine Nesting Level:	City: 6921 Zuzenhausen
Assembler/Compiler Used: 8080	State: Western Germany

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

SCAN - SCANNER AND SELECTIVE FILE LINE PRINTER

Function

Program allows rapid scanning of a disk text file using the console CRT. Any segment may also be directed to the Line Printer. Historical printing is used to insure all of a selected segment is printed.

Required Hardware

MDS, Floppy Disk Drive, Line Printer, CRT, 32K RAM

Required Software

ISIS-II, MDS Monitor

Input Parameters

Program implemented by entering:
SCAN FILENAME

Output Results

The selected file is output to the CRT.
Controls may be entered at any time.

CTRL/S -Freeze Output
CTRL/Q -Unfreeze Output
SPACE -Starts or stops printer. When started printer will print historically up to 22 lines. If more than this number have been output without printing, then a line of asterisks is printed to indicate the skipped data.
CTRL/C -Abort and Return to ISIS

Registers Modified:	All	Programmer:	R. Ryan
RAM Required:	32K	Company:	TOTCO
ROM Required:		Address:	P. O. Box 1307 W. Rock Creek Road
Maximum Subroutine Nesting Level:		City:	Norman
Assembler/Compiler Used:	ISIS-II 8080/8085 ASSEM. V2.0	State:	Oklahoma

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

SDK85 -- Monitor for the 8085 System Design Kit

Function

Monitor for the Intel 8085 System Design Kit (SDK-85). Provides a minimum level of utility functions for the user as well as such functions as memory and register manipulation, program loading and execution, and single step capability.

Required Hardware

SDK-85 (also supports TTY)

Required Software

None

Input Parameters

Monitor includes rudimentary command line interpreter for operator interface. See SDK-85 System Design Kit User's Manual (Manual # 9800451A) for details.

Output Results

Available on non-system diskette only for $35.00 (source & object code included)

Registers Modified: All	Programmer: This monitor is provided in ROM form to customers purchasing the SDK-85 Kit.
RAM Required: 38 bytes + stack	Company:
ROM Required: 2K bytes	Address:
Maximum Subroutine Nesting Level: 4	City:
Assembler/Compiler Used: 8080/8085 Macro Assembler	State:

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

DKDUMP - ISIS DISK FILE DUMP

Function

DUMPS ISIS disk files in HEX and ASCII to an output file (typically :CO: or :LP:)

Required Hardware

MDS DISK SYSTEM

Required Software

ISIS-II

Input Parameters

FROM KEYBOARD:
DKDUMP: :Fn:filemane to :00:
Where: (:Fn: is the drive number); (filename is the fully qualified file name); and (:00: is the output devise).

i.e. DKDUMP DKDUMP TO :LP:

Output Results

A hexadecimal interpretation of the file with a separate column for the ASCII equivalent code. 16 Bytes are displayed on each line, with a hexadecimal relative - to byte Ø of the file address displayed in the left margin.

Registers Modified:	Programmer: Stu Adler
RAM Required:	Company: Litton Energy Control
ROM Required:	Address: 8944 Mason Ave
Maximum Subroutine Nesting Level:	City: Chatsworth
Assembler/Compiler Used: PL/M-80	State: CA 91311

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☑ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

DDUMP - DISKETTE DUMP

Function

Dumps diskette on a block basis to :CO: or :LP: in hexadecimal and ASCII format

Required Hardware

MDS, MDS-DOS, console device

Required Software

MDS Monitor, ISIS-II

Input Parameters

DDUMP Program prompts for output device, beginning track and sector, ending track and sector

↑ S - suspends display
↑ Q - continues display
↑ R - restarts DDUMP
↑ E - reboots ISIS

Output Results

Outputs to selected device contents of specified block (or blocks) in hexadecimal and ASCII format

Registers Modified:	All	Programmer:	Carl Harcourt
RAM Required:	32K	Company:	Naval Avionics Center
ROM Required:		Address:	6000 E. 21st Street
Maximum Subroutine Nesting Level:		City:	Indianapolis
Assembler/Compiler Used:	ISIS-II Macro-Assembler	State:	Indiana 46218

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

RATE - BAUD RATE SELECTION FOR MDS 220 AND 230 SYSTEMS

Function

Initializes serial ports 1 and 2 for Intel 220 and 230 Development Systems

Required Hardware

Intel 220 or 230 Development System

Required Software

ISIS-II System Programs

Input Parameters

Main console

Output Results

Serial port's baud rate, stop bits, parity, word length are selected by operator for both ports

Registers Modified: 8080-all	Programmer: Tom Wrenn
RAM Required: 2K bytes	Company: Dayton Scientific Inc
ROM Required: 2K bytes	Address: 121 West Park
Maximum Subroutine Nesting Level:	City: Dayton
Assembler/Compiler Used: ASM80, V2.0	State: Ohio 45459

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

SEARCH - KEYWORD FILE SEARCH FOR ISIS-II ENVIRONMENT

Function

Searches specified source file for lines containing specified keywords. Source lines containing all the keywords (maximum of 10) are printed to the specified list file.

Required Hardware

MDS800 (with at least 32K RAM)
system console
floppy disk drive

Required Software

ISIS-II System Files - program contains linkages to ISIS-II system library routines for file input/output

Input Parameters

Command syntax is (assuming file :FØ:SEARCH is the compiled and located object module):

SEARCH FILENAME ⟨PRINT LISTFILE⟩ KEY KEYWORD1 ⟨KEYWORD2 KEYWORD3. . .⟩

WHERE: -FILENAME IS THE FILE TO BE SEARCHED
-LISTFILE IS FILE TO WHICH LINES CONTAINING SPECIFIED KEYWORDS ARE TO BE PRINTED (DEFAULTS TO :CO:)
-KEYWORD(N) ARE KEYWORDS FOR WHICH THE PROGRAM SEARCHES
-A MAXIMUM OF TEN KEYWORDS (DELIMITED BY BLANKS)
-TABS IN FILENAME ARE CONVERTED TO SPACES IN LISTFILE

Output Results

Error Messages (in addition to ISIS-II error messages):

Error #	
101	Reserved word "KEY" not found in command buffer
102	No keywords found in command buffer
103	(CRLF) (LINE SEPARATOR) was not found in source file
104	(CR) not found in command buffer

Registers Modified: ALL	Programmer: Richard D. Heslip
RAM Required: Program length about 2K	Company: Gandalf Data Communications
ROM Required: None	Address: 9 Slack Road
Maximum Subroutine Nesting Level: 1Ø	City: Ottawa
Assembler/Compiler Used: PLM80	State: Ontario CANADA

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☒ 8008 ☐ 8048 ☐ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

TIMER - MEASURES EXECUTION TIMES OF USER PROGRAMS

Function

Determines the time taken by any user program to execute

RANGE: 1 Milli second to 4½ hours

Required Hardware

MDS 800 with Real Time Clock on front panel control module

Required Software

ISIS-II operating system V2.2
MDS Monitor V2.0

Input Parameters

Starting address of the user program to be timed. The program must exit with a 'RETURN' statement as timer treats it as a subroutine

Output Results

Time in milliseconds displayed on the system console

Registers Modified: ALL	Programmer: M. Master
RAM Required: 16 bytes	Company: Univ of Ottawa Electrical Engrg. Dept
ROM Required: 181 bytes	Address: 770 King Edward Ave
Maximum Subroutine Nesting Level: User program +2	City: Ottawa, Ontario
Assembler/Compiler Used: ASM80, V2.0	State: CANADA K1N 6N5

Ref. # AB161

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

KAPIAR, V1.2 - A GENERAL PURPOSE MACRO PROCESSOR

Function

Processes user-defined MACROS. Builtin MACROS are:
DEFINED, DELETE, SAVE, IFELSE, SUBSTR, APPEND, LENGTH, TYPE
INDEX, INCR, DECR, REG, LOAD, ADD, SUB, TOKEN, LINE, and BUFFER.
Includes KAPIAR.SKL, a stripped-down version for skeleton of dedicated Macroprocessors-extensive debugging facilities.

Required Hardware

40 Kb or RAM, Diskette files, keyboard, video monitor

Required Software

ISIS-II

Input Parameters

DOES NOT APPLY

Output Results

DOES NOT APPLY

AVAILABLE ON DISKETTE ONLY

Registers Modified: ALL	Programmer: Steve Newberry
RAM Required: 40Kb	Company: Stanford University
ROM Required: None	Address:
Maximum Subroutine Nesting Level:	City: Stanford
Assembler/Compiler Used: PL/M80, V3.Ø	State: California 94305

Ref. # AB162

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

ECCO PAPER TAPE READER

Function

Loads an Intel Hexadecimal Object File Formated Paper Tape using an ECCO paper tape reader.

Required Hardware

ECCO Paper Tape Reader Model 2001-2 or equivalent
SBC 80/10 with SBC 116 or SBC 80/20

Required Software

SBC 80P Monitor or
SBC 80P20 Monitor

Input Parameters

Output Results

Memory area as specified on Intel Hexadecimal Object File Formated Paper Tape.

Registers Modified: ALL	Programmer: H.R. Pinnich, Jr.
RAM Required:	Company: Southeast Missouri State Univ.
ROM Required: OEEH	Address: Department of Chemistry
Maximum Subroutine Nesting Level:	City: Cape Girardeau
Assembler/Compiler Used: MAC80 Cross Assembler V2.4	State: MO 63701

Ref. # AB163

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

TRACE.ICE

Function

Symbolic disassembly and register dump in ICE80 .

Required Hardware

Intellec MDS, ICE80

Required Software

ISIS-II, ICE80

Input Parameters

Trace is called as an MDSCALL after GO or STEP emulation. The user types the number of emulations and TRACE halts and waits for command when they have been completed. Exit to ice command is provided by esc.

Output Results

Trace displays the current timer, all flags as symbols (e.g. +ZC) all registers in hex, P.C. in hex and symbolic, mnemonic, operand in hex and symbolic. All display is on one line. Symbols are taken from the ice symbol tables and HL/M line no. tables. In addition in GO mode Trace displays the 44 cycle history, by symbolic disassembly.

Registers Modified: N/A	Programmer: C.J. Lusby Taylor
RAM Required: 4C5H	Company: Intel International
ROM Required: None	Address: Rue du Moulin a Papier 51
Maximum Subroutine Nesting Level: N/A	City: B-1160 Brussels
Assembler/Compiler Used: ISIS ASM80 V2.0	State: Belgium

Ref. # AB164

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

DOWN 80

Function

To down load an Intel hexadecimal formatted program from a Series II to PROMPT-80/85

Required Hardware

PROMPT-SER (RS232C Cable)
User build (MDS Series II, PROMPT 80) male-to-male connector

Required Software

ISIS-II V3.4

Input Parameters

Program must be linked and located to ISIS-II system files.

Output Results

Program loaded into PROMPT RAM

Registers Modified:	Programmer: Conrad Weiderhold
RAM Required:	Company: Intel Corporation
ROM Required:	Address: 900 Jorie Blvd. Suite 220
Maximum Subroutine Nesting Level:	City: Oakbrook
Assembler/Compiler Used: PL/M80, V3.1	State: Illinois 60521

Ref. AB165

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☒ 8048 ☐ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

IBM BI-SYNC CRC16 GENERATION SUBROUTINE

Function

Generates IBM CRC 16 Check Bytes using polynomial $X^{16} + X^{15} + X^2 + 1$

Required Hardware

8048 CPU

Required Software

User Drive Program

Input Parameters

Transmission data byte in accumulator

Output Results

Reg Bank 1 R5 & R6 Contain CRC

N.B. STX Character (02H) Resets CRC
PAD Character (FFH) Ignored (IBM Convention)

Registers Modified: RBI - R4, - 7	Programmer: Andy Belton
RAM Required: User Defined	Company: Tech-Nel Data Products LTD
ROM Required: 37H Bytes	Address: PO Box 1 75 High Street
Maximum Subroutine Nesting Level: Just entry call	City: Brackley Northants
Assembler/Compiler Used: 8048/41 Macro Assembler	State: NN13 5NN England

insite T.M. **INTEL® USER'S LIBRARY SUBMITTAL FORM**

Ref. # AB166

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

HXEDIT - HEXADECIMAL DISK FILE EDITOR

Function

To modify hexadecimal disk files.

Required Hardware

MDS

Required Software

ISIS-II

Input Parameters

HXEDIT file
Program must be linked and located to ISIS-II system files.

Output Results

Arbitrary changes may be made to any located object disk file (not write-protected). Patches in machine language may be made to located objects, thereby avoiding the necessity for re-assembling and re-locating.

Registers Modified:	Programmer: Ben A. Harris
RAM Required: 1070 Bytes	Company: Techtran Industries
ROM Required:	Address: 200 Commerce Drive
Maximum Subroutine Nesting Level: 4	City: Rochester
Assembler/Compiler Used: ASM80	State: New York 14623

Ref. # AB167

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

ALPHANUMERIC INPUT FROM NUMERIC KEYBOARD

Function

Enter a set of 44 alphanumeric characters from a 16-key keyboard into a 8080 system.

Required Hardware

8080 System, 16 key keyboard,
16-character display, console

Required Software

None

Input Parameters

EXAMPLE: keyboard input sequence

" 3, *5, *7, *2, 4, 9, 8, Ø, 8, Ø, 1"

will produce text "INTEL 8080A" on the display

Output Results

Keyboard 'LF" will output this text to the console.

Registers Modified: All	Programmer: B. Hauert, R. Farkas
RAM Required: 400 bytes	Company: Battelle-Geneva
ROM Required: 20 bytes	Address: 7, route de Drize
Maximum Subroutine Nesting Level:	City: CH-1227 Carouge/Geneve
Assembler/Compiler Used: ISIS-II 8080/8085 Macro Assembler	State: Switzerland

insite T.M. **INTEL® USER'S LIBRARY SUBMITTAL FORM**

Ref. # AD6

☐ 8008 ☐ 8048 ☐ 8080/8085 ☒ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

8089 -- BREAK.89

Function

8089 Break Point routine for saving all registers and displaying to CRT.

Required Hardware

8086, 8089 MDS System, PL/M 86
(must include 8251 resident on 86BUS, channel attention decode with select=AØ.)

Required Software

Input Parameters

8089 CP, CC, TP, channel 1 or 2, Breakpoint (program requested)

*This program must be compiled (large) mode (see line 14 of list code).

Output Results

Display all registers to CRT, Restore code.

Registers Modified:	Programmer: Dave Ferguson
RAM Required:	Company: Intel Corporation
ROM Required:	Address: 3065 Bowers Avenue
Maximum Subroutine Nesting Level:	City: Santa Clara
Assembler/Compiler Used: PL/M-86	State: California 95051

Ref. AB168

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other SDK 80 (use additional sheets if necessary)

Program Title

Modified SDK-80 Restart Routine

Function

The original SDK-80 monitor restart routine destroys the carry bit because of the use of the DAD SP instruction before the flags are saved; the DAD SP instruction affects the carry bit.

This routine does not destroy the carry bit.

Required Hardware

SDK-80 Kit or any 8080 computer

Required Software

SDK-80 monitor

Input Parameters

This routine replaces the original SDK-80 monitor restart routine. To make use of it the SDK-80 owner should place a JMP 1300 instruction in memory location 13FD (user branch location). Also, he should use RST 7 instead of RST1 to re-enter the monitor. The SDK-80 monitor contains code to branch to location 13FD when a RST 7 instruction is executed. The JMP 1300 instruction stored in location 13FD transfers control to the mofidied SDK-80 restart routine that stores the register contents like the SDK-80 restart routine does but the carry bit will not be destroyed. The modified restart routine transfers control back to the SDK-80 monitor (GETCM).

Output Results

Registers Modified: ALL	Programmer: Guenter Ruschitzka
RAM Required: 27 Bytes	Company:
ROM Required:	Address: Im Bruehl 1
Maximum Subroutine Nesting Level:	City: 6921 Zuzenhausen
Assembler/Compiler Used: 8080/8085 Macro Assembler V2.0	State: Western Germany

Ref. AB169

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

MDS SERIES II - DUMB TERMINAL

Function

Allows MDS Series II keyboard and CRT to be used as a "dumb" terminal using Serial Port 2 on back panel.

Required Hardware

MDS Series II (Model 220 or 230)

Required Software

ISIS-II Operating System
SYSTEM.LIB library for CO, CSTS routines.

Input Parameters

Characters typed on keyboard of MDS.
Characters sent into serial port by modem, etc.
Control-Z terminates program, and returns to ISIS.

Output Results

Characters typed on MDS keyboard are sent out to serial port and characters received at serial port are sent to CRT on MDS.

Registers Modified:	A,C,D,E,SP,H,L	Programmer:	Dave Mabry
RAM Required:	128 Bytes	Company:	Chrysler Corporation
ROM Required:	None	Address:	CIMS-418-32-03 P.O. Box 1118
Maximum Subroutine Nesting Level:	2	City:	Detroit
Assembler/Compiler Used:	8080/8085 Macro Assembler V2.0	State:	Michigan 48288

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 ☒ 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

FILE GENERATOR

Function

To create and load a source file from an off-line terminal into an ISIS file.

Required Hardware

MDS, Bulk storage device - such as Techtran's 800 and 900 Series Disc and cassette terminals.

Required Software

ISIS-II Systems files

Input Parameters

FLGEN File

Output Results

Flgen opens file for writing and sends a Control-Q to the terminal; thereby starting reading, which continues until reading device supplies a Control-S, at which time "Continue?" is typed. Keying "Y" causes another block (to Control-S) to be read in. Codes which cause assembler errors are eliminated. Keying anything but "Y" causes Flgen to exit to ISIS.

Registers Modified: ALL	Programmer: Ben Harris
RAM Required: 495 Bytes	Company: Techtran Industries
ROM Required:	Address: 200 Commerce Drive
Maximum Subroutine Nesting Level: 4	City: Rochester
Assembler/Compiler Used: 8080/8085 Macro Assembler V2.0	State: New York 14623

insite T.M. INTEL® USER'S LIBRARY SUBMITTAL FORM

☐ 8008 ☐ 8048 XX 8080/8085 ☐ 8086 ☐ Other _____ (use additional sheets if necessary)

Program Title

PROGRAMMABLE SOFTWARE TIMERS

Function

This program allows a user to set a software timer by specifying the time (# of counts) and vector (Address of subroutine) to be executed when the timer expires. This program allows for 24 timers.

Required Hardware

Timer or other device to provide real time clock interrupts.

Required Software

None

Input Parameters

Timeon: Turns on the first available timer. The timer # is returned in the C Reg. This subroutine must be called with the time and vector on the stack.

Timeoff: Turns off the timer who's number is in the C Reg.

Output Results

Timer: Is the actual interrupt routine that counts the counters and calls for the execution of the subroutines at the associated vector.

After expiration, that timer is turned off.

Registers Modified:	None on Interrupt All on setup	Programmer:	Gary Tebbett
RAM Required:	Enough for counters	Company:	Consulting Engineer
ROM Required:		Address:	39 Hill Street
Maximum Subroutine Nesting Level:	2	City:	Pittsburg
Assembler/Compiler Used:	8080/8085 Macro Assembler V2.0	State:	California 94565

8/79

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, California 95051 • (408) 987-8080

Printed in U.S.A./B-24/0378/1.3K NCG