



**HEWLETT-PACKARD COMPANY
LOGIC SYSTEMS DIVISION**

**HP 64000
Logic Development
System**

SYSTEM RELEASE BULLETIN

Part Number: **5958-6019**
E1086

Printed: **OCTOBER 1986**

SSSSS	RRRRRR	BBBBBB
S S	R R	B B
S	R R	B B
SSSSS	RRRRRR	BBBBBB
S	R R	B B
S S	R R	B B
SSSSS	R R	BBBBBB

SYSTEM RELEASE BULLETIN

64000 Logic Development System

OCTOBER 1986

This System Release Bulletin (SRB) documents all fixes and enhancements that are incorporated in the latest release of software for the 64000 Logic Development System.

The SRB is provided as a benefit of Hewlett-Packard's Software Support Services.

The five sections of the SRB are:

SOFTWARE RELEASE CONTENTS - lists the new revision codes for the 64000 products.

PRODUCT INDEX - lists product names and numbers which are included in this issue.

KPR NUMBER INDEX - sequential list of SR numbers.

KEYWORD INDEX - brief description of each SR.

KNOWN PROBLEM REPORTS - the actual reports.

Software release contents

Product name	Product number	uu.ff
*6800 C	64821	01.06
*6800 C	300 64821S004	01.10
*6800 C	500 64821S001	01.50
*6800 C	VAX 64821S003	01.80
*6800 PASCAL	64811	01.10
*6800 PASCAL	300 64811S004	01.10
*6800 PASCAL	500 64811S001	01.40
*6800 PASCAL	VAX 64811S003	01.60
*6800/2 ASSEMB	64841	01.15
*6800/2 ASSEMB	300 64841S004	01.10
*6800/2 ASSEMB	500 64841S001	01.40
*6800/2 ASSEMB	VAX 64841S003	01.50
*68000 C	64819	01.09
*68000 C	300 64819S004	01.10
*68000 C	500 64819S001	01.50
*68000 C	VAX 64819S003	01.80
*68000 PASCAL	64815	01.11
*68000 PASCAL	300 64815S004	01.10
*68000 PASCAL	500 64815S001	01.40
*68000 PASCAL	VAX 64815S003	01.60
*6805/9 ASSEMB	64844	01.11
*6805/9 ASSEMB	300 64844S004	01.10
*6805/9 ASSEMB	500 64844S001	01.40
*6805/9 ASSEMB	VAX 64844S003	01.60
*6809 C	64822	01.07
*6809 C	300 64822S004	01.10
*6809 C	500 64822S001	01.30
*6809 C	VAX 64822S003	01.50
*6809 EMULATION	64215	01.08
*6809 PASCAL	64813	01.10
*6809 PASCAL	300 64813S004	01.10
*6809 PASCAL	500 64813S001	01.20
*6809 PASCAL	VAX 64813S003	01.30
*6809E EMULATION	64216	01.08
*8085 B PASCAL	64825	01.03
*8085 B PASCAL	300 64825S004	01.10
*8085 B PASCAL	500 64825S001	01.40
*8085 B PASCAL	VAX 64825S003	01.60
*8085 C	64826	01.03
*8085 C	300 64826S004	01.10
*8085 C	500 64826S001	01.50
*8085 C	VAX 64826S003	01.80
*8086/8 C	64818	03.01
*8086/8 C	300 64818S004	03.10
*8086/8 C	500 64818S001	03.20
*8086/8 C	VAX 64818S003	03.40
*8086/8 PASCAL	64814	03.01
*8086/8 PASCAL	300 64814S004	03.10
*8086/8 PASCAL	500 64814S001	03.10
*8086/8 PASCAL	VAX 64814S003	03.20
*F9450 EMULATION	64286	01.03
*OP_SYS DEC-VAX / VMS	64882	01.70
*OP_SYS HP-UX / 500	64880	01.60
*RS-232 TRANSFER	300 64885	01.10
*RS-232 TRANSFER	500 64884	01.10

Software release contents

Product name	Product number	uu.ff
*RS-232 TRANSFER	VAX 64886	01.10
*USER DEF ASSEMB	500 64851S001	01.50
*USER DEF ASSEMB	VAX 64851S003	01.50
*Z80 ASSEMB	64842	01.12
*Z80 ASSEMB	300 64842S004	01.10
*Z80 ASSEMB	500 64842S001	01.40
*Z80 ASSEMB	VAX 64842S003	01.60
*Z80/NSC800 C	64824	01.03
*Z80/NSC800 C	300 64824S004	01.10
*Z80/NSC800 C	500 64824S001	01.50
*Z80/NSC800 C	VAX 64824S003	01.80
*Z80/NSC800PASCAL	64823	01.03
*Z80/NSC800PASCAL	300 64823S004	01.10
*Z80/NSC800PASCAL	500 64823S001	01.40
*Z80/NSC800PASCAL	VAX 64823S003	01.60
*Z8000 C	64820	01.05
*Z8000 C	300 64820S004	01.10
*Z8000 C	500 64820S001	01.50
*Z8000 C	VAX 64820S003	01.80
*Z8000 PASCAL	64816	01.11
*Z8000 PASCAL	300 64816S004	01.10
*Z8000 PASCAL	500 64816S001	01.40
*Z8000 PASCAL	VAX 64816S003	01.60
*Z80H EMULATION	64253	01.02

Product index

Product name		Product number	Page
6800 C		64821	1
6800 C	300	64821S004	7
6800 C	500	64821S001	9
6800 C	VAX	64821S003	16
6800 PASCAL		64811	24
6800 PASCAL	300	64811S004	29
6800 PASCAL	500	64811S001	31
6800 PASCAL	VAX	64811S003	35
6800/2 ASSEMB		64841	39
6800/2 ASSEMB	300	64841S004	41
6800/2 ASSEMB	VAX	64841S003	43
68000 C		64819	45
68000 C	300	64819S004	60
68000 C	500	64819S001	63
68000 C	VAX	64819S003	72
68000 PASCAL	300	64815S004	83
68000 PASCAL	500	64815S001	86
68000 PASCAL	VAX	64815S003	90
6805/9 ASSEMB	300	64844S004	95
6805/9 ASSEMB	500	64844S001	97
6805/9 ASSEMB	VAX	64844S003	101
6809 C		64822	104
6809 C	300	64822S004	109
6809 C	500	64822S001	111
6809 C	VAX	64822S003	112
6809 PASCAL		64813	117
6809 PASCAL	300	64813S004	121
6809 PASCAL	500	64813S001	123
6809 PASCAL	VAX	64813S003	126
8085 B PASCAL		64825	130
8085 B PASCAL	300	64825S004	137
8085 B PASCAL	500	64825S001	140
8085 B PASCAL	VAX	64825S003	148
8085 C		64826	156
8085 C	300	64826S004	166
8085 C	500	64826S001	169
8085 C	VAX	64826S003	178
8086/8 C		64818	188
8086/8 C	300	64818S004	193
8086/8 C	500	64818S001	195
8086/8 C	VAX	64818S003	201
8086/8 PASCAL		64814	207
8086/8 PASCAL	300	64814S004	212
8086/8 PASCAL	500	64814S001	214
8086/8 PASCAL	VAX	64814S003	217
F9450 EMULATION		64286	220
OP_SYS DEC-VAX / VMS		64882	221
OP_SYS HP-UX / 500		64880	225
USER DEF ASSEMB	500	64851S001	228
USER DEF ASSEMB	VAX	64851S003	230
Z80 ASSEMB		64842	234
Z80 ASSEMB	300	64842S004	236
Z80 ASSEMB	500	64842S001	238
Z80 ASSEMB	VAX	64842S003	240
Z80/NSC800 C		64824	242

Product index

Product name		Product number	Page
Z80/NSC800 C	300	64824S004	254
Z80/NSC800 C	500	64824S001	256
Z80/NSC800 C	VAX	64824S003	266
Z80/NSC800PASCAL		64823	276
Z80/NSC800PASCAL	300	64823S004	286
Z80/NSC800PASCAL	500	64823S001	290
Z80/NSC800PASCAL	VAX	64823S003	300
Z8000 C		64820	310
Z8000 C	300	64820S004	313
Z8000 C	500	64820S001	315
Z8000 C	VAX	64820S003	319
Z8000 PASCAL		64816	324
Z8000 PASCAL	300	64816S004	326
Z8000 PASCAL	500	64816S001	327
Z8000 PASCAL	VAX	64816S003	330
Z80H EMULATION		64253	333

Report number index

Report #	page	Report #	page	Report #	page	Report #	page
1650004630	276	D200015396	16	D200026419	280	D200028886	292
1650006536	230	D200015446	9	D200026427	188	D200028894	302
1650007054	63	D200015453	16	D200026484	291	D200029215	268
2700004804	24	D200015644	10	D200026492	301	D200029223	258
2700005173	1	D200015651	112	D200026500	131	D200029694	104
2700005181	2	D200015669	17	D200026518	142	D200029710	112
2700005371	277	D200015883	48	D200026526	149	D200029728	315
2700005512	31	D200015891	63	D200026666	195	D200030569	31
2700005900	130	D200015909	72	D200026674	201	D200030577	35
5000084806	24	D200015990	49	D200026781	157	D200030627	86
5000095687	86	D200016014	49	D200026989	243	D200030635	90
5000096594	117	D200016022	72	D200026997	256	D200030734	51
5000103218	188	D200016030	63	D200027003	266	D200030742	51
5000103267	277	D200016063	72	D200027011	169	D200031013	64
5000104612	24	D200016071	63	D200027029	178	D200031021	73
5000104620	24	D200016329	278	D200027458	243	D200031039	65
5000107888	140	D200016337	290	D200027631	35	D200031047	74
5000109934	278	D200016345	300	D200027649	214	D200031070	39
5000114777	117	D200016592	50	D200027656	217	D200031096	43
5000115097	97	D200016600	64	D200027664	86	D200031104	159
5000115402	278	D200016618	72	D200027672	90	D200031294	189
5000118414	333	D200019307	130	D200027680	327	D200031302	195
5000118828	207	D200019877	230	D200027698	330	D200031310	201
5000119925	118	D200020115	290	D200027706	189	D200031328	52
5000120378	25	D200020123	300	D200027714	50	D200031336	65
5000121178	240	D200020131	130	D200027722	310	D200031344	74
5000124040	226	D200020149	140	D200027730	4	D200031351	310
5000126516	45	D200020156	148	D200027748	104	D200031369	315
5000132720	234	D200021725	10	D200027755	292	D200031377	319
5000135780	156	D200021733	17	D200027763	302	D200031385	4
5000136234	45	D200022434	130	D200027771	244	D200031393	11
D200008870	46	D200022442	140	D200027789	142	D200031401	18
D200013938	47	D200022459	148	D200027797	150	D200031419	105
D200013946	104	D200022467	279	D200027805	158	D200031427	246
D200013953	2	D200022475	290	D200027888	244	D200031435	259
D200013961	188	D200022483	300	D200027896	257	D200031443	269
D200013979	310	D200022491	131	D200027904	267	D200031450	171
D200013987	242	D200022509	141	D200027912	158	D200031468	180
D200013995	156	D200022517	149	D200027920	170	D200032052	52
D200014282	47	D200022525	279	D200027938	179	D200032391	105
D200014779	31	D200022533	291	D200028621	50	D200033100	189
D200014787	35	D200022541	301	D200028746	245	D200033118	195
D200014795	26	D200025387	156	D200028753	258	D200033126	201
D200014993	48	D200025668	242	D200028761	268	D200033134	53
D200015230	207	D200025676	256	D200028779	246	D200033142	66
D200015313	3	D200025684	266	D200028852	142	D200033159	75
D200015370	3	D200025692	169	D200028860	150	D200033167	311
D200015388	9	D200025700	178	D200028878	280	D200033175	315

Report number index

Report #	page	Report #	page	Report #	page	Report #	page
D200033183	319	D200036624	56	D200037804	144	D200041293	14
D200033191	4	D200036632	67	D200037812	151	D200041301	21
D200033209	11	D200036640	76	D200038273	97	D200041327	107
D200033217	18	D200036699	31	D200038281	101	D200041343	114
D200033225	247	D200036707	36	D200038950	208	D200041350	262
D200033233	259	D200036764	26	D200040204	32	D200041368	272
D200033241	269	D200036772	118	D200040212	37	D200041376	162
D200033258	159	D200036780	208	D200040220	33	D200041384	175
D200033266	171	D200036798	324	D200040238	37	D200041392	184
D200033274	180	D200036806	281	D200040246	294	D200041749	145
D200033407	234	D200036814	132	D200040253	304	D200041756	152
D200033423	39	D200036871	214	D200040261	133	D200041830	58
D200033449	54	D200036939	57	D200040279	144	D200041848	69
D200033613	55	D200036947	87	D200040287	152	D200041855	78
D200034108	280	D200036954	91	D200040618	174	D200042044	228
D200034132	292	D200036962	32	D200040626	183	D200043372	127
D200034140	302	D200036970	36	D200040634	191	D200043398	333
D200034157	132	D200036988	123	D200040642	198	D200043422	59
D200034165	143	D200036996	126	D200040659	204	D200043570	221
D200034173	150	D200037002	217	D200040667	57	D200043588	225
D200034181	123	D200037010	87	D200040675	69	D200043596	250
D200034199	126	D200037028	91	D200040683	78	D200043851	294
D200034207	87	D200037036	327	D200040691	311	D200043869	304
D200034215	90	D200037044	330	D200040709	316	D200043935	221
D200034264	247	D200037051	197	D200040717	320	D200043943	58
D200034272	260	D200037069	203	D200040725	5	D200043968	250
D200034280	270	D200037077	68	D200040733	13	D200044032	70
D200034298	160	D200037085	77	D200040741	20	D200044040	79
D200034306	172	D200037093	316	D200040758	107	D200044685	250
D200034314	181	D200037101	320	D200040774	113	D200044719	294
D200034959	26	D200037119	13	D200040782	249	D200044727	304
D200035782	190	D200037127	20	D200040790	262	D200044735	134
D200035790	196	D200037143	113	D200040808	272	D200044743	145
D200035808	202	D200037150	293	D200040816	162	D200044750	152
D200035816	55	D200037168	303	D200040824	174	D200045054	221
D200035824	66	D200037176	261	D200040832	183	D200045237	119
D200035832	75	D200037184	271	D200041145	134	D200045245	107
D200035840	12	D200037192	143	D200041186	249	D200045518	251
D200035857	19	D200037200	151	D200041194	191	D200045526	251
D200035865	106	D200037218	173	D200041202	198	D200045856	79
D200035881	112	D200037226	182	D200041210	204	D200045872	252
D200035899	248	D200037234	208	D200041228	58	D200045906	198
D200035907	260	D200037291	215	D200041236	69	D200045914	204
D200035915	270	D200037309	218	D200041244	78	D200045922	79
D200035923	160	D200037465	161	D200041251	312	D200045930	317
D200035931	172	D200037663	27	D200041269	317	D200045948	321
D200035949	181	D200037713	27	D200041277	321	D200045955	14
D200036509	234	D200037796	133	D200041285	5	D200045963	21

Report number index

Report #	page	Report #	page	Report #	page	Report #	page
D200045989	114	D200047506	205	D200048728	60	D200049288	96
D200045997	263	D200047514	59	D200048736	38	D200049395	233
D200046003	272	D200047522	70	D200048744	30	D200049635	199
D200046011	175	D200047530	79	D200048751	125	D200049650	71
D200046029	184	D200047548	312	D200048769	128	D200049684	317
D200046037	163	D200047555	317	D200048777	122	D200049718	14
D200046078	263	D200047563	321	D200048785	216	D200049742	111
D200046086	273	D200047571	6	D200048793	219	D200049775	264
D200046110	222	D200047589	14	D200048801	213	D200049809	176
D200046144	222	D200047597	21	D200048819	89	D200049841	192
D200046151	34	D200047605	108	D200048827	94	D200049858	199
D200046177	252	D200047621	114	D200048835	85	D200049866	205
D200046185	264	D200047639	281	D200048843	329	D200049874	193
D200046193	273	D200047647	295	D200048850	332	D200049890	283
D200046201	175	D200047654	305	D200048868	326	D200050260	7
D200046219	184	D200047662	253	D200048876	200	D200050278	109
D200046276	198	D200047670	264	D200048884	206	D200050740	254
D200046318	215	D200047688	274	D200048892	194	D200050757	166
D200046607	204	D200047696	134	D200048900	71	D200050922	92
D200046615	218	D200047704	145	D200048918	82	D200050955	92
D200046631	208	D200047712	153	D200048926	62	D200051011	83
D200046748	215	D200047720	163	D200048934	318	D200051078	109
D200046755	218	D200047738	176	D200048942	323	D200051110	83
D200046797	39	D200047746	185	D200048959	314	D200051193	60
D200046813	43	D200047811	79	D200048967	15	D200051235	193
D200046821	234	D200047944	281	D200048975	23	D200051243	61
D200046839	238	D200047969	222	D200048983	8	D200051250	313
D200046847	240	D200047985	222	D200049007	116	D200051268	7
D200046896	98	D200048025	223	D200049015	110	D200051284	114
D200046904	101	D200048066	228	D200049023	299	D200051292	109
D200047019	228	D200048074	282	D200049031	309	D200051300	254
D200047027	230	D200048090	295	D200049049	289	D200051318	166
D200047332	27	D200048108	305	D200049056	265	D200051359	94
D200047340	33	D200048116	283	D200049064	275	D200051508	83
D200047357	37	D200048207	43	D200049072	255	D200051631	84
D200047365	119	D200048215	41	D200049080	147	D200051870	29
D200047373	124	D200048223	238	D200049098	155	D200051888	29
D200047381	127	D200048231	240	D200049106	139	D200051987	27
D200047399	208	D200048249	236	D200049114	177	D200052001	166
D200047407	215	D200048280	98	D200049122	187	D200052084	137
D200047415	218	D200048298	102	D200049130	168	D200052217	33
D200047431	88	D200048306	95	D200049189	44	D200052225	33
D200047449	92	D200048413	231	D200049197	42	D200052241	284
D200047456	324	D200048645	124	D200049205	239	D200052258	193
D200047464	328	D200048652	127	D200049213	241	D200052266	62
D200047472	331	D200048660	121	D200049221	237	D200052274	313
D200047480	192	D200048702	70	D200049262	100	D200052282	8
D200047498	199	D200048710	80	D200049270	103	D200052290	110

Report number index

Report #	page	Report #	page	Report #	page	Report #	page
D200052308	254	D200053207	241	D200055293	167	D200059014	22
D200052357	296	D200053215	236	D200055525	229	D200059022	8
D200052365	306	D200053306	44	D200055533	232	D200059048	116
D200052373	286	D200053314	41	D200055608	39	D200059055	110
D200052381	134	D200053322	239	D200055939	99	D200059063	265
D200052399	145	D200053330	236	D200058693	38	D200059071	275
D200052407	153	D200053371	99	D200058701	30	D200059089	255
D200052415	137	D200053389	102	D200058719	125	D200059097	177
D200052449	28	D200053397	95	D200058727	128	D200059105	187
D200052456	34	D200053496	228	D200058735	122	D200059113	168
D200052464	37	D200053504	231	D200058743	216	D200059121	38
D200052472	29	D200053728	210	D200058750	219	D200059139	30
D200052480	119	D200053736	211	D200058768	212	D200059147	125
D200052498	124	D200053744	297	D200058776	89	D200059154	128
D200052506	127	D200053751	307	D200058784	94	D200059162	122
D200052514	121	D200053769	287	D200058792	85	D200059170	216
D200052522	208	D200053777	163	D200058800	328	D200059188	219
D200052530	215	D200053819	223	D200058818	331	D200059196	212
D200052548	218	D200053884	224	D200058826	326	D200059204	89
D200052555	212	D200053892	223	D200058834	299	D200059212	94
D200052571	88	D200053900	223	D200058842	309	D200059220	85
D200052589	93	D200054312	226	D200058859	289	D200059238	299
D200052597	84	D200054320	225	D200058867	147	D200059246	309
D200052605	324	D200054338	225	D200058875	154	D200059253	289
D200052613	328	D200054346	225	D200058883	139	D200059261	147
D200052621	331	D200054635	82	D200058917	199	D200059279	155
D200052639	326	D200055129	205	D200058925	206	D200059287	139
D200052647	297	D200055137	81	D200058933	194	D200059295	229
D200052654	307	D200055145	321	D200058941	71	D200059303	232
D200052662	287	D200055152	21	D200058958	82	D200059410	232
D200052670	135	D200055160	115	D200058966	62	D200059949	229
D200052688	146	D200055178	274	D200058974	318	D200059956	233
D200052696	154	D200055186	185	D200058982	322	D200060269	226
D200052704	138	D200055251	176	D200058990	313	D200060277	226
D200053181	209	D200055277	164	D200059006	14	D200060301	220
D200053199	238	D200055285	186				

Keyword index

- 6800 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64821	01.04	No form feed between the expanded listing and the cross reference table.	D200027730	4
	64821	01.04	++ and -- operators evaluated with improper precedence.	D200031385	4
	64821	01.04	Comparing character to zero in while loop generates incorrect code.	D200033191	4
	64821	01.04	Problem with integer pointer in conditional statement.	D200041285	5
	64821	01.04	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047571	6
CODE GENERATOR	64821	01.02	16 bit comparison on a 8 bit unsigned short field.	2700005173	1
	64821	01.02	Left shift operator when shifting by one in a logical expr. is incorrect	2700005181	2
	64821	01.04	An erroneous CLRA is generated if a char var. is decr. in a "while" loop	D200015313	3
	64821	01.04	A shift assignment operation (<=<) generates incorrect code.	D200015370	3
PASS 1	64821	01.04	No warning or error: taking the size of a struct var. not declared.	D200013953	2
PASS 3	64821	01.04	Pass 3 fails to detect relative jump address out-of-range.	D200040725	5

- 6800 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64821S004	00.00	Linker output file should use alternate file extension.	D200048983	8
	64821S004	01.00	++ and -- operators evaluated with improper precedence.	D200051268	7
	64821S004	01.00	Host compilers do not put absolute pats specifications in relocatables	D200059022	8
CODE GENERATOR	64821S004	00.00	Incorrect opcode "MOV A,ACC" allowed by our assembler	D200052282	8
PASS 1	64821S004	01.00	Incorrect code is generated when complementing a parm. in a return stmt.	D200050260	7

- 6800 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64821S001	00.00	Linker output file should use alternate file extension.	D200048967	15
	64821S001	00.00	NO CROSS REFERENCE TABLE IS GENERATED	D200049718	14
	64821S001	01.10	Left shift operator when shifting by one in a logical expr. is incorrect	D200021725	10
	64821S001	01.10	++ and -- operators evaluated with improper precedence.	D200031393	11
	64821S001	01.10	Comparing character to zero in while loop generates incorrect code.	D200033209	11
	64821S001	01.20	Problem with integer pointer in conditional statement.	D200041293	14
	64821S001	01.20	Title description is incorrect.	D200045955	14
	64821S001	01.20	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047589	14
CODE GENERATOR	64821S001	01.40	Host compilers do not put absolute pats specifications in relocatables	D200059006	14
	64821S001	01.00	An erroneous CLRA is gen. if a char var. is the counter in a "while"	D200015388	9
	64821S001	01.00	A shift assignment operation (<=<) generates incorrect code.	D200015446	9
	64821S001	01.10	16 bit comparison on a 8 bit unsigned short field.	D200035840	12
PASS 1	64821S001	01.00	Incorrect code is generated when complementing a parm. in a return stmt.	D200015644	10
PASS 3	64821S001	01.20	Compiler option \$LIST OBJ ON\$ generates wrong output information.	D200037119	13
	64821S001	01.20	Pass 3 fails to detect relative jump address out-of-range.	D200040733	13

- 6800 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64821S003	00.00	Linker output file should use alternate file extension.	D200048975	23
	64821S003	01.10	Left shift operator when shifting by one in a logical expr. is incorrect	D200021733	17
	64821S003	01.20	++ and -- operators evaluated with improper precedence.	D200031401	18
	64821S003	01.20	Comparing character to zero in while loop generates incorrect code.	D200033217	18
	64821S003	01.20	Problem with integer pointer in conditional statement.	D200041301	21
	64821S003	01.20	Title description is incorrect.	D200045963	21
	64821S003	01.20	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047597	21
	64821S003	01.50	Compilation on the VAX using batch mode generates incorrect listing file	D200055152	21

Keyword index

- 6800 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64821S003	01.50	Host compilers do not put absolute pats specifications in relocatables	D200059014	22
CODE GENERATOR	64821S003	01.00	An erroneous CLRA is gen. if a char var. is used as a ctr. in a "while"	D200015396	16
	64821S003	01.00	A shift assignment operation (<=<) generates incorrect code.	D200015453	16
	64821S003	01.20	16 bit comparison on a 8 bit unsigned short field.	D200035857	19
PASS 1	64821S003	01.00	Incorrect code is generated when complementing a parm. in a return stmt.	D200015669	17
PASS 3	64821S003	01.20	Compiler option \$LIST_OBJ ON\$ generates wrong output information.	D200037127	20
	64821S003	01.20	Pass 3 fails to detect relative jump address out-of-range.	D200040741	20

- 6800 PASCAL -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64811	01.00	Statement Sequences.	D200014795	26
	64811	01.08	"IF B2" after "REPEAT..UNTIL B1 OR B2" doesn't work.	D200034959	26
	64811	01.08	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047332	27
	64811	01.09	Missing semicolon causes compiler to hang in Pass 1.	D200052449	28
CONSTANTS	64811	01.09	Constants may not be assigned their full 32 bit values.	D200051987	27
DEBUG LIBRARY	64811	01.08	X-reg modified after MUL or DIV operations.	2700004804	24
INCLUDE	64811	01.08	Nested INCLUDE files 3 or more deep cause 64000 to "hang" in pass 3.	D200036764	26
PARAMETERS	64811	01.08	Incorrect parameter passing with \$RANGE ON\$.	5000084806	24
	64811	01.08	Compiler accepts actual and formal parameters of different types.	5000120378	25
PASS 2	64811	01.08	Stops in Pass 2 if a long program using real with \$RANGE ON\$.	D200037663	27
	64811	01.08	ODD(INTEGER) in recursive procedure causes too many pass 2 errors.	D200037713	27
RANGE	64811	01.08	Incorrect parameter passing with \$RANGE ON\$.	5000084806	24
	64811	01.08	Incorrect code generated for multiple array comparisons.	5000104612	24
	64811	01.08	RECORD accesses using WITH generate call to EMPTY_SET if \$RANGE ON\$.	5000104620	24
	64811	01.08	Stops in Pass 2 if a long program using real with \$RANGE ON\$.	D200037663	27
REAL	64811	01.08	Stops in Pass 2 if a long program using real with \$RANGE ON\$.	D200037663	27

- 6800 PASCAL -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64811S004	00.00	Linker output file should use alternate file extension.	D200048744	30
	64811S004	01.00	Missing semicolon causes compiler to hang in Pass 1.	D200052472	29
	64811S004	01.00	Host compilers do not put absolute pats specifications in relocatables	D200059139	30
PREPROCESSOR	64811S004	01.00	Preprocessor reports errors when symbols hp64000, vms or hpux w #if	D200058701	30
RANGE	64811S004	01.00	Incorrect code generated for multiple array comparisons.	D200051870	29
	64811S004	01.00	RECORD accesses using WITH generate call to EMPTY_SET if \$RANGE ON\$.	D200051888	29

- 6800 PASCAL -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64811S001	00.00	Linker output file should use alternate file extension.	D200046151	34
	64811S001	01.00	Statement sequences.	D200014779	31
	64811S001	01.08	No form feed between the expanded listing and the cross reference table.	2700005512	31
	64811S001	01.20	"IF B2" after "REPEAT..UNTIL B1 OR B2" doesn't work.	D200036699	31
	64811S001	01.20	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047340	33
	64811S001	01.30	Host compilers do not put absolute pats specifications in relocatables	D200052217	33
	64811S001	01.30	Missing semicolon causes compiler to hang in Pass 1.	D200052456	34
PARAMETERS	64811S001	01.10	Incorrect parameter passing with \$RANGE ON\$.	D200030569	31
PASS 3	64811S001	01.20	Compiler option \$LIST_OBJ ON\$ generates wrong output information.	D200036962	32
PREPROCESSOR	64811S001	01.30	Preprocessor reports errors when symbols hp64000, vms or hpux w #if	D200052225	33

Keyword index

- 6800 PASCAL -

Keyword	Product number	uu.ff	Description	Report #	page
RANGE	64811S001	01.20	Incorrect code generated for multiple array comparisons.	D200040204	32
	64811S001	01.20	RECORD accesses using WITH generate call to EMPTY_SET_ if \$RANGE ON\$.	D200040220	33

- 6800 PASCAL -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64811S003	00.00	Linker output file should use alternate file extension.	D200048736	38
	64811S003	01.00	Statement sequences.	D200014787	35
	64811S003	01.20	No form feed between the expanded listing and the cross reference table.	D200027631	35
	64811S003	01.20	"IF B2" after "REPEAT..UNTIL B1 OR B2" doesn't work.	D200036707	36
	64811S003	01.20	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047357	37
	64811S003	01.40	Missing semicolon causes compiler to hang in Pass 1.	D200052464	37
	64811S003	01.40	Host compilers do not put absolute pats specifications in relocatables	D200059121	38
PARAMETERS	64811S003	01.20	Incorrect parameter passing with \$RANGE ON\$.	D200030577	35
PASS 3	64811S003	01.20	Compiler option \$LIST_OBJ ON\$ generates wrong output information.	D200036970	36
PREPROCESSOR	64811S003	01.40	Preprocessor reports errors when symbols hp64000, vms or hpux w #if	D200058693	38
RANGE	64811S003	01.20	Incorrect code generated for multiple array comparisons.	D200040212	37
	64811S003	01.20	RECORD accesses using WITH generate call to EMPTY_SET_ if \$RANGE ON\$.	D200040238	37

- 6800/2 ASSEMB -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64841	01.13	Assembler flagging out of range error when it should not.	D200031070	39
	64841	01.13	Error when using .NT. operator with immediate value whose MSB is set.	D200033423	39
	64841	01.13	Assembler should denote an error on non-absolute .SET expressions.	D200046797	39
	64841	01.14	Four bit operations are now unsupported.	D200055608	39

- 6800/2 ASSEMB -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64841S004	00.00	Linker output file should use alternate file extension.	D200049197	42
	64841S004	01.00	Macro def. including .IF, within a IF causes assembler to stop code gen.	D200053314	41
MACRO	64841S004	01.00	Conditional instr. .IF with rational oper. in Macro creates bad code	D200048215	41

- 6800/2 ASSEMB -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64841S003	00.00	Linker output file should use alternate file extension.	D200049189	44
	64841S003	01.20	Assembler flagging out of range error when it should not.	D200031096	43
	64841S003	01.20	Assembler should denote an error on non-absolute .SET expressions.	D200046813	43
	64841S003	01.40	Macro def. including .IF, within a IF causes assembler to stop code gen.	D200053306	44
MACRO	64841S003	01.40	Conditional instr. .IF with rational oper. in Macro creates bad code	D200048207	43

- 68000 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64819	01.07	Incorrect code when hex values are bit or-ed and passed as parameters.	5000126516	45
	64819	01.07	No error generated when an interrupt routine is explicitly called.	D200015883	48
	64819	01.07	No form feed between the expanded listing and the cross reference table.	D200027714	50

Keyword index

- 68000 C -

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64819	01.07 Comp_symb file not being loaded on user specified disc.	D200028621	50
	64819	01.07 ++ and -- operators evaluated with improper precedence.	D200031328	52
	64819	01.07 Comparing character to zero in while loop generates incorrect code.	D200033134	53
	64819	01.07 Case statement involving double indirection is not generating right code	D200033449	54
	64819	01.07 RTS rather than RTE generated to return from interrupt routine.	D200033613	55
	64819	01.07 Passing a complicated expression as a parameter may generate bad code.	D200036624	56
	64819	01.07 Problem with integer pointer in conditional statement.	D200041228	58
	64819	01.07 Compiler calculating wrong offset to parameter.	D200041830	58
	64819	01.07 Compiler generating inefficient code for certain "switch" statements.	D200043422	59
	64819	01.07 TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047514	59
CODE GENERATOR	64819	00.56 Station reboot or bad code, statements of the form: x += (*ptr)*(*ptr);	D200008870	46
	64819	01.07 Comparing a variable to zero in a "for" statement often fails.	D200014282	47
	64819	01.07 Argument of a switch is sign-extended to long when it should remain int.	D200014993	48
	64819	01.07 Wrong addressing mode used with \$BASE_PAGE\$ on in ASM68000 file.	D200015990	49
	64819	01.07 The wrong byte is accessed when a union is defined within a struct.	D200016014	49
	64819	01.07 Structure with an odd-numbered char or short array gens. wrong code.	D200016592	50
	64819	01.07 Incorrect code generated if fields are defined in a structure.	D200030734	51
	64819	01.07 Variable may not be defined before an array in a structure.	D200030742	51
	64819	01.07 16 bit comparison on a 8 bit unsigned short field.	D200035816	55
PASS 1	64819	01.07 No warning or error: taking the sizeof a struct var. not declared.	D200013938	47
	64819	01.07 Multiple warning's may cause messages to be intermixed.	D200036939	57
PASS 2	64819	01.07 Stations jumps to PV when compiling file with syntax error.	D200032052	52
PASS 3	64819	01.00 Pass 3 error flagged when 143-146 external functions are declared.	5000136234	45
	64819	01.07 Pass 3 fails to detect relative jump address out-of-range.	D200040667	57
	64819	01.07 ASM reloc. and compiler reloc differ.	D200043943	58

- 68000 C -

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64819S004	00.00 Linker output file should use alternate file extension.	D200048926	62
	64819S004	01.00 Incorrect code when hex values are bit or-ed and passed as parameters.	D200048728	60
	64819S004	01.00 ++ and -- operators evaluated with improper precedence.	D200051243	61
	64819S004	01.00 Host compilers do not put absolute pats specifications in relocatables	D200058966	62
CODE GENERATOR	64819S004	00.00 Incorrect opcode "MOV A,ACC" allowed by our assembler	D200052266	62
	64819S004	01.00 Incorrect code generated if fields are defined in a structure.	D200051193	60

- 68000 C -

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64819S001	00.00 Linker output file should use alternate file extension.	D200048900	71
	64819S001	00.00 NO CROSS REFERENCE TABLE IS GENERATED	D200049650	71
	64819S001	01.00 No error generated when an interrupt routine is explicitly called.	D200015891	63
	64819S001	01.10 ++ and -- operators evaluated with improper precedence.	D200031336	65
	64819S001	01.10 Comparing character to zero in while loop generates incorrect code.	D200033142	66
	64819S001	01.20 Passing a complicated expression as a parameter may generate bad code.	D200036632	67
	64819S001	01.20 Problem with integer pointer in conditional statement.	D200041236	69
	64819S001	01.20 Compiler calculating wrong offset to parameter.	D200041848	69
	64819S001	01.20 TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047522	70
	64819S001	01.40 Declaring 128 external functions causes compiler to bomb in code.	1650007054	63
	64819S001	01.40 Incorrect code when hex values are bit or-ed and passed as parameters.	D200048702	70
	64819S001	01.40 Host compilers do not put absolute pats specifications in relocatables	D200058941	71
CODE GENERATOR	64819S001	01.00 Wrong addressing mode used with \$BASE_PAGE\$ on in ASM68000 file.	D200016030	63

Keyword index

- 68000 C -

Keyword	Product number	uu.ff	Description	Report #	page
CODE GENERATOR	64819S001	01.00	The wrong byte is accessed when a union is defined within a structure.	D200016071	63
	64819S001	01.10	Structure with an odd-numbered char or short array gens. wrong code.	D200016600	64
	64819S001	01.10	Incorrect code generated if fields are defined in a structure.	D200031013	64
	64819S001	01.10	Variable may not be defined before an array in a structure.	D200031039	65
	64819S001	01.10	16 bit comparison on a 8 bit unsigned short field.	D200035824	66
PASS 3	64819S001	01.20	Compiler option \$LIST_OBJ_ON\$ generates wrong output information.	D200037077	68
	64819S001	01.20	Pass 3 fails to detect relative jump address out-of-range.	D200040675	69
	64819S001	01.20	ASM reloc. and compiler reloc differ.	D200044032	70

- 68000 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64819S003	00.00	Linker output file should use alternate file extension.	D200048918	82
	64819S003	01.00	No error code generated when an interrupt is explicitly called.	D200015909	72
	64819S003	01.20	++ and -- operators evaluated with improper precedence.	D200031344	74
	64819S003	01.20	Comparing character to zero in while loop generates incorrect code.	D200033159	75
	64819S003	01.20	Passing a complicated expression as a parameter may generate bad code.	D200036640	76
	64819S003	01.20	Problem with integer pointer in conditional statement.	D200041244	78
	64819S003	01.20	Compiler calculating wrong offset to parameter.	D200041855	78
	64819S003	01.20	Title description is incorrect.	D200045856	79
	64819S003	01.20	Title description is incorrect.	D200045922	79
	64819S003	01.20	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047530	79
	64819S003	01.20	Illegal instruction being generated by compiler.	D200047811	79
	64819S003	01.50	Incorrect code when hex values are bit or-ed and passed as parameters.	D200048710	80
	64819S003	01.50	Compilation on the VAX using batch mode generates incorrect listing file	D200055137	81
	64819S003	01.50	Host compilers do not put absolute pats specifications in relocatables	D200058958	82
CODE GENERATOR	64819S003	01.00	Wrong addressing mode used with \$BASE_PAGES on in ASM68000 file.	D200016022	72
	64819S003	01.00	The wrong byte is accessed when a union is defined within a structure.	D200016063	72
	64819S003	01.10	Structure with an odd-numbered char or short array gens. wrong code.	D200016618	72
	64819S003	01.20	Incorrect code generated if fields are defined in a structure.	D200031021	73
	64819S003	01.20	Variable may not be defined before an array in a structure.	D200031047	74
	64819S003	01.20	16 bit comparison on a 8 bit unsigned short field.	D200035832	75
ENHANCEMENT	64819S003	01.50	68010 directive not supported on the 9000.	D200054635	82
PASS 3	64819S003	01.20	Compiler option \$LIST_OBJ_ON\$ generates wrong output information.	D200037085	77
	64819S003	01.20	Pass 3 fails to detect relative jump address out-of-range.	D200040683	78
	64819S003	01.20	ASM reloc. and compiler reloc differ.	D200044040	79

- 68000 PASCAL -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64815S004	00.00	Linker output file should use alternate file extension.	D200048835	85
	64815S004	01.00	Program causes compiler to hang up.	D200051011	83
	64815S004	01.00	Missing semicolon causes compiler to hang in Pass 1.	D200052597	84
	64815S004	01.00	Host compilers do not put absolute pats specifications in relocatables	D200059220	85
BOOLEAN	64815S004	01.00	NOT(function) as boolean expression in "IF" statement doesn't work.	D200051110	83
CODE GENERATOR	64815S004	01.00	B := ABS(B) fails to write to the data area.	D200051508	83
PASS 2	64815S004	01.00	K := K + K + K; causes too many pass 2 errors to continue.	D200051631	84
PREPROCESSOR	64815S004	01.00	Preprocessor reports errors when symbols hp64000, vms or hpux w #if	D200058792	85

Keyword index

- 68000 PASCAL -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64815S001	00.00	Linker output file should use alternate file extension.	D200048819	89
	64815S001	01.10	No form feed between the expanded listing and the cross reference table.	D200027664	86
	64815S001	01.20	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047431	88
	64815S001	01.30	Missing semicolon causes compiler to hang in Pass 1.	D200052571	88
	64815S001	01.30	Host compilers do not put absolute pats specifications in relocatables	D200059204	89
BOOLEAN	64815S001	01.10	NOT(function) as boolean expression in "IF" statement doesn't work.	D200030627	86
CASE STATEMENT	64815S001	01.10	Different code generated between Host and 64000 for case statement.	5000095687	86
CODE GENERATOR	64815S001	01.10	B := ABS(B) fails to write to the data area.	D200034207	87
PASS 2	64815S001	01.20	K := K + K + K; causes too many pass 2 errors to continue.	D200036947	87
PASS 3	64815S001	01.20	Compiler option \$LIST_OBJ ON\$ generates wrong output information.	D200037010	87
PREPROCESSOR	64815S001	01.30	Preprocessor reports errors when symbols hp64000, vms or hpux w #if	D200058776	89

- 68000 PASCAL -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64815S003	00.00	Linker output file should use alternate file extension.	D200048827	94
	64815S003	01.20	No form feed between the expanded listing and the cross reference table.	D200027672	90
	64815S003	01.20	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047449	92
	64815S003	01.30	Program causes compiler to hang up.	D200050922	92
	64815S003	01.30	Compiler generates illegal 68000 instruction LEAMOVEM.L	D200050955	92
	64815S003	01.30	Request for date and time of link on linker output file.	D200051359	94
	64815S003	01.30	Missing semicolon causes compiler to hang in Pass 1.	D200052589	93
	64815S003	01.30	Host compilers do not put absolute pats specifications in relocatables	D200059212	94
BOOLEAN	64815S003	01.20	NOT(function) as boolean expression in "IF" statement doesn't work.	D200030635	90
CODE GENERATOR	64815S003	01.20	B := ABS(B) fails to write to the data area.	D200034215	90
PASS 2	64815S003	01.20	K := K + K + K; causes too many pass 2 errors to continue.	D200036954	91
PASS 3	64815S003	01.20	Compiler option \$LIST_OBJ ON\$ generates wrong output information.	D200037028	91
PREPROCESSOR	64815S003	01.30	Preprocessor reports errors when symbols hp64000, vms or hpux w #if	D200058784	94

- 6805/9 ASSEMB -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64844S004	00.00	Linker output file should use alternate file extension.	D200049288	96
	64844S004	01.00	Macro def. including .IF, within a IF causes assembler to stop code gen.	D200053397	95
MACRO	64844S004	01.00	Conditional instr. .IF with rational oper. in Macro creates bad code	D200048306	95

- 6805/9 ASSEMB -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64844S001	00.00	Linker output file should use alternate file extension.	D200049262	100
	64844S001	01.10	Passing an undefined parameter to a macro is not flagged as an error.	5000115097	97
	64844S001	01.20	Variable declared BEXT generates incorrect record in absolute file.	D200038273	97
	64844S001	01.20	Assembler should denote an error on non-absolute .SET expressions.	D200046896	98
	64844S001	01.30	Macro def. including .IF, within a IF causes assembler to stop code gen.	D200053371	99
	64844S001	01.30	Relative address is calculated incorrectly when macro call has null parm	D200055939	99
MACRO	64844S001	01.30	Conditional instr. .IF with rational oper. in Macro creates bad code	D200048280	98

Keyword index

- 6805/9 ASSEMB -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64844S003	00.00	Linker output file should use alternate file extension.	D200049270	103
	64844S003	01.20	Variable declared BEXT generates incorrect record in absolute file.	D200038281	101
	64844S003	01.20	Assembler should denote an error on non-absolute .SET expressions.	D200046904	101
	64844S003	01.40	Macro def. including .IF, within a IF causes assembler to stop code gen.	D200053389	102
MACRO	64844S003	01.40	Conditional instr. .IF with rational oper. in Macro creates bad code	D200048298	102

- 6809 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64822	01.04	No form feed between the expanded listing and the cross reference table.	D200027748	104
	64822	01.04	File fails to compile. Error 1113 is generated.	D200029694	104
	64822	01.04	++ and -- operators evaluated with improper precedence.	D200031419	105
	64822	01.04	Comparing character to zero in while loop generates incorrect code.	D200032391	105
	64822	01.05	Problem with integer pointer in conditional statement.	D200041327	107
	64822	01.05	DIFFERENT BUT EQUAL OBJECT CODE GENERATED ON 64000 THAN IN THE UNIX ENV.	D200045245	107
	64822	01.05	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047605	108
CODE GENERATOR	64822	01.04	16 bit comparison on a 8 bit unsigned short field.	D200035865	106
PASS 1	64822	00.56	No warning or err: taking the sizeof a struct var. not declared.	D200013946	104
PASS 3	64822	01.05	Pass 3 fails to detect relative jump address out-of-range.	D200040758	107

- 6809 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64822S004	00.00	Linker output file should use alternate file extension.	D200049015	110
	64822S004	01.00	File fails to compile. Error 1113 is generated.	D200051078	109
	64822S004	01.00	++ and -- operators evaluated with improper precedence.	D200051292	109
	64822S004	01.00	Host compilers do not put absolute pats specifications in relocatables	D200059055	110
CODE GENERATOR	64822S004	00.00	Incorrect opcode "MOV A,ACC" allowed by our assembler	D200052290	110
PASS 1	64822S004	01.00	Incorrect code is generated when complementing a parm. in a return stmt.	D200050278	109

- 6809 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64822S001	00.00	NO CROSS REFERENCE TABLE IS GENERATED	D200049742	111

- 6809 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64822S003	00.00	Problem with integer pointer in conditional statement.	D200041343	114
	64822S003	00.00	Title description is incorrect.	D200045989	114
	64822S003	00.00	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047621	114
	64822S003	00.00	Linker output file should use alternate file extension.	D200049007	116
	64822S003	01.00	File fails to compile. Error 1113 is generated.	D200029710	112
	64822S003	01.20	++ and -- operators evaluated with improper precedence.	D200051284	114
	64822S003	01.20	Compilation on the VAX using batch mode generates incorrect listing file	D200055160	115
	64822S003	01.20	Host compilers do not put absolute pats specifications in relocatables	D200059048	116
CODE GENERATOR	64822S003	00.00	16 bit comparison on a 8 bit unsigned short field.	D200035881	112
PASS 1	64822S003	01.00	Incorrect code is generated when complementing a parm. in a return stmt.	D200015651	112
PASS 3	64822S003	00.00	Compiler option \$LIST OBJ ON\$ generates wrong output information.	D200037143	113
	64822S003	00.00	Pass 3 fails to detect relative jump address out-of-range.	D200040774	113

Keyword index

- 6809 PASCAL -

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64813	01.08 DIFFERENT BUT EQUAL OBJECT CODE GENERATED ON 64000 THAN IN THE UNIX ENV.	D200045237	119
	64813	01.08 TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047365	119
	64813	01.09 Missing semicolon causes compiler to hang in Pass 1.	D200052480	119
CODE GENERATOR	64813	01.08 SHIFT funct. used as an array reference creates incorrect code.	5000114777	117
	64813	01.08 An automat. BYTE to INT. conversion within a WITH statmnt. - gen. bad cd	5000119925	118
ENHANCEMENT	64813	01.08 Superfluous code generated for bounds checking in FOR loop with consts.	5000096594	117
INCLUDE	64813	01.08 Nested INCLUDE files 3 or more deep cause 64000 to "hang" in pass 3.	D200036772	118

- 6809 PASCAL -

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64813S004	00.00 Linker output file should use alternate file extension.	D200048777	122
	64813S004	01.00 Missing semicolon causes compiler to hang in Pass 1.	D200052514	121
	64813S004	01.00 Host compilers do not put absolute pats specifications in relocatables	D200059162	122
CODE GENERATOR	64813S004	01.00 SHIFT funct. used as an array reference creates incorrect code.	D200048660	121
PREPROCESSOR	64813S004	01.00 Preprocessor reports errors when symbols hp64000, vms or hpux w #if	D200058735	122

- 6809 PASCAL -

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64813S001	00.00 Linker output file should use alternate file extension.	D200048751	125
	64813S001	01.00 TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047373	124
	64813S001	01.10 Missing semicolon causes compiler to hang in Pass 1.	D200052498	124
	64813S001	01.10 Host compilers do not put absolute pats specifications in relocatables	D200059147	125
CODE GENERATOR	64813S001	01.10 SHIFT funct. used as an array reference creates incorrect code.	D200048645	124
ENHANCEMENT	64813S001	01.00 Superfluous code generated for bounds checking in FOR loop with consts.	D200034181	123
PASS 3	64813S001	01.00 Compiler option \$LIST_OBJ ON\$ generates wrong output information.	D200036988	123
PREPROCESSOR	64813S001	01.10 Preprocessor reports errors when symbols hp64000, vms or hpux w #if	D200058719	125

- 6809 PASCAL -

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64813S003	00.00 Linker output file should use alternate file extension.	D200048769	128
	64813S003	01.00 COMPILER ASSIGNS INCORRECT TEMP STORAGE SOMETIMES BYTE TO REAL.	D200043372	127
	64813S003	01.00 TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047381	127
	64813S003	01.10 Missing semicolon causes compiler to hang in Pass 1.	D200052506	127
	64813S003	01.10 Host compilers do not put absolute pats specifications in relocatables	D200059154	128
CODE GENERATOR	64813S003	01.10 SHIFT funct. used as an array reference creates incorrect code.	D200048652	127
ENHANCEMENT	64813S003	01.00 Superfluous code generated for bounds checking in FOR loop with consts.	D200034199	126
PASS 3	64813S003	01.00 Compiler option \$LIST_OBJ ON\$ generates wrong output information.	D200036996	126
PREPROCESSOR	64813S003	01.10 Preprocessor reports errors when symbols hp64000, vms or hpux w #if	D200058727	128

- 8085 B PASCAL -

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64825	00.00 Incorrect code generated for WHILE construct.	2700005900	130
	64825	01.01 Defining TRUE and FALSE as global may result in duplicate symbol names.	D200026500	131
	64825	01.01 Bad code generated for assignment statement.	D200037796	133
	64825	01.01 Bad code generated for IF.. statement (including WITH).	D200041145	134
	64825	01.01 TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047696	134

Keyword index

- 8085 B PASCAL -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64825	01.02	Incorrect code generated when a CHAR is converted to an UNSIGNED_16.	D200052381	134
	64825	01.02	Missing semicolon causes compiler to hang in Pass 1.	D200052670	135
CODE GENERATOR	64825	01.01	Incorrect code generated for IF statement.	D200022434	130
	64825	01.01	Incorrect code generated for SET inclusion statement.	D200022491	131
FOR LOOP	64825	01.01	FOR Signed8 := 0 TO 2 DO REAL1 := REAL1/REAL2 overwrites the A-register.	D200044735	134
INCLUDE	64825	01.01	Nested INCLUDE files 3 or more deep cause 64000 to "hang" in pass 3.	D200036814	132
PASS 2	64825	01.01	Program re-BOOTS 64000 station.	D200019307	130
SETS	64825	01.01	SUPERSET or SUBSET checking doesn't work.	D200040261	133
STRING	64825	01.01	Pointers to STRINGS cannot be assigned a string of length one.	D200034157	132
STRING ARRAYS	64825	01.01	Multidimensional arrays of packed string arrays cannot be assigned to.	D200020131	130

- 8085 B PASCAL -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64825S004	00.00	Linker output file should use alternate file extension.	D200049106	139
	64825S004	01.00	Bad code generated for IF.. statement (including WITH).	D200052084	137
	64825S004	01.00	Incorrect code generated when a CHAR is converted to an UNSIGNED_16.	D200052415	137
	64825S004	01.00	Missing semicolon causes compiler to hang in Pass 1.	D200052704	138
	64825S004	01.00	Host compilers do not put absolute pats specifications in relocatables	D200059287	139
PREPROCESSOR	64825S004	01.00	Preprocessor reports errors when symbols hp64000, vms or hpux w #if	D200058883	139

- 8085 B PASCAL -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64825S001	00.00	Linker output file should use alternate file extension.	D200049080	147
	64825S001	01.10	Defining TRUE and FALSE as global may result in duplicate symbol names.	D200026518	142
	64825S001	01.10	No form feed between the expanded listing and the cross reference table.	D200027789	142
	64825S001	01.10	Incorrect code generated for WHILE construct.	D200028852	142
	64825S001	01.20	Bad code generated for assignment statement.	D200037804	144
	64825S001	01.20	Bad code generated for IF.. statement (including WITH).	D200041749	145
	64825S001	01.20	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047704	145
	64825S001	01.30	Incorrect code generated when a CHAR is converted to an UNSIGNED_16.	D200052399	145
	64825S001	01.30	Missing semicolon causes compiler to hang in Pass 1.	D200052688	146
	64825S001	01.30	Host compilers do not put absolute pats specifications in relocatables	D200059261	147
CODE GENERATOR	64825S001	01.10	Incorrect code generated for IF statement.	D200022442	140
	64825S001	01.10	Incorrect code generated for SET inclusion statement.	D200022509	141
FOR LOOP	64825S001	01.20	FOR Signed8 := 0 TO 2 DO REAL1 := REAL1/REAL2 overwrites the A-register.	D200044743	145
PASS 2	64825S001	01.10	Array element as argument of CASE statement causes compile to fail.	5000107888	140
PASS 3	64825S001	01.20	Compiler option \$LIST_OBJ_ON\$ generates wrong output information.	D200037192	143
PREPROCESSOR	64825S001	01.30	Preprocessor reports errors when symbols hp64000, vms or hpux w #if	D200058867	147
SETS	64825S001	01.20	SUPERSET or SUBSET checking doesn't work.	D200040279	144
STRING	64825S001	01.10	Pointers to STRINGS cannot be assigned a string of length one.	D200034165	143
STRING ARRAYS	64825S001	01.10	Multidimensional arrays of packed string arrays cannot be assigned to.	D200020149	140

- 8085 B PASCAL -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64825S003	00.00	Linker output file should use alternate file extension.	D200049098	155
	64825S003	01.10	Defining TRUE and FALSE as global may result in duplicate symbol names.	D200026526	149
	64825S003	01.20	No form feed between the expanded listing and the cross reference table.	D200027797	150
	64825S003	01.20	Incorrect code generated for WHILE construct.	D200028860	150

Keyword index

- 8085 B PASCAL -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64825S003	01.20	Bad code generated for assignment statement.	D200037812	151
	64825S003	01.20	Bad code generated for IF.. statement (including WITH).	D200041756	152
	64825S003	01.20	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047712	153
	64825S003	01.50	Incorrect code generated when a CHAR is converted to an UNSIGNED_16.	D200052407	153
	64825S003	01.50	Missing semicolon causes compiler to hang in Pass 1.	D200052696	154
CODE GENERATOR	64825S003	01.50	Host compilers do not put absolute pats specifications in relocatables	D200059279	155
	64825S003	01.10	Incorrect code generated for IF statement.	D200022459	148
	64825S003	01.10	Incorrect code generated for SET inclusion statement.	D200022517	149
FOR LOOP	64825S003	01.20	FOR Signed8 := 0 TO 2 DO REAL1 := REAL1/REAL2 overwrites the A-register.	D200044750	152
PASS 3	64825S003	01.20	Compiler option \$LIST_OBJ ON\$ generates wrong output information.	D200037200	151
PREPROCESSOR	64825S003	01.50	Preprocessor reports errors when symbols hp64000, vms or hpux w #if	D200058875	154
SETS	64825S003	01.20	SUPERSET or SUBSET checking doesn't work.	D200040287	152
STRING	64825S003	01.20	Pointers to STRINGS cannot be assigned a string of length one.	D200034173	150
STRING ARRAYS	64825S003	01.10	Multidimensional arrays of packed string arrays cannot be assigned to.	D200020156	148

- 8085 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64826	01.01	Incorrect code gen by assignment to deref'd 8 bit field of structure.	D200026781	157
	64826	01.01	No form feed between the expanded listing and the cross reference table.	D200027805	158
	64826	01.01	Addition of dereferenced pointers to structures may fail.	D200027912	158
	64826	01.01	++ and -- operators evaluated with improper precedence.	D200031104	159
	64826	01.01	Comparing character to zero in while loop generates incorrect code.	D200033258	159
	64826	01.01	Run time UNDERFLOW error using ZDSBSUB library if result has even parity	D200037465	161
	64826	01.01	Problem with integer pointer in conditional statement.	D200041376	162
	64826	01.01	Post increment of pointer results in incorrect code.	D200046037	163
	64826	01.01	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047720	163
	64826	01.02	Function return address is incorrect and program returns to wrong place.	5000135780	156
	64826	01.02	Incorrect code for multiplication dependent on order of operands.	D200053777	163
	64826	01.02	Compiler loses track of array index.	D200055277	164
CODE GENERATOR	64826	01.01	Dereferenced and incremented 2nd field of structure fails when parameter	D200025387	156
	64826	01.01	A shift assignment operation (<<=) generates incorrect code.	D200034298	160
	64826	01.01	16 bit comparison on a 8 bit unsigned short field.	D200035923	160
PASS 1	64826	01.01	No warning or error: taking the sizeof a struct var. not declared.	D200013995	156
PASS 3	64826	01.01	Pass 3 fails to detect relative jump address out-of-range.	D200040816	162

- 8085 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64826S004	00.00	Linker output file should use alternate file extension.	D200049130	168
	64826S004	01.00	Defining TRUE and FALSE as global may result in duplicate symbol names.	D200050757	166
	64826S004	01.00	++ and -- operators evaluated with improper precedence.	D200051318	166
	64826S004	01.00	Run time UNDERFLOW error using ZDSBSUB library if result has even parity	D200052001	166
	64826S004	01.00	Compiler loses track of array index.	D200055293	167
	64826S004	01.00	Host compilers do not put absolute pats specifications in relocatables	D200059113	168

- 8085 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64826S001	00.00	Linker output file should use alternate file extension.	D200049114	177
	64826S001	00.00	NO CROSS REFERENCE TABLE IS GENERATED	D200049809	176

Keyword index

- 8085 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64826S001	01.10	Incorrect code gen by assignment to deref'd 8 bit field of structure.	D200027011	169
	64826S001	01.10	Addition of dereferenced pointers to structures may fail.	D200027920	170
	64826S001	01.10	++ and -- operators evaluated with improper precedence.	D200031450	171
	64826S001	01.10	Comparing character to zero in while loop generates incorrect code.	D200033266	171
	64826S001	01.20	Run time UNDERFLOW error using ZDSBSUB library if result has even parity	D200040618	174
	64826S001	01.20	Problem with integer pointer in conditional statement.	D200041384	175
	64826S001	01.20	Title description is incorrect.	D200046011	175
	64826S001	01.20	Post increment of pointer results in incorrect code.	D200046201	175
	64826S001	01.20	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047738	176
	64826S001	01.40	Compiler loses track of array index.	D200055251	176
	64826S001	01.40	Host compilers do not put absolute pats specifications in relocatables	D200059097	177
CODE GENERATOR	64826S001	01.10	Dereferenced and incremented 2nd field of structure fails when parameter	D200025692	169
	64826S001	01.10	A shift assignment operation (<<=) generates incorrect code.	D200034306	172
	64826S001	01.10	16 bit comparison on an 8 bit unsigned short field.	D200035931	172
PASS 3	64826S001	01.20	Compiler option \$LIST OBJ ON\$ generates wrong output information.	D200037218	173
	64826S001	01.20	Pass 3 fails to detect relative jump address out-of-range.	D200040824	174

- 8085 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64826S003	00.00	Linker output file should use alternate file extension.	D200049122	187
	64826S003	01.20	Incorrect code gen by assignment to deref'd 8 bit field of structure.	D200027029	178
	64826S003	01.20	Addition of dereferenced pointers to structures may fail.	D200027938	179
	64826S003	01.20	++ and -- operators evaluated with improper precedence.	D200031468	180
	64826S003	01.20	Comparing character to zero in while loop generates incorrect code.	D200033274	180
	64826S003	01.20	Run time UNDERFLOW error using ZDSBSUB library if result has even parity	D200040626	183
	64826S003	01.20	Problem with integer pointer in conditional statement.	D200041392	184
	64826S003	01.20	Title description is incorrect.	D200046029	184
	64826S003	01.20	Post increment of pointer results in incorrect code.	D200046219	184
	64826S003	01.20	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047746	185
	64826S003	01.60	Compilation on the VAX using batch mode generates incorrect listing file	D200055186	185
	64826S003	01.60	Compiler loses track of array index.	D200055285	186
	64826S003	01.60	Host compilers do not put absolute pats specifications in relocatables	D200059105	187
CODE GENERATOR	64826S003	01.10	Dereferenced and incremented 2nd field of structure fails when parameter	D200025700	178
	64826S003	01.20	A shift assignment operation (<<=) generates incorrect code.	D200034314	181
	64826S003	01.20	16 bit comparison on a 8 bit unsigned short field.	D200035949	181
PASS 3	64826S003	01.20	Compiler option \$LIST OBJ ON\$ generates wrong output information.	D200037226	182
	64826S003	01.20	Pass 3 fails to detect relative jump address out-of-range.	D200040832	183

- 8086/8 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64818	01.06	No error when illegal assignment to a pointer is made.	D200026427	188
	64818	02.00	ASM file created by compiler generates errors when assembled.	5000103218	188
	64818	02.00	No form feed between the expanded listing and the cross reference table.	D200027706	189
	64818	02.00	++ and -- operators evaluated with improper precedence.	D200031294	189
	64818	02.00	Comparing character to zero in while loop generates incorrect code.	D200033100	189
	64818	02.00	Problem with integer pointer in conditional statement.	D200041194	191
	64818	02.00	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047480	192
	64818	03.00	ES pushed instead of DS when POINTER SIZE = 32.	D200049841	192
CODE GENERATOR	64818	02.00	16 bit comparison on a 8 bit unsigned short field.	D200035782	190
PASS 1	64818	01.06	No warning or error: taking the sizeof a struct var. not declared.	D200013961	188

Keyword index

- 8086/8 C -

Keyword	Product number	uu.ff	Description	Report #	page
PASS 3	64818	02.00	Pass 3 fails to detect relative jump address out-of-range.	D200040634	191

- 8086/8 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64818S004	00.00	Linker output file should use alternate file extension.	D200048892	194
	64818S004	03.00	ES pushed instead of DS when POINTER SIZE = 32.	D200049874	193
	64818S004	03.00	++ and -- operators evaluated with improper precedence.	D200051235	193
	64818S004	03.00	Host compilers do not put absolute pats specifications in relocatables	D200058933	194
CODE GENERATOR	64818S004	00.00	Incorrect opcode "MOV A,ACC" allowed by our assembler	D200052258	193

- 8086/8 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64818S001	00.00	Linker output file should use alternate file extension.	D200048876	200
	64818S001	00.00	NO CROSS REFERENCE TABLE IS GENERATED	D200049635	199
	64818S001	01.10	No error when illegal assignment to a pointer is made.	D200026666	195
	64818S001	01.20	NULL CHARACTERS IN ASM SOURCE PRODUCED WITH \$ASM_FILES\$	D200046276	198
	64818S001	02.00	++ and -- operators evaluated with improper precedence.	D200031302	195
	64818S001	02.00	Comparing character to zero in while loop generates incorrect code.	D200033118	195
	64818S001	02.01	Problem with integer pointer in conditional statement.	D200041202	198
	64818S001	02.01	Title description is incorrect.	D200045906	198
	64818S001	02.01	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047498	199
	64818S001	03.10	ES pushed instead of DS when POINTER SIZE = 32.	D200049858	199
	64818S001	03.10	Host compilers do not put absolute pats specifications in relocatables	D200058917	199
CODE GENERATOR	64818S001	02.00	16 bit comparison on a 8 bit unsigned short field.	D200035790	196
PASS 3	64818S001	02.01	Compiler option \$LIST OBJ ON\$ generates wrong output information.	D200037051	197
	64818S001	02.01	Pass 3 fails to detect relative jump address out-of-range.	D200040642	198

- 8086/8 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64818S003	00.00	Linker output file should use alternate file extension.	D200048884	206
	64818S003	01.10	No error when illegal assignment to a pointer is made.	D200026674	201
	64818S003	02.00	++ and -- operators evaluated with improper precedence.	D200031310	201
	64818S003	02.00	Comparing character to zero in while loop generates incorrect code.	D200033126	201
	64818S003	02.00	Problem with integer pointer in conditional statement.	D200041210	204
	64818S003	02.00	Title description is incorrect.	D200045914	204
	64818S003	02.00	NULL CHARACTERS IN ASM SOURCE PRODUCED WITH \$ASM_FILES\$	D200046607	204
	64818S003	02.00	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047506	205
	64818S003	03.10	ES pushed instead of DS when POINTER SIZE = 32.	D200049866	205
	64818S003	03.10	Compilation on the VAX using batch mode generates incorrect listing file	D200055129	205
	64818S003	03.10	Host compilers do not put absolute pats specifications in relocatables	D200058925	206
CODE GENERATOR	64818S003	02.00	16 bit comparison on a 8 bit unsigned short field.	D200035808	202
PASS 3	64818S003	02.00	Compiler option \$LIST OBJ ON\$ generates wrong output information.	D200037069	203
	64818S003	02.00	Pass 3 fails to detect relative jump address out-of-range.	D200040659	204

Keyword index

- 8086/8 PASCAL -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64814	01.10	Only two bytes of a three byte array are passed correctly as parameters.	D200015230	207
	64814	02.00	Param of WRITELN not separated by , 's cause compiler to abort.	5000118828	207
	64814	02.01	Bad "machine" code generated for LEA assembly instruction.	D200037234	208
	64814	02.01	Incorrect machine code generated for LEA ... instruction.	D200038950	208
	64814	02.01	Error 1102: register needed but not available.	D200046631	208
	64814	02.01	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047399	208
	64814	03.00	Missing semicolon causes compiler to hang in Pass 1.	D200052522	208
CODE GENERATOR	64814	03.00	Register needed but not available	D200053728	210
	64814	03.00	Width option causes 64000 to enter PV during compilation	D200053181	209
	64814	03.00	Variable addresses calculated incorrectly	D200053736	211
INCLUDE	64814	02.01	Nested INCLUDE files 3 or more deep cause 64000 to "hang" in pass 3.	D200036780	208

- 8086/8 PASCAL -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64814S004	00.00	Linker output file should use alternate file extension.	D200048801	213
	64814S004	03.00	Missing semicolon causes compiler to hang in Pass 1.	D200052555	212
PREPROCESSOR	64814S004	03.00	Host compilers do not put absolute pats specifications in relocatables	D200059196	212
	64814S004	03.00	Preprocessor reports errors when symbols hp64000, vms or hpux w #if	D200058768	212

- 8086/8 PASCAL -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64814S001	00.00	Linker output file should use alternate file extension.	D200048785	216
	64814S001	01.30	NULL CHARACTERS IN ASM SOURCE PRODUCED WITH \$ASM_FILE\$	D200046318	215
	64814S001	02.00	No form feed between the expanded listing and the cross reference table.	D200027649	214
	64814S001	02.00	Bad "machine" code generated for LEA assembly instruction.	D200037291	215
	64814S001	02.00	Error 1102: register needed but not available.	D200046748	215
	64814S001	02.00	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047407	215
	64814S001	03.00	Missing semicolon causes compiler to hang in Pass 1.	D200052530	215
PASS 3	64814S001	03.00	Host compilers do not put absolute pats specifications in relocatables	D200059170	216
PREPROCESSOR	64814S001	02.00	Compiler option \$LIST_OBJ ON\$ generates wrong output information.	D200036871	214
	64814S001	03.00	Preprocessor reports errors when symbols hp64000, vms or hpux w #if	D200058743	216

- 8086/8 PASCAL -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64814S003	00.00	Linker output file should use alternate file extension.	D200048793	219
	64814S003	02.00	No form feed between the expanded listing and the cross reference table.	D200027656	217
	64814S003	02.00	Bad "machine" code generated for LEA assembly instruction.	D200037309	218
	64814S003	02.00	NULL CHARACTERS IN ASM SOURCE PRODUCED WITH \$ASM_FILE\$	D200046615	218
	64814S003	02.00	Error 1102: register needed but not available.	D200046755	218
	64814S003	02.00	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047415	218
	64814S003	03.00	Missing semicolon causes compiler to hang in Pass 1.	D200052548	218
	64814S003	03.00	Host compilers do not put absolute pats specifications in relocatables	D200059188	219
PASS 3	64814S003	02.00	Compiler option \$LIST_OBJ ON\$ generates wrong output information.	D200037002	217
PREPROCESSOR	64814S003	03.00	Preprocessor reports errors when symbols hp64000, vms or hpux w #if	D200058750	219

Keyword index

- F9450 EMULATION -

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64286	01.02 Intermittent PV failures occur on test 8 (IO Cycles)	D200060301	220

- OP_SYS DEC-VAX / VMS -

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64882	01.20 Mapbus output is "hardwired" to the system console.	D200046110	222
	64882	01.20 Debug transfers will not work when '.PAS' file extensions are used.	D200046144	222
	64882	01.60 REMOTE CONTROL HP6400 LOCKING MECHANISM WAS MADE MORE RELIABLE	D200053884	224
	64882	01.60 Foreground signal can kill a background batch remote control job.	D200053892	223
	64882	01.60 Hp 64000 exit message is not outputted for exits when needed	D200053900	223
HIGH SPEED LINK	64882	01.20 TRANSFER/H/A/T from anACL controlled directory does not work.	D200043935	221
	64882	01.20 File list transfers may not work under certain conditions.	D200045054	221
	64882	01.20 The HPIB configuration on the OPA0: doesn't contain line-feeds.	D200047969	222
	64882	01.20 A CSIB with a pending MAPBUS, changes priority from 12 to 14 and back.	D200047985	222
	64882	01.20 High speed link transfer does not work from passworded userids.	D200048025	223
TRANSFER	64882	01.20 The wrong protection can be left on HSL0.DAT when MAPBUS completes.	D200043570	221
	64882	01.20 TRANSFER/H/A/T from anACL controlled directory does not work.	D200043935	221
	64882	01.60 Certain length filename.extension's will not transfer.	D200053819	223

- OP_SYS HP-UX / 500 -

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64880	01.20 High Speed Link transfer can remove files from protected directories.	D200043588	225
	64880	01.50 REMOTE CONTROL HP6400 LOCKING MECHANISM WAS MADE MORE RELIABLE	D200054312	226
	64880	01.50 Foreground signal can kill a background batch remote control job.	D200054320	225
	64880	01.50 Hp 64000 exit message is not outputted for exits when needed	D200054338	225
	64880	01.50 An escaped shell from the menu can return prematurely	D200054346	225
	64880	01.50 Problem with make utility.	D200060269	226
	64880	01.50 Problems with the linker listing file and map.	D200060277	226
LINKER	64880	01.30 Linker is VERY "picky" about the use of file extensions.	5000124040	226

- USER DEF ASSEMB -5

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64851S001	01.20 Assemblers should denote an error on non-absolute .SET expressions.	D200047019	228
	64851S001	01.20 Assembler flags error on host but NOT on 64000.	D200048066	228
	64851S001	01.30 Macro def. including .IF, within a IF causes assembler to stop code gen.	D200053496	228
	64851S001	01.40 Comments not delimited by semi-colons appear in the assembler xref.	D200055525	229
	64851S001	01.40 Host compilers do not put absolute pats specifications in relocatables	D200059295	229
	64851S001	01.40 QUOTING CHARACTERS WITHIN STRINGS ARE ALL TRANSLATED TO "."	D200059949	229
LINKER	64851S001	00.00 LINKER WILL NOT LINK FILENAMES STARTING WITH A NUMBER	D200042044	228

- USER DEF ASSEMB -V

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64851S003	00.00 Linker output file should use alternate file extension.	D200049395	233
	64851S003	01.10 Code generated differs from code generated on HP 64000.	D200019877	230
	64851S003	01.20 Assembler should denote an error on non-absolute .SET expressions.	D200047027	230
	64851S003	01.40 Macro def. including .IF, within a IF causes assembler to stop code gen.	D200053504	231
	64851S003	01.40 Comments not delimited by semi-colons appear in the assembler xref.	D200055533	232

Keyword index

- USER DEF ASSEMB -V

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64851S003	01.40 Host compilers do not put absolute pats specifications in relocatables	D200059303	232
	64851S003	01.40 PROBLEMS WHEN USING "FDB" OR "FCB" WITH A STRING	D200059410	232
	64851S003	01.40 QUOTING CHARACTERS WITHIN STRINGS ARE ALL TRANSLATED TO "."	D200059956	233
MACRO	64851S003	01.20 string comparison does not function using conditional .if instr.	1650006536	230
	64851S003	01.40 Conditional instr. .IF with rational oper. in Macro creates bad code	D200048413	231

- Z80 ASSEMB -

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64842	00.01 Legal range error is flagged when .NT. logical operator is used.	D200033407	234
	64842	00.01 No error flagged when illegal 16 bit addition is preformed.	D200036509	234
	64842	00.01 Assembler should denote an error on non-absolute .SET expressions.	D200046821	234
	64842	01.11 Z80 assembler allowing illegal instructions.	5000132720	234

- Z80 ASSEMB -

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64842S004	00.00 Linker output file should use alternate file extension.	D200049221	237
	64842S004	01.00 Z80 assembler allowing illegal instructions.	D200053215	236
	64842S004	01.00 Macro def. including .IF, within a IF causes assembler to stop code gen.	D200053330	236
MACRO	64842S004	01.00 Conditional instr. .IF with rational oper. in Macro creates bad code	D200048249	236

- Z80 ASSEMB -

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64842S001	00.00 Linker output file should use alternate file extension.	D200049205	239
	64842S001	01.20 Assembler should denote an error on non-absolute .SET expressions.	D200046839	238
	64842S001	01.30 Z80 assembler allowing illegal instructions.	D200053199	238
	64842S001	01.30 Macro def. including .IF, within a IF causes assembler to stop code gen.	D200053322	239
MACRO	64842S001	01.30 Conditional instr. .IF with rational oper. in Macro creates bad code	D200048223	238

- Z80 ASSEMB -

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64842S003	00.00 Linker output file should use alternate file extension.	D200049213	241
	64842S003	01.20 Assembler should denote an error on non-absolute .SET expressions.	D200046847	240
	64842S003	01.30 Macro def. including .IF, within a IF causes assembler to stop code gen.	5000121178	240
	64842S003	01.40 Z80 assembler allowing illegal instructions.	D200053207	241
MACRO	64842S003	01.40 Conditional instr. .IF with rational oper. in Macro creates bad code	D200048231	240

- Z80/NSC800 C -

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64824	01.01 Incorrect code gen by assignment to deref'd 8 bit field of structure.	D200026989	243
	64824	01.01 Incorrect code for switch on dereferenced non-integer structure element.	D200027458	243
	64824	01.01 No form feed between the expanded listing and the cross reference table.	D200027771	244
	64824	01.01 Addition of dereferenced pointers to structures may fail.	D200027888	244
	64824	01.01 Incorrect code when indexing into an array passed as a parameter.	D200028746	245
	64824	01.01 Dereferencing pointers to structures in assignment statements may fail.	D200028779	246

Keyword index

- Z80/NSC800 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64824	01.01	++ and -- operators evaluated with improper precedence.	D200031427	246
	64824	01.01	Comparing character to zero in while loop generates incorrect code.	D200033225	247
	64824	01.01	Problem with integer pointer in conditional statement.	D200041186	249
	64824	01.01	STACK POINTER OFFSETS ARE INCORRECT WHEN ENTERING REAL TRUNC.	D200043596	250
	64824	01.01	Illegal forward reference error generated when initializing structures.	D200043968	250
	64824	01.01	Stack offset to parameter is incorrect.	D200044685	250
	64824	01.01	Conditional containing 'pointer to func' is not calling correct func.	D200045518	251
	64824	01.01	Character being sign converted to a word causing conditional to be false	D200045526	251
	64824	01.01	Updating & assigning ptr a new value causes compiler to genera	D200045872	252
	64824	01.01	Post increment of pointer results in incorrect code.	D200046177	252
	64824	01.01	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047662	253
CODE GENERATOR	64824	01.01	Dereferenced and incremented 2nd field of structure fails when parameter	D200025668	242
	64824	01.01	A shift assignment operation (<<=) generates incorrect code.	D200034264	247
	64824	01.01	16 bit comparison on a 8 bit unsigned short field.	D200035899	248
PASS 1	64824	01.01	No warning or error: taking the sizeof a struct var. not declared.	D200013987	242
PASS 3	64824	01.01	Pass 3 fails to detect relative jump address out-of-range.	D200040782	249

- Z80/NSC800 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64824S004	00.00	Linker output file should use alternate file extension.	D200049072	255
	64824S004	01.00	Defining TRUE and FALSE as global may result in duplicate symbol names.	D200050740	254
	64824S004	01.00	++ and -- operators evaluated with improper precedence.	D200051300	254
	64824S004	01.00	Host compilers do not put absolute pats specifications in relocatables	D200059089	255
CODE GENERATOR	64824S004	00.00	Incorrect opcode "MOV A,ACC" allowed by our assembler	D200052308	254

- Z80/NSC800 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64824S001	00.00	Linker output file should use alternate file extension.	D200049056	265
	64824S001	00.00	NO CROSS REFERENCE TABLE IS GENERATED	D200049775	264
	64824S001	01.10	Incorrect code gen by assignment to deref'd 8 bit field of structure.	D200026997	256
	64824S001	01.10	Addition of dereferenced pointers to structures may fail.	D200027896	257
	64824S001	01.10	Incorrect code when indexing into an array passed as a parameter.	D200028753	258
	64824S001	01.10	Dereferencing pointers to structures in assignment statements may fail.	D200029223	258
	64824S001	01.10	++ and -- operators evaluated with improper precedence.	D200031435	259
	64824S001	01.10	Comparing character to zero in while loop generates incorrect code.	D200033233	259
	64824S001	01.20	Problem with integer pointer in conditional statement.	D200041350	262
	64824S001	01.20	Title description is incorrect.	D200045997	263
	64824S001	01.20	Updating & assigning ptr a new value causes compiler to genera	D200046078	263
	64824S001	01.20	Post increment of pointer results in incorrect code.	D200046185	264
	64824S001	01.20	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047670	264
	64824S001	01.40	Host compilers do not put absolute pats specifications in relocatables	D200059063	265
CODE GENERATOR	64824S001	01.10	Dereferenced and incremented 2nd field of structure fails when parameter	D200025676	256
	64824S001	01.10	A shift assignment operation (<<=) generates incorrect code.	D200034272	260
	64824S001	01.10	16 bit comparison on a 8 bit unsigned short field.	D200035907	260
PASS 3	64824S001	01.20	Compiler option \$LIST_OBJ ON\$ generates wrong output information.	D200037176	261
	64824S001	01.20	Pass 3 fails to detect relative jump address out-of-range.	D200040790	262

Keyword index

- Z80/NSC800 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64824S003	00.00	Linker output file should use alternate file extension.	D200049064	275
	64824S003	01.20	Incorrect code gen by assignment to deref'd 8 bit field of structure.	D200027003	266
	64824S003	01.20	Addition of dereferenced pointers to structures may fail.	D200027904	267
	64824S003	01.20	Incorrect code when indexing into an array passed as a parameter.	D200028761	268
	64824S003	01.20	Dereferencing pointers to structures in assignment statements may fail.	D200029215	268
	64824S003	01.20	++ and -- operators evaluated with improper precedence.	D200031443	269
	64824S003	01.20	Comparing character to zero in while loop generates incorrect code.	D200033241	269
	64824S003	01.20	Problem with integer pointer in conditional statement.	D200041368	272
	64824S003	01.20	Title description is incorrect.	D200046003	272
	64824S003	01.20	Updating & assigning ptr a new value causes compiler to genera	D200046086	273
	64824S003	01.20	Post increment of pointer results in incorrect code.	D200046193	273
	64824S003	01.20	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047688	274
	64824S003	01.50	Compilation on the VAX using batch mode generates incorrect listing file	D200055178	274
CODE GENERATOR	64824S003	01.50	Host compilers do not put absolute pats specifications in relocatables	D200059071	275
	64824S003	01.10	Dereferenced and incremented 2nd field of structure fails when parameter	D200025684	266
	64824S003	01.20	A shift assignment operation (<=>) generates incorrect code.	D200034280	270
	64824S003	01.20	16 bit comparison on a 8 bit unsigned short field.	D200035915	270
PASS 3	64824S003	01.20	Compiler option \$LIST OBJ ONS generates wrong output information.	D200037184	271
	64824S003	01.20	Pass 3 fails to detect relative jump address out-of-range.	D200040808	272

- Z80/NSC800PASCAL -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64823	01.01	Accessing parameter two nesting levels up is not working.	1650004630	276
	64823	01.01	Defining TRUE and FALSE as global may result in duplicate symbol names.	D200026419	280
	64823	01.01	Incorrect code generated for WHILE construct.	D200028878	280
	64823	01.01	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047639	281
	64823	01.01	Zcaseerror jumped to rather than called.	D200047944	281
	64823	01.01	Level 3 recursive procedure or function causes Error 1008 - Stack Error.	D200048074	282
	64823	01.01	Missing semicolon causes compiler to hang in Pass 1.	D200048116	283
	64823	01.02	Level 3 access of level 1 variables generates incorrect code.	D200049890	283
CODE GENERATOR	64823	01.02	Incorrect code generated when a CHAR is converted to an UNSIGNED_16.	D200052241	284
	64823	01.01	Incorrect code generated for IF statement.	D200022467	279
	64823	01.01	Incorrect code generated for SET inclusion statement.	D200022525	279
FOR LOOP	64823	01.01	FOR Signed8 := 0 TO 2 DO REAL1 := REAL1/REAL2 overwrites the A-register.	5000115402	278
INCLUDE	64823	01.01	Nested INCLUDE files 3 or more deep cause 64000 to "hang" in pass 3.	D200036806	281
PASS 3	64823	01.01	Pass 3 fails to detect relative jump address out-of-range.	D200016329	278
RECURSIVE	64823	01.01	FOR loops don't work with \$RECURSIVE +\$ and WITH.	5000109934	278
SETS	64823	01.01	SUPERSET or SUBSET checking doesn't work.	5000103267	277
STRING	64823	01.01	Pointers to STRINGS cannot be assigned a string of length one.	D200034108	280
STRING ARRAYS	64823	00.00	Multidimensional arrays of packed string arrays cannot be assigned to.	2700005371	277

- Z80/NSC800PASCAL 300 -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64823S004	00.00	Linker output file should use alternate file extension.	D200049049	289
	64823S004	01.00	Incorrect code generated when a CHAR is converted to an UNSIGNED_16.	D200052373	286
	64823S004	01.00	Missing semicolon causes compiler to hang in Pass 1.	D200052662	287
	64823S004	01.00	Accessing parameter two nesting levels up is not working.	D200053769	287
	64823S004	01.00	Host compilers do not put absolute pats specifications in relocatables	D200059253	289
PREPROCESSOR	64823S004	01.00	Preprocessor reports errors when symbols hp64000, vms or hpux w #if	D200058859	289

Keyword index

- Z80/NSC800PASCAL 500 -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64823S001	00.00	Linker output file should use alternate file extension.	D200049023	299
	64823S001	01.10	Defining TRUE and FALSE as global may result in duplicate symbol names.	D200026484	291
	64823S001	01.10	No form feed between the expanded listing and the cross reference table.	D200027755	292
	64823S001	01.10	Incorrect code generated for WHILE construct.	D200028886	292
	64823S001	01.20	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047647	295
	64823S001	01.20	Level 3 recursive procedure or function causes Error 1008 - Stack Error.	D200048090	295
	64823S001	01.30	Incorrect code generated when a CHAR is converted to an UNSIGNED_16.	D200052357	296
	64823S001	01.30	Missing semicolon causes compiler to hang in Pass 1.	D200052647	297
	64823S001	01.30	Accessing parameter two nesting levels up is not working.	D200053744	297
CODE GENERATOR	64823S001	01.30	Host compilers do not put absolute pats specifications in relocatables	D200059238	299
	64823S001	01.10	Incorrect code generated for IF statement.	D200022475	290
FOR LOOP	64823S001	01.10	Incorrect code generated for SET inclusion statement.	D200022533	291
PASS 3	64823S001	01.20	FOR Signed8 := 0 TO 2 DO REAL1 := REAL1/REAL2 overwrites the A-register.	D200044719	294
	64823S001	01.10	Pass 3 fails to detect relative jump address out-of-range.	D200016337	290
PREPROCESSOR	64823S001	01.20	Compiler option \$LIST_OBJ ON\$ generates wrong output information.	D200037150	293
RECURSIVE	64823S001	01.30	Preprocessor reports errors when symbols hp64000, vms or hpux w #if	D200058834	299
SETS	64823S001	01.20	FOR loops don't work with \$RECURSIVE +\$ and WITH.	D200043851	294
STRING	64823S001	01.20	SUPERSET or SUBSET checking doesn't work.	D200040246	294
STRING ARRAYS	64823S001	01.10	Pointers to STRINGS cannot be assigned a string of length one.	D200034132	292
	64823S001	01.10	Multidimensional arrays of packed string arrays cannot be assigned to.	D200020115	290

- Z80/NSC800PASCAL VAX -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64823S003	00.00	Linker output file should use alternate file extension.	D200049031	309
	64823S003	01.10	Defining TRUE and FALSE as global may result in duplicate symbol names.	D200026492	301
	64823S003	01.20	No form feed between the expanded listing and the cross reference table.	D200027763	302
	64823S003	01.20	Incorrect code generated for WHILE construct.	D200028894	302
	64823S003	01.20	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047654	305
	64823S003	01.20	Level 3 recursive procedure or function causes Error 1008 - Stack Error.	D200048108	305
	64823S003	01.40	Incorrect code generated when a CHAR is converted to an UNSIGNED_16.	D200052365	306
	64823S003	01.40	Missing semicolon causes compiler to hang in Pass 1.	D200052654	307
	64823S003	01.40	Accessing parameter two nesting levels up is not working.	D200053751	307
CODE GENERATOR	64823S003	01.40	Host compilers do not put absolute pats specifications in relocatables	D200059246	309
	64823S003	01.10	Incorrect code generated for IF statement.	D200022483	300
FOR LOOP	64823S003	01.10	Incorrect code generated for SET inclusion statement.	D200022541	301
PASS 3	64823S003	01.20	FOR Signed8 := 0 TO 2 DO REAL1 := REAL1/REAL2 overwrites the A-register.	D200044727	304
	64823S003	01.10	Pass 3 fails to detect relative jump address out-of-range.	D200016345	300
PREPROCESSOR	64823S003	01.20	Compiler option \$LIST_OBJ ON\$ generates wrong output information.	D200037168	303
RECURSIVE	64823S003	01.40	Preprocessor reports errors when symbols hp64000, vms or hpux w #if	D200058842	309
SETS	64823S003	01.20	FOR loops don't work with \$RECURSIVE +\$ and WITH.	D200043869	304
STRING	64823S003	01.20	SUPERSET or SUBSET checking doesn't work.	D200040253	304
STRING ARRAYS	64823S003	01.20	Pointers to STRINGS cannot be assigned a string of length one.	D200034140	302
	64823S003	01.10	Multidimensional arrays of packed string arrays cannot be assigned to.	D200020123	300

- Z8000 C -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64820	01.03	No form feed between the expanded listing and the cross reference table.	D200027722	310
	64820	01.03	++ and -- operators evaluated with improper precedence.	D200031351	310
	64820	01.03	Comparing character to zero in while loop generates incorrect code.	D200033167	311
	64820	01.03	Problem with integer pointer in conditional statement.	D200041251	312

Keyword index

- Z8000 C -

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64820	01.03 TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047548	312
PASS 1	64820	01.03 No warning or error: taking the sizeof a struct var. not declared	D200013979	310
PASS 3	64820	01.03 Pass 3 fails to detect relative jump address out-of-range.	D200040691	311

- Z8000 C -

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64820S004	00.00 Linker output file should use alternate file extension.	D200048959	314
	64820S004	01.00 ++ and -- operators evaluated with improper precedence.	D200051250	313
	64820S004	01.00 Host compilers do not put absolute pats specifications in relocatables	D200058990	313
CODE GENERATOR	64820S004	00.00 Incorrect opcode "MOV A,ACC" allowed by our assembler	D200052274	313

- Z8000 C -

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64820S001	00.00 Linker output file should use alternate file extension.	D200048934	318
	64820S001	00.00 NO CROSS REFERENCE TABLE IS GENERATED	D200049684	317
	64820S001	01.10 Program compiles on 64K, not 9000. Pass 3 error generated.	D200029728	315
	64820S001	01.10 ++ and -- operators evaluated with improper precedence.	D200031369	315
	64820S001	01.10 Comparing character to zero in while loop generates incorrect code.	D200033175	315
	64820S001	01.20 Problem with integer pointer in conditional statement.	D200041269	317
	64820S001	01.20 Title description is incorrect.	D200045930	317
	64820S001	01.20 TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047555	317
	64820S001	01.40 Host compilers do not put absolute pats specifications in relocatables	D200058974	318
PASS 3	64820S001	01.20 Compiler option \$LIST OBJ ON\$ generates wrong output information.	D200037093	316
	64820S001	01.20 Pass 3 fails to detect relative jump address out-of-range.	D200040709	316

- Z8000 C -

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64820S003	00.00 Linker output file should use alternate file extension.	D200048942	323
	64820S003	01.20 ++ and -- operators evaluated with improper precedence.	D200031377	319
	64820S003	01.20 Comparing character to zero in while loop generates incorrect code.	D200033183	319
	64820S003	01.20 Problem with integer pointer in conditional statement.	D200041277	321
	64820S003	01.20 Title description is incorrect.	D200045948	321
	64820S003	01.20 TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047563	321
	64820S003	01.50 Compilation on the VAX using batch mode generates incorrect listing file	D200055145	321
	64820S003	01.50 Host compilers do not put absolute pats specifications in relocatables	D200058982	322
PASS 3	64820S003	01.20 Compiler option \$LIST OBJ ON\$ generates wrong output information.	D200037101	320
	64820S003	01.20 Pass 3 fails to detect relative jump address out-of-range.	D200040717	320

- Z8000 PASCAL -

Keyword	Product number	uu.ff Description	Report #	page
*****none*****	64816	01.09 TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047456	324
	64816	01.10 Missing semicolon causes compiler to hang in Pass 1.	D200052605	324
INCLUDE	64816	01.09 Nested INCLUDE files 3 or more deep cause 64000 to "hang" in pass 3.	D200036798	324

Keyword index

- Z8000 PASCAL -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64816S004	00.00	Linker output file should use alternate file extension.	D200048868	326
	64816S004	01.00	Missing semicolon causes compiler to hang in Pass 1.	D200052639	326
PREPROCESSOR	64816S004	01.00	Preprocessor reports errors when symbols hp64000, vms or hpux w #if	D200058826	326

- Z8000 PASCAL -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64816S001	00.00	Linker output file should use alternate file extension.	D200048843	329
	64816S001	01.10	No form feed between the expanded listing and the cross reference table.	D200027680	327
	64816S001	01.20	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047464	328
	64816S001	01.30	Missing semicolon causes compiler to hang in Pass 1.	D200052613	328
PASS 3	64816S001	01.20	Compiler option \$LIST_OBJ ON\$ generates wrong output information.	D200037036	327
PREPROCESSOR	64816S001	01.30	Preprocessor reports errors when symbols hp64000, vms or hpux w #if	D200058800	328

- Z8000 PASCAL -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64816S003	00.00	Linker output file should use alternate file extension.	D200048850	332
	64816S003	01.20	No form feed between the expanded listing and the cross reference table.	D200027698	330
	64816S003	01.20	TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES	D200047472	331
	64816S003	01.30	Missing semicolon causes compiler to hang in Pass 1.	D200052621	331
PASS 3	64816S003	01.20	Compiler option \$LIST_OBJ ON\$ generates wrong output information.	D200037044	330
PREPROCESSOR	64816S003	01.30	Preprocessor reports errors when symbols hp64000, vms or hpux w #if	D200058818	331

- Z80H EMULATION -

Keyword	Product number	uu.ff	Description	Report #	page
*****none*****	64253	01.00	modify memory word to VALUE has bytes reversed from Z80 point of view	5000118414	333
	64253	01.00	Error in guided softkey syntax.	D200043398	333

Number: 2700005173 Product: 6800 C

64821

01.02

Keywords: CODE GENERATOR

One-line description:

16 bit comparison on a 8 bit unsigned short field.

Problem:

IMPROPER CODE GENERATED FOR STATEMENT INVOLVING unsigned short
VARIABLE UNLESS EXPLICITLY RE-CAST AS unsigned short.

```
main()
{
static unsigned short digit_index;
static unsigned short digit[12];
int a,b;
if (digit[digit_index]--){
a=4;
b=4;}
else{
a=5;
b=5;}
}
```

IMPROPER CODE IS GENERATED FOR THE COMPARISON (ie THE COMPARISON IS DONE
ON 16 BITS (8 OF WHICH HAVE BEEN CLEARED) AGAINST #0FFFFH.

12/10/85: The problem also arises if you compare a constant against
an unsigned short. For example if you declared:

```
#define constant ~0
unsigned short var;
and later compared these two the compiler will zero out the upper byte
of the variable var and then compare it to FFFFH. Thus, the condition
is never met.
```

12/16/85: Another example of incorrect code being generated when a
char variable is used in a test condition is as follows:

```
char a;
main()
{
a = -1;
if(a == -1)
a = 'A';
}
```

Temporary solution:

IF THE LINE IN QUESTION IS CHANGED TO:

```
if ((unsigned short)digit[digit_index]--){
```

CORRECT CODE IS GENERATED ALTHOUGH digit[] HAS ALREADY BEEN
DECLARED unsigned short.

12/10/85: Declare the constant as a short. In other words:
#define constant 0FFH.

12/16/85: If only 128 valid characters are required the variable can
be declared as a short int.

Signed off 08/25/86 in release 101.06

Number: 2700005181 Product: 6800 C

64821

01.02

Keywords: CODE GENERATOR

One-line description:

Left shift operator when shifting by one in a logical expr. is incorrect

Problem:

ORDER OF ELEMENTS FOR AN OR TYPE OPERATION MAY IMPACT
THE FOLLOWING PROGRAM GENERATES IMPROPER CODE:

```
"C"
"6800"
fct(data)
unsigned short data;
{
data = data << 1 | data >> 7;
}
```

Temporary solution:

CHANGING ORDER OF ELEMENTS IN "OR" :

```
data = data >> 7 | data << 1;
```

GENERATES CORRECT CODE. The correct code is also generated if the var-
iable "data" is global. This bug only occurs if left shifting by 1.

Signed off 08/25/86 in release 101.06

Number: D200013953 Product: 6800 C

64821

01.04

Keywords: PASS 1

One-line description:

No warning or error: taking the sizeof a struct var. not declared.

Problem:

The compiler should generate an error in the following code.

```
"C"
"6800"
main () {
int y;
y = sizeof(struct x);
}
```

If x is not declared or is declared as anything other than a structure,
the program compiles with no error messages or warnings. It stores as
the size zero bytes.

Signed off 08/25/86 in release 101.06

Number: D200015313 Product: 6800 C 64821 01.04

Keywords: CODE GENERATOR

One-line description:

An erroneous CLRA is generated if a char var. is decr. in a "while" loop

Problem:

When a variable declared as a char. is decremented when used as a counter in a while expression, an erroneous CLRA instruction is generated.

The following exemplifies this:

```
"C"
"6800"
char count=5;
main() {
    while (count--);
}
```

After count is decremented and stored into the data area, a CLRA instruction is executed. This happens before the jump to TFR_DtoX and as a result the new value of X is 00xxH since A was cleared before the transfer of D to X. This only happens when "count" is declared a character variable and is being decremented in the "while" loop.

Temporary solution:

Use a for loop for this segment.

```
for ( count = 5; count = 0; count--);
```

Signed off 08/25/86 in release 101.06

Number: D200015370 Product: 6800 C 64821 01.04

Keywords: CODE GENERATOR

One-line description:

A shift assignment operation (<<=) generates incorrect code.

Problem:

If a shift assignment is used instead of a shift within an assignment, the compiler uses the high byte of the variable to be used as the shift counter instead of the low byte. The following is an example:

```
"C"
"procesor name"
char data=1;
int shift=4;
main () {
    data=data<<shift; /* works correctly */
    data<<=shift; /* uses higher order byte of "shift" */
}
```

Temporary solution:

```
Use
data=data<<shift;
instead of
data<<=shift;
```

Signed off 08/25/86 in release 101.06

Number: D200027730 Product: 6800 C 64821 01.04

One-line description:

No form feed between the expanded listing and the cross reference table.

Problem:

During compilation, with XREF option on, the compiler does not provide a form feed (FF) in the listing file. The XREF starts on the same page as the end of the listing. Also, the page number says 535 when it should be page 2.

Temporary solution:

After compiling with the xref option, edit the expanded listing file and insert a "control L" before the beginning of the cross reference listing.

Signed off 08/25/86 in release 101.06

Number: D200031385 Product: 6800 C 64821 01.04

One-line description:

++ and -- operators evaluated with improper precedence.

Problem:

According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

```
Example 1: array[index++] = 1;
```

```
Example 2: array[index] = 1;
            index++;
```

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:

Separate the expression as shown in example 2.

Signed off 08/25/86 in release 101.06

Number: D200033191 Product: 6800 C 64821 01.04

One-line description:

Comparing character to zero in while loop generates incorrect code.

Problem:

If you compare a character variable to zero in a while loop, incorrect code is generated. The following code demonstrates the problem.

```
"C"
"6809"
```

```
proc()
{
```

```

char timeout = 10;
while(timeout--); /* Code generated here causes infinite loop.
}

```

The code generated for the while loop clears the A register and then compares the D register to -1. Therefore the condition is never met.

Temporary solution:

Declare the variable used in the test condition as an integer.

```

"C"
"6809"

proc()
{
int timeout = 10;
while (timeout--);
}

```

Signed off 08/25/86 in release 101.06

Number: D200040725 Product: 6800 C 64821 01.04

Keywords: PASS 3

One-line description:

Pass 3 fails to detect relative jump address out-of-range.

Problem:

Pass 3 of the compilation process may fail to detect a relative jump which is out of range. In the test program submitted the relative jump is generated for an IF..THEN statement while the compiler option OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:

As a temporary work around disable the compiler option OPTIMIZE around those sections of code which are suspect.

Signed off 08/25/86 in release 101.06

Number: D200041285 Product: 6800 C 64821 01.04

One-line description:

Problem with integer pointer in conditional statement.

Problem:

In the following example, two loads are performed, but no other code is generated to check for zero value.

```

"C"
"processor name"
#define NULL 0
fct(parm)
int *parm;
{
if (parm - NULL)

```

```

}
parm = 10;
}

```

Signed off 08/25/86 in release 101.06

Number: D200047571 Product: 6800 C 64821 01.04

One-line description:

TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 101.06

Number: D200050260 Product: 6800 C 300 64821S004 01.00

Keywords: PASS 1

One-line description:
Incorrect code is generated when complementing a parm. in a return stmt.

Problem:
In the following program the incorrect code is generated for the complement of the parameter to be returned.

```
"C"
"processor name"
unsigned short bug()
{
    return(~x);
}
```

The compiler generates a "NEGB" when it should be a "COMB"

Temporary solution:
Set up a temporary variable and assign the complement of the parameter to it and then return the temporary. For example,

```
unsigned short temp;
temp = ~x;
return temp;
```

Signed off 08/25/86 in release 401.10

Number: D200051268 Product: 6800 C 300 64821S004 01.00

One-line description:
++ and -- operators evaluated with improper precedence.

Problem:
According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

```
Example 1: array[index++] = 1;
Example 2: array[index] = 1;
           index++;
```

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:
Separate the expression as shown in example 2.

Signed off 08/25/86 in release 401.10

Number: D200052282 Product: 6800 C 300 64821S004 00.00

Keywords: CODE GENERATOR

One-line description:
Incorrect opcode "MOV A,ACC" allowed by our assembler

Problem:
The instruction "MOV A,ACC" was assemble and emulated by our products; however, the Intel 8051 goes into the weeds at this instruction. At first glance the machine code in the assembler listing appears valid (MOV A,ACC ->0000 E5E0), but the bottom of page 8-35 in Intel's microcontroller handbook states: *MOV A,ACC is not a valid instruction.

Neither our manuals nor AMD's user manual mention this instruction.

Signed off 08/25/86 in release 401.10

Number: D200059022 Product: 6800 C 300 64821S004 01.00

One-line description:
Host compilers do not put absolute pats specifications in relocatables

Problem:
Host compilers do not specify the full path name in the relocatable file.

Signed off 08/25/86 in release 401.10

Number: D200048983 Product: 6800 C 300 64821S004 00.00

One-line description:
Linker output file should use alternate file extension.

Signed off 08/25/86 in release 401.10

Number: D200015388 Product: 6800 C 500 64821S001 01.00

Keywords: CODE GENERATOR

One-line description:

An erroneous CLRA is gen. if a char var. is the counter in a "while"

Problem:

When a variable declared as a char. is decremented when used as a counter in a while expression, an erroneous CLRA instruction is generated.

The following exemplifies this:

```
"C"
"6800"
char count=5;
main() {
    while (count--);
}
```

After count is decremented and stored into the data area, a CLRA instruction is executed. This happens before the jump to TFR_DtoX and as a result the new value of X is 00xxH since A was cleared before the transfer of D to X. This only happens when "count" is declared a character variable and is being decremented in the "while" loop.

Temporary solution:

Use a for loop for this segment.
for (count = 5; count = 0; count--);

Signed off 08/25/86 in release 101.50

Number: D200015446 Product: 6800 C 500 64821S001 01.00

Keywords: CODE GENERATOR

One-line description:

A shift assignment operation (<<=) generates incorrect code.

Problem:

If a shift assignment is used instead of a shift within an assignment, the compiler uses the high byte of the variable to be used as the shift counter instead of the low byte. The following is an example:

```
"C"
"6800"
char data=1;
int shift=4;
main () {
    data=data<<shift; /* works correctly */
    data<<=shift; /* uses higher order byte of "shift" */
}
```

Temporary solution:

Don't use a shift assignment statement like those above.

Signed off 08/25/86 in release 101.50

Number: D200015644 Product: 6800 C 500 64821S001 01.00

Keywords: PASS 1

One-line description:

Incorrect code is generated when complementing a parm. in a return stmt.

Problem:

In the following program the incorrect code is generated for the complement of the parameter to be returned.

```
"C"
"6800"
unsigned short bug()
{
    return(~x);
}
```

The compiler generates a "NEGB" when it should be a "COMB"

Temporary solution:

Set up a temporary variable and assign the complement of the parameter to it and then return the temporary. For example,

```
unsigned short temp;
temp = ~x;
return temp;
```

Signed off 08/25/86 in release 101.50

Number: D200021725 Product: 6800 C 500 64821S001 01.10

One-line description:

Left shift operator when shifting by one in a logical expr. is incorrect

Problem:

ORDER OF ELEMENTS FOR AN OR TYPE OPERATION MAY IMPACT
THE FOLLOWING PROGRAM GENERATES IMPROPER CODE:

CORRECT CODE GENERATION.

```
"C"
"6800"
fct(data)
unsigned short data;
{
    data = data << 1 | data >> 7;
}
```

CHANGING ORDER OF ELEMENTS IN "OR" :
data = data >> 7 | data << 1;

GENERATES CORRECT CODE. The correct code is also generated if the variable "data" is global. This bug only occurs if left shifting by 1.

Signed off 08/25/86 in release 101.50

Number: D200031393 Product: 6800 C 500 64821S001 01.10

One-line description:

++ and -- operators evaluated with improper precedence.

Problem:

According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

Example 1: array[index++] = 1;

Example 2: array[index] = 1;
index++;

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:

Separate the expression as shown in example 2.

Signed off 08/25/86 in release 101.50

Number: D200033209 Product: 6800 C 500 64821S001 01.10

One-line description:

Comparing character to zero in while loop generates incorrect code.

Problem:

If you compare a character variable to zero in a while loop, incorrect code is generated. The following code demonstrates the problem.

"C"
"6809"

```
proc()
{
    char timeout = 10;

    while(timeout--); /* Code generated here causes infinite loop.
}
```

The code generated for the while loop clears the A register and then compares the D register to -1. Therefore the condition is never met.

Temporary solution:

Declare the variable used in the test condition as an integer.

"C"
"6809"

```
proc()
{
    int timeout = 10;

    while (timeout--);
}
```

Signed off 08/25/86 in release 101.50

- 6800 C -

Number: D200035840 Product: 6800 C 500 64821S001 01.10

Keywords: CODE GENERATOR

One-line description:

16 bit comparison on a 8 bit unsigned short field.

Problem:

IMPROPER CODE GENERATED FOR STATEMENT INVOLVING unsigned short VARIABLE UNLESS EXPLICITLY RE-CAST AS unsigned short.

```
main()
{
    static unsigned short digit_index;
    static unsigned short digit[12];
    int a,b;
    if (digit[digit_index--]){
        a=4;
        b=4;}
    else{
        a=5;
        b=5;}
}
IMPROPER CODE IS GENERATED FOR THE COMPARISON (ie THE COMPARISON IS DONE ON 16 BITS (8 OF WHICH HAVE BEEN CLEARED) AGAINST #0FFFFH.
12/10/85: The problem also arises if you compare a constant against an unsigned short. For example if you declared:
#define constant ~0
unsigned short var;
and later compared these two the compiler will zero out the upper byte of the variable var and then compare it to FFFFH. Thus, the condition is never met.
```

12/16/85: Another example of incorrect code being generated when a char variable is used in a test condition is as follows:

```
char a;
main()
{
    a = -1;
    if(a == -1)
        a = 'A';
}
```

Temporary solution:

IF THE LINE IN QUESTION IS CHANGED TO:

```
if ((unsigned short)digit[digit_index--]){
CORRECT CODE IS GENERATED ALTHOUGH digit[] HAS ALREADY BEEN DECLARED unsigned short.
12/10/85: Declare the constant as a short. In other words:
#define constant 0FFH.
12/16/85: If only 128 valid characters are required the variable can be declared as a short int.
```

- 6800 C -

Signed off 08/25/86 in release 101.50

Number: D200037119 Product: 6800 C 500 64821S001 01.20

Keywords: PASS 3

One-line description:

Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:

Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect data being output to the list file. In selected cases, machine code will be incorrectly listed. For example, consider the following Pascal program.

```
$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;
```

```
VAR
  a, b : BOOLEAN;
```

```
PROCEDURE one;
```

```
  BEGIN
    a := b;
  END;
```

In the example listed above, the output file will denote machine code of the form FFFFC00001 for one of the generated assembly statements. The correct value should be C8000001. This problem is caused by an incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 101.50

Number: D200040733 Product: 6800 C 500 64821S001 01.20

Keywords: PASS 3

One-line description:

Pass 3 fails to detect relative jump address out-of-range.

Problem:

Pass 3 of the compilation process may fail to detect a relative jump which is out of range. In the test program submitted the relative jump is generated for an IF..THEN statement while the compiler option OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:

As a temporary work around disable the compiler option OPTIMIZE around those sections of code which are suspect.

Signed off 08/25/86 in release 101.50

- 6800 C -

Number: D200041293 Product: 6800 C 500 64821S001 01.20

One-line description:

Problem with integer pointer in conditional statement.

Problem:

In the following example, two loads are performed, but no other code is generated to check for zero value.

```
"C"
"processor name"
#define NULL 0
fct(parm)
int *parm;
{
  if (parm - NULL)
    parm = 10;
}
```

Signed off 08/25/86 in release 101.50

Number: D200045955 Product: 6800 C 500 64821S001 01.20

One-line description:

Title description is incorrect.

Signed off 08/25/86 in release 101.50

Number: D200047589 Product: 6800 C 500 64821S001 01.20

One-line description:

TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 101.50

Number: D200049718 Product: 6800 C 500 64821S001 00.00

One-line description:

NO CROSS REFERENCE TABLE IS GENERATED

Problem:

"C" COMPILERS DO NOT GENERATE A CROSS REFERENCE TABLE ON THE VAX.

Temporary solution:

NONE KNOWN AT PRESENT

Signed off 04/18/86 in release 101.50

Number: D200059006 Product: 6800 C 500 64821S001 01.40

One-line description:

Host compilers do not put absolute pats specifications in relocatables

Problem:

Host compilers do not specify the full path name in the

- 6800 C -

relocatable file.

Signed off 08/25/86 in release 101.50

Number: D200048967 Product: 6800 C 500 64821S001 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 101.50

Number: D200015396 Product: 6800 C

VAX 64821S003

01.00

Keywords: CODE GENERATOR

One-line description:

An erroneous CLRA is gen. if a char var. is used as a ctr. in a "while"

Problem:

When a variable declared as a char. is decremented when used as a counter in a while expression, an erroneous CLRA instruction is generated.

The following exemplifies this:

```
"C"
"6800"
char count=5;
main() {
    while (count--);
}
```

After count is decremented and stored into the data area, a CLRA instruction is executed. This happens before the jump to TFR_DtoX and as a result the new value of X is 00xxH since A was cleared before the transfer of D to X. This only happens when "count" is declared a character variable and is being decremented in the "while" loop.

Temporary solution:

Use a for loop for this segment.

```
for ( count = 5; count = 0; count--);
```

Signed off 08/25/86 in release 301.80

Number: D200015453 Product: 6800 C

VAX 64821S003

01.00

Keywords: CODE GENERATOR

One-line description:

A shift assignment operation (<<=) generates incorrect code.

Problem:

If a shift assignment is used instead of a shift within an assignment, the compiler uses the high byte of the variable to be used as the shift counter instead of the low byte. The following is an example:

```
"C"
"6800"
char data=1;
int shift=4;
main () {
    data=data<<shift; /* works correctly */
    data<<=shift; /* uses higher order byte of "shift" */
}
```

Temporary solution:

Don't use a shift assignment statement like those above.

Signed off 08/25/86 in release 301.80

Number: D200015669 Product: 6800 C VAX 64821S003 01.00

Keywords: PASS 1

One-line description:
 Incorrect code is generated when complementing a parm. in a return stmt.

Problem:
 In the following program the incorrect code is generated for the complement of the parameter to be returned.

```
"C"
"6800"
unsigned short bug()
{
    return(~x);
}
```

The compiler generates a "NEGB" when it should be a "COMB"

Temporary solution:
 Set up a temporary variable and assign the complement of the parameter to it and then return the temporary. For example,

```
    unsigned short temp;
    temp = ~x;
    return temp;
```

Signed off 08/25/86 in release 301.80

 Number: D200021733 Product: 6800 C VAX 64821S003 01.10

One-line description:
 Left shift operator when shifting by one in a logical expr. is incorrect

Problem:
 ORDER OF ELEMENTS FOR AN OR TYPE OPERATION MAY IMPACT
 THE FOLLOWING PROGRAM GENERATES IMPROPER CODE:

CORRECT CODE GENERATION.

```
"C"
"6800"
fct(data)
unsigned short data;
{
    data = data << 1 | data >> 7;
}
CHANGING ORDER OF ELEMENTS IN "OR" :
data = data >> 7 | data << 1;
```

GENERATES CORRECT CODE. The correct code is also generated if the variable "data" is global. This bug only occurs if left shifting by 1.

Signed off 08/25/86 in release 301.80

Number: D200031401 Product: 6800 C VAX 64821S003 01.20

One-line description:
 ++ and -- operators evaluated with improper precedence.

Problem:
 According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

```
Example 1: array[index++] = 1;
Example 2: array[index] = 1;
           index++;
```

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:
 Separate the expression as shown in example 2.

Signed off 08/25/86 in release 301.80

 Number: D200033217 Product: 6800 C VAX 64821S003 01.20

One-line description:
 Comparing character to zero in while loop generates incorrect code.

Problem:
 If you compare a character variable to zero in a while loop, incorrect code is generated. The following code demonstrates the problem.

```
"C"
"6809"
proc()
{
    char timeout = 10;

    while(timeout--); /* Code generated here causes infinite loop.
}
}
```

The code generated for the while loop clears the A register and then compares the D register to -1. Therefore the condition is never met.

Temporary solution:
 Declare the variable used in the test condition as an integer.

```
"C"
"6809"
proc()
{
    int timeout = 10;

    while (timeout--);
}
```

Signed off 08/25/86 in release 301.80

Number: D200035857 Product: 6800 C VAX 64821S003 01.20

Keywords: CODE GENERATOR

One-line description:

16 bit comparison on a 8 bit unsigned short field.

Problem:

IMPROPER CODE GENERATED FOR STATEMENT INVOLVING unsigned short VARIABLE UNLESS EXPLICITLY RE-CAST AS unsigned short.

```
main()
{
static unsigned short digit_index;
static unsigned short digit[12];
int a,b;
if (digit[digit_index--]){
a=4;
b=4;}
else{
a=5;
b=5;}
}
IMPROPER CODE IS GENERATED FOR THE COMPARISON (ie THE COMPARISON IS DONE
ON 16 BITS (8 OF WHICH HAVE BEEN CLEARED) AGAINST #0FFFFH.
12/10/85: The problem also arises if you compare a constant against
an unsigned short. For example if you declared:
#define constant ~0
unsigned short var;
and later compared these two the compiler will zero out the upper byte
of the variable var and then compare it to FFFFH. Thus, the condition
is never met.
```

12/16/85: Another example of incorrect code being generated when a char variable is used in a test condition is as follows:

```
char a;
main()
{
a = -1;
if(a == -1)
a = 'A';
}
```

Temporary solution:

IF THE LINE IN QUESTION IS CHANGED TO:

if ((unsigned short)digit[digit_index--]){

CORRECT CODE IS GENERATED ALTHOUGH digit[] HAS ALREADY BEEN DECLARED unsigned short.

12/10/85: Declare the constant as a short. In other words:

#define constant 0FFH.

12/16/85: If only 128 valid characters are required the variable can be declared as a short int.

- 6800 C -

Signed off 08/25/86 in release 301.80

Number: D200037127 Product: 6800 C VAX 64821S003 01.20

Keywords: PASS 3

One-line description:

Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:

Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect data being output to the list file. In selected cases, machine code will be incorrectly listed. For example, consider the following Pascal program.

```
$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;

VAR
a, b : BOOLEAN;

PROCEDURE one;

BEGIN
a := b;
END;
```

In the example listed above, the output file will denote machine code of the form FFFFC00001 for one of the generated assembly statements. The correct value should be C8000001. This problem is caused by an incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE. THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 301.80

Number: D200040741 Product: 6800 C VAX 64821S003 01.20

Keywords: PASS 3

One-line description:

Pass 3 fails to detect relative jump address out-of-range.

Problem:

Pass 3 of the compilation process may fail to detect a relative jump which is out of range. In the test program submitted the relative jump is generated for an IF..THEN statement while the compiler option OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:

As a temporary work around disable the compiler option OPTIMIZE around those sections of code which are suspect.

Signed off 08/25/86 in release 301.80

- 6800 C -

 Number: D200041301 Product: 6800 C VAX 64821S003 01.20

One-line description:
 Problem with integer pointer in conditional statement.

Problem:
 In the following example, two loads are performed, but no other code is generated to check for zero value.

```
"C"
"processor name"
#define NULL 0
fct(parm)
int *parm;
{
  if (parm - NULL)
    parm = 10;
}
```

Signed off 08/25/86 in release 301.80

 Number: D200045963 Product: 6800 C VAX 64821S003 01.20

One-line description:
 Title description is incorrect.

Signed off 08/25/86 in release 301.80

 Number: D200047597 Product: 6800 C VAX 64821S003 01.20

One-line description:
 TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 301.80

 Number: D200055152 Product: 6800 C VAX 64821S003 01.50

One-line description:
 Compilation on the VAX using batch mode generates incorrect listing file

Problem:
 The test files can be found on the VAX750 under user\$disk:[robin.hughes.rgalo.test]. The following test files were used:

1. MTINHST_C. - File which contains one error- a missing '}' on line 70
2. TMTINHST_C. - Error-free version of MTINHST_C.
3. MTOPNDF_C. - File which contains one error - missing declaration for integer 'j'
4. MTOPNDF_C. - Error-free version of MTOPNDF_C.

One logical name must be defined as follows to access the include files referenced by the test programs:

- 6800 C -

```
$define BSLN user$disk:[robin.hughes.wsbsln.baseline]
```

When the four files were compiled interactively, the two error-free versions generated correct listings. The first file (MTINHST_C.) generated an incomplete and incorrect listing file. The listing showed the include files inserted first, followed by "C", "8086" and a few other lines of the program. The output displayed on the screen looked like:

```
In pass1.
  70 else ^25
      136 ^408
In C Nocode.
comp: C Nocode cannot recover from errors.
```

When the third file (MTOPNDF_C.) was compiled, the listing appeared fine except for the insertion a some strange control charaters.

These last two files were compiled in batch mode (file: user\$disk:[robin.hughes.rgalo.test]hughes.com). The first file (MTINHST_C.) generated a complete but incorrect listing. Although two errors were found (25 & 408) the line at the bottom stated that errors = 0. The include file expansion preceeded the "C" and "8086" in the listing, and lines like, #include filename, were still in the file. The error message was at line 72 of the listing instead of line 2472 were the '}' was actual missing. Finally the last 100 lines had useless numbers in the left margin.

When the third file (MTOPNDF_C.) was compiled, an incomplete listing was generated with the include file expansions listed first.

All of these tests were done on the VAX750 with the /e/v/o options.

This problem also occurs on the 68000.

Temporary solution:
 No temporary solution available

Signed off 08/25/86 in release 301.80

 Number: D200059014 Product: 6800 C VAX 64821S003 01.50

One-line description:
 Host compilers do not put absolute pats specifications in relocatables

Problem:
 Host compilers do not specify the full path name in the relocatable file.

Signed off 08/25/86 in release 301.80

- 6800 C -

Number: D200048975 Product: 6800 C VAX 64821S003 00.00

One-line description:
Linker output file should use alternate file extension.

Signed off 08/25/86 in release 301.80

Number: 2700004804 Product: 6800 PASCAL 64811 01.08

Keywords: DEBUG LIBRARY

One-line description:
X-reg modified after MUL or DIV operations.

Signed off 08/25/86 in release 101.10

Number: 5000084806 Product: 6800 PASCAL 64811 01.08

Keywords: PARAMETERS RANGE

One-line description:
Incorrect parameter passing with \$RANGE ON\$.

Problem:
If range is on and the parameter to be passed is not the first element of a record, the parameter is passed incorrectly.

Temporary solution:
Don't turn range on around function or procedure calls that pass elements of a record.

Signed off 08/25/86 in release 101.10

Number: 5000104612 Product: 6800 PASCAL 64811 01.08

Keywords: RANGE

One-line description:
Incorrect code generated for multiple array comparisons.

Problem:
\$EXTENSIONS;RANGE\$
VAR LA : ARRAY [0..1] OF BYTE;
B : BYTE;
BOOL : BOOLEAN;

BEGIN
BOOL := (B > LA[0]) OR (B > LA[1]); {GENERATES INCORRECT CODE. E.G., A
CALL TO EMPTY_SET_}

Temporary solution:
\$RANGE OFF\$

Signed off 08/25/86 in release 101.10

Number: 5000104620 Product: 6800 PASCAL 64811 01.08

Keywords: RANGE

One-line description:
RECORD accesses using WITH generate call to EMPTY_SET_ if \$RANGE ON\$.

Problem:


```

$EXTENSIONS;RANGE$
VAR I : INTEGER;
REC : RECORD
    PLACE HOLDER : BYTE;
    B : BYTE;
END;

```

```

BEGIN
WITH REC DO I := B; {GENERATES A CALL TO EMPTY_SET_, USED BY PASS 2 AS
    A MEANS OF ERROR RECOVERY}
WITH REC DO i := BYTE(B); {OK}

```

Signed off 08/25/86 in release 101.10

```

Number: 5000120378 Product: 6800 PASCAL 64811 01.08

```

Keywords: PARAMETERS

One-line description:
 Compiler accepts actual and formal parameters of different types.

Problem:
 The manual states that actual and formal parameters must match in number, order and type. If the formal and actual parameters are of different types but are the same size, an error message is not generated. If the formal parameter is a different type and size of the actual parameter, a warning message is generated (505 - type change changes physical size). Neither case produces the expected 142 error - illegal parameter substitution.

The following program demonstrates the problem:

```

"processor name"
PROGRAM TEST;

$EXTENSIONS ON $

TYPE T1 = 0..10;
    T2 = -20..20;

VAR V1 : T2;
    V2 : BYTE;

PROCEDURE PROC1 (VAR P1 : T1);

    BEGIN
    END;

PROCEDURE PROC2 (VAR P2 : INTEGER);

    BEGIN
    END;

BEGIN
    PROC1(V1);
    PROC2(V2);
END.

```

- 6800 PASCAL -

This problem occurred on all pascal compilers.

Temporary solution:
 No known temporary solution.

Signed off 08/25/86 in release 101.10

```

Number: D200014795 Product: 6800 PASCAL 64811 01.00

```

One-line description:
 Statement Sequences.

Problem:
 Certain statement sequences involving mixed real and integer expressions with the \$RANGE_ON\$ option, may cause "Too many errors in Pass2" error message.

Temporary solution:
 Turn off the \$RANGE_ON\$ option if this occurs.
 Note: a brief example is not verifiable at this time.
 The error can only be created in a moderately large file.

Signed off 08/25/86 in release 101.10

```

Number: D200034959 Product: 6800 PASCAL 64811 01.08

```

One-line description:
 "IF B2" after "REPEAT..UNTIL B1 OR B2" doesn't work.

Problem:
 VAR BOOL1, BOOL2 : BOOLEAN;

```

BEGIN
REPEAT
UNTIL BOOL1 OR BOOL2
IF BOOL2 THEN.....{THIS CHECKS TH B REGISTER WHICH CONTAINS
    BOOL1 + BOOL2, NOOT BOOL2}
$AMNESIA +$

```

Signed off 08/25/86 in release 101.10

```

Number: D200036764 Product: 6800 PASCAL 64811 01.08

```

Keywords: INCLUDE

One-line description:
 Nested INCLUDE files 3 or more deep cause 64000 to "hang" in pass 3.

Problem:
 Nested INCLUDE files 3 or more deep cause 64000 to hang in pass 3.

Temporary solution:
 None at this time.

Signed off 08/25/86 in release 101.10

- 6800 PASCAL -

SRB detail reports as of 08/25/86 Page: 27

Number: D200037663 Product: 6800 PASCAL 64811 01.08

Keywords: PASS 2 RANGE REAL

One-line description:

Stops in Pass 2 if a long program using real with \$RANGE ON\$.

Problem:

The compiler stops in pass 2 in long programs using real numbers if \$RANGE ON\$.

Signed off 08/25/86 in release 101.10

Number: D200037713 Product: 6800 PASCAL 64811 01.08

Keywords: PASS 2

One-line description:

ODD(INTEGER) in recursive procedure causes too many pass 2 errors.

Problem:

The use of ODD(16-bit INTEGER TYPE) may cause the compiler to stop in PASS 2 with too many errors to continue if it is done in a recursive procedure.

Signed off 08/25/86 in release 101.10

Number: D200047332 Product: 6800 PASCAL 64811 01.08

One-line description:

TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 101.10

Number: D200051987 Product: 6800 PASCAL 64811 01.09

Keywords: CONSTANTS

One-line description:

Constants may not be assigned their full 32 bit values.

Problem:

CONST
C1 = (OFFFFFF80H); will not be acceptable to the compiler even though in some situations we specify that a constant must be defined this way.

Temporary solution:

No known temporary solution.

Signed off 08/25/86 in release 101.10

SRB detail reports as of 08/25/86 Page: 28

Number: D200052449 Product: 6800 PASCAL 64811 01.09

One-line description:

Missing semicolon causes compiler to hang in Pass 1.

Problem:

The following code causes the 64000 to hang in pass 1. An error is generated on the hosts stating that parsing has stopped at a particular line number.

```
"processor name"
```

```
PROGRAM MAIN;  
TYPE  
STRUCTURED= RECORD  
    INT1:INTEGER;  
    INT2:INTEGER;  
END;
```

```
PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);  
VAR I:INTEGER;  
BEGIN  
I:=P1      <--This missing semicolon causes the problem  
I:=P1.2;  
I:=P2;  
END;
```

```
BEGIN  
END.
```

Temporary solution:

If the compiler hangs, look for a statement without a semicolon. On the 64000, the status line will show which line of code it stopped on. On the hosts, the error message generated indicates which line of code parsing stopped on.

Signed off 08/25/86 in release 101.10

Number: D200051870 Product: 6800 PASCAL 300 64811S004 01.00

Keywords: RANGE

One-line description:

Incorrect code generated for multiple array comparisons.

Problem:

```
$EXTENSIONS;RANGE$
VAR LA : ARRAY [0..1] OF BYTE;
    B : BYTE;
    BOOL : BOOLEAN;
```

```
BEGIN
BOOL := (B > LA[0]) OR (B > LA[1]); {GENERATES INCORRECT CODE. E.G., A
CALL TO EMPTY_SET_}
```

Temporary solution:

\$RANGE OFF\$

Signed off 08/25/86 in release 401.10

Number: D200051888 Product: 6800 PASCAL 300 64811S004 01.00

Keywords: RANGE

One-line description:

RECORD accesses using WITH generate call to EMPTY_SET_ if \$RANGE ON\$.

Problem:

```
$EXTENSIONS;RANGE$
VAR I : INTEGER;
    REC : RECORD
        PLACEHOLDER : BYTE;
        B : BYTE;
    END;
```

```
BEGIN
WITH REC DO I := B; {GENERATES A CALL TO EMPTY_SET_, USED BY PASS 2 AS
A MEANS OF ERROR RECOVERY}
WITH REC DO i := BYTE(B); {OK}
```

Temporary solution:

No known temporary solution.

Signed off 08/25/86 in release 401.10

Number: D200052472 Product: 6800 PASCAL 300 64811S004 01.00

One-line description:

Missing semicolon causes compiler to hang in Pass 1.

Problem:

The following code causes the 64000 to hang in pass 1. An error is generated on the hosts stating that parsing has stopped at a particular line number.

```
"processor name
PROGRAM MAIN;
TYPE
STRUCTURED= RECORD
    INT1:INTEGER;
    INT2:INTEGER;
END;
```

```
PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
I:=P1      <--This missing semicolon causes the problem
I:=P1.2;
I:=P2;
END;
```

```
BEGIN
END.
```

Temporary solution:

If the compiler hangs, look for a statement without a semicolon. On the 64000, the status line will show which line of code it stopped on. On the hosts, the error message generated indicates which line of code parsing stopped on.

Signed off 08/25/86 in release 401.10

Number: D200058701 Product: 6800 PASCAL 300 64811S004 01.00

Keywords: PREPROCESSOR

One-line description:

Preprocessor reports errors when symbols hp64000, vms or hpux w #if

Signed off 08/25/86 in release 401.10

Number: D200059139 Product: 6800 PASCAL 300 64811S004 01.00

One-line description:

Host compilers do not put absolute paths specifications in relocatables

Problem:

Host compilers do not specify the full path name in the relocatable file.

Signed off 08/25/86 in release 401.10

Number: D200048744 Product: 6800 PASCAL 300 64811S004 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 401.10

Number: 2700005512 Product: 6800 PASCAL 500 64811S001 01.08

One-line description:

No form feed between the expanded listing and the cross reference table.

Problem:

During compilation, with XREF option on, the compiler does not provide a form feed (FF) in the listing file. The XREF starts on the same page as the end of the listing. Also, the page number says 535 when it should be page 2.

Temporary solution:

After compiling with the xref option, edit the expanded listing file and insert a "control L" before the beginning of the cross reference listing.

Signed off 08/25/86 in release 101.40

Number: D200014779 Product: 6800 PASCAL 500 64811S001 01.00

One-line description:

Statement sequences.

Problem:

Certain statement sequences invoking the ODD(x) function cause "Too many errors in Pass2" error message.

Temporary solution:

error: IF ODD(x) AND (i<>j) THEN ...may produce this error
work around: IF (ODD(x)=TRUE) AND (i<>j) THEN ... should work OK.

Signed off 08/25/86 in release 101.40

Number: D200030569 Product: 6800 PASCAL 500 64811S001 01.10

Keywords: PARAMETERS

One-line description:

Incorrect parameter passing with \$RANGE ON\$.

Problem:

If range is on and the parameter to be passed is not the first element of a record, the parameter is passed incorrectly.

Temporary solution:

Don't turn range on around function or procedure calls that pass elements of a record.

Signed off 08/25/86 in release 101.40

Number: D200036699 Product: 6800 PASCAL 500 64811S001 01.20

One-line description:

"IF B2" after "REPEAT..UNTIL B1 OR B2" doesn't work.

Problem:

- 6800 PASCAL -

VAR BOOL1, BOOL2 : BOOLEAN;

BEGIN

REPEAT

UNTIL BOOL1 OR BOOL2

IF BOOL2 THEN.....{THIS CHECKS TH B REGISTER WHICH CONTAINS
BOOL1 + BOOL2, NOOT BOOL2}

\$AMNESIA +\$

Signed off 08/25/86 in release 101.40

Number: D200036962 Product: 6800 PASCAL 500 64811S001 01.20

Keywords: PASS 3

One-line description:

Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:

Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect data being output to the list file. In selected cases, machine code will be incorrectly listed. For example, consider the following Pascal program.

\$EXTENSIONS ON\$

\$LIST_OBJ ON\$

PROGRAM test;

VAR

a, b : BOOLEAN;

PROCEDURE one;

BEGIN

a := b;

END;

In the example listed above, the output file will denote machine code of the form FFFFC00001 for one of the generated assembly statements. The correct value should be C8000001. This problem is caused by an incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 101.40

Number: D200040204 Product: 6800 PASCAL 500 64811S001 01.20

Keywords: RANGE

One-line description:

Incorrect code generated f r multiple array comparisons.

Problem:

\$EXTENSIONS;RANGE\$

- 6800 PASCAL -

```
VAR LA      : ARRAY [0..1] OF BYTE;
    B       : BYTE;
    BOOL    : BOOLEAN;
```

```
BEGIN
BOOL := (B > LA[0]) OR (B > LA[1]); {GENERATES INCORRECT CODE. E.G., A
CALL TO EMPTY_SET_.}
```

Temporary solution:
\$RANGE OFF\$

Signed off 08/25/86 in release 101.40

Number: D200040220 Product: 6800 PASCAL 500 64811S001 01.20

Keywords: RANGE

One-line description:
RECORD accesses using WITH generate call to EMPTY_SET_ if \$RANGE ON\$.

Problem:
TYPE SET_TYPE = SET OF (B0,B1,B2,B3,B4,B5,B6,B7);
VAR X : SET_TYPE;
BEGIN
IF X <= [B3,B4] THEN; {GENERATES INCORRECT CODE}
IF X >= [B3,B4] THEN; {GENERATES INCORRECT CODE}

Temporary solution:
None at this time.

Signed off 08/25/86 in release 101.40

Number: D200047340 Product: 6800 PASCAL 500 64811S001 01.20

One-line description:
TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 101.40

Number: D200052217 Product: 6800 PASCAL 500 64811S001 01.30

One-line description:
Host compilers do not put absolute pats specifications in relocatables

Problem:
Host compilers do not specify the full path name in the
relocatable file.

Signed off 08/25/86 in release 101.40

Number: D200052225 Product: 6800 PASCAL 500 64811S001 01.30

Keywords: PREPROCESSOR

One-line description:
Preprocessor reports errors when symbols hp64000, vms or hpux w #if

- 6800 PASCAL -

Signed off 08/25/86 in release 101.40

Number: D200052456 Product: 6800 PASCAL 500 64811S001 01.30

One-line description:
Missing semicolon causes compiler to hang in Pass 1.

Problem:
The following code causes the 64000 to hang in pass 1. An error
is generated on the hosts stating that parsing has stopped at
a particular line number.

```
"processor name"
PROGRAM MAIN;
TYPE
STRUCTURED= RECORD
    INT1:INTEGER;
    INT2:INTEGER;
END;
```

```
PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
I:=P1      <--This missing semicolon causes the problem
I:=P1.2;
I:=P2;
END;
```

```
BEGIN
END.
```

Temporary solution:
If the compiler hangs, look for a statement without a semicolon.
On the 64000, the status line will show which line of code it
stopped on. On the hosts, the error message generated indicates
which line of code parsing stopped on.

Signed off 08/25/86 in release 101.40

Number: D200046151 Product: 6800 PASCAL 500 64811S001 00.00

One-line description:
Linker output file should use alternate file extension.

Signed off 08/25/86 in release 101.40

- 6800 PASCAL -

Number: D200014787 Product: 6800 PASCAL VAX 64811S003 01.00

One-line description:
Statement sequences.

Problem:
Certain statement sequences invoking the ODD(x) function cause
"Too many errors in Pass2" error message.

Temporary solution:
error: IF ODD(x) AND (i<>j) THEN ...may produce this error
work around: IF (ODD(x)=TRUE) AND (i<>j) THEN ... should work OK.

Signed off 08/25/86 in release 301.60

Number: D200027631 Product: 6800 PASCAL VAX 64811S003 01.20

One-line description:
No form feed between the expanded listing and the cross reference table.

Problem:
During compilation, with XREF option on, the compiler does not provide
a form feed (FF) in the listing file. The XREF starts on the same page
as the end of the listing. Also, the page number says 535 when it
should be page 2.

Temporary solution:
After compiling with the xref option, edit the expanded listing file
and insert a "control L" before the beginning of the cross reference
listing.

Signed off 08/25/86 in release 301.60

Number: D200030577 Product: 6800 PASCAL VAX 64811S003 01.20

Keywords: PARAMETERS

One-line description:
Incorrect parameter passing with \$RANGE ON\$.

Problem:
If range is on and the parameter to be passed is not the first element
of a record, the parameter is passed incorrectly.

Temporary solution:
Don't turn range on around function or procedure calls that pass
elements of a record.

Signed off 08/25/86 in release 301.60

Number: D200036707 Product: 6800 PASCAL VAX 64811S003 01.20

One-line description:
"IF B2" after "REPEAT..UNTIL B1 OR B2" doesn't work.

Problem:
VAR BOOL1, BOOL2 : BOOLEAN;

```
BEGIN
REPEAT
UNTIL BOOL1 OR BOOL2
IF BOOL2 THEN.....{THIS CHECKS TH B REGISTER WHICH CONTAINS
                    BOOL1 + BOOL2, NOOT BOOL2}
```

\$AMNESIA +\$

Signed off 08/25/86 in release 301.60

Number: D200036970 Product: 6800 PASCAL VAX 64811S003 01.20

Keywords: PASS 3

One-line description:
Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:
Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect
data being output to the list file. In selected cases, machine code
will be incorrectly listed. For example, consider the following
Pascal program.

```
$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;
```

```
VAR
  a, b : BOOLEAN;
```

```
PROCEDURE one;
```

```
  BEGIN
    a := b;
  END;
```

In the example listed above, the output file will denote machine code
of the form FFFFC00001 for one of the generated assembly statements.
The correct value should be C8000001. This problem is caused by an
incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 301.60

Number: D200040212 Product: 6800 PASCAL VAX 64811S003 01.20

Keywords: RANGE

One-line description:
Incorrect code generated for multiple array comparisons.

Problem:
\$EXTENSIONS;RANGE\$
VAR LA : ARRAY [0..1] OF BYTE;
B : BYTE;
BOOL : BOOLEAN;

```
BEGIN
BOOL := (B > LA[0]) OR (B > LA[1]); {GENERATES INCORRECT CODE. E.G., A
CALL TO EMPTY_SET_.
```

Temporary solution:
\$RANGE OFF\$

Signed off 08/25/86 in release 301.60

Number: D200040238 Product: 6800 PASCAL VAX 64811S003 01.20

Keywords: RANGE

One-line description:
RECORD accesses using WITH generate call to EMPTY_SET_ if \$RANGE ON\$.

Problem:
TYPE SET_TYPE = SET OF (B0,B1,B2,B3,B4,B5,B6,B7);
VAR X : SET_TYPE;
BEGIN
IF X <= [B3,B4] THEN; {GENERATES INCORRECT CODE}
IF X >= [B3,B4] THEN; {GENERATES INCORRECT CODE}

Temporary solution:
None at this time.

Signed off 08/25/86 in release 301.60

Number: D200047357 Product: 6800 PASCAL VAX 64811S003 01.20

One-line description:
TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 301.60

Number: D200052464 Product: 6800 PASCAL VAX 64811S003 01.40

One-line description:
Missing semicolon causes compiler to hang in Pass 1.

Problem:
The following code causes the 64000 to hang in pass 1. An error is generated on the hosts stating that parsing has stopped at a particular line number.

- 6800 PASCAL -

```
"processor name"
PROGRAM MAIN;
TYPE
STRUCTURED= RECORD
    INT1:INTEGER;
    INT2:INTEGER;
END;
```

```
PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
I:=P1      <--This missing semicolon causes the problem
I:=P1.2;
I:=P2;
END;
```

```
BEGIN
END.
```

Temporary solution:
If the compiler hangs, look for a statement without a semicolon. On the 64000, the status line will show which line of code it stopped on. On the hosts, the error message generated indicates which line of code parsing stopped on.

Signed off 08/25/86 in release 301.60

Number: D200058693 Product: 6800 PASCAL VAX 64811S003 01.40

Keywords: PREPROCESSOR

One-line description:
Preprocessor reports errors when symbols hp64000, vms or hpux w #if

Signed off 08/25/86 in release 301.60

Number: D200059121 Product: 6800 PASCAL VAX 64811S003 01.40

One-line description:
Host compilers do not put absolute pats specifications in relocatables

Problem:
Host compilers do not specify the full path name in the relocatable file.

Signed off 08/25/86 in release 301.60

Number: D200048736 Product: 6800 PASCAL VAX 64811S003 00.00

One-line description:
Linker output file should use alternate file extension.

Signed off 08/25/86 in release 301.60

- 6800 PASCAL -

Number: D200031070 Product: 6800/2 ASSEMB 64841 01.13

One-line description:
Assembler flagging out of range error when it should not.

Problem:
There is a discrepancy on how out of range errors are handled. The below line will load the lower sixteen bits into register D (this seems appropriate):

LDD #10000000H

While the following line will flag an out of range error:

LDAA #10000000H

Temporary solution:
And the operand with OFFH. This will force it to eight bits.
"6800"

LDAA (#10000000H).AN.OFFH

Signed off 08/25/86 in release 101.15

Number: D200033423 Product: 6800/2 ASSEMB 64841 01.13

One-line description:
Error when using .NT. operator with immediate value whose MSB is set.

Problem:
If you use the .NT. logical operator on an immediate value whose upper bit is set, a legal range error is flagged. The opcode generated is correct.
"6801"

BITA #.NT.A0H ; LEGAL RANGE ERROR IS FLAGGED
BITA #.NT.7FH ; NO ERROR FLAGGED.

Temporary solution:
The code generated is correct, so ignore the error message.

Signed off 08/25/86 in release 101.15

Number: D200046797 Product: 6800/2 ASSEMB 64841 01.13

One-line description:
Assembler should denote an error on non-absolute .SET expressions.

Signed off 08/25/86 in release 101.15

Number: D200055608 Product: 6800/2 ASSEMB 64841 01.14

One-line description:
Four bit operations are now unsupported.

Problem:

- 6800/2 ASSEMB -

The following four mnemonics are not supported by the 6301/03 assembler:

BTST
BSET
BTGT
BCLR

Signed off 08/25/86 in release 101.15

- 6800/2 ASSEMB -

Number: D200048215 Product: 6800/2 ASSEMB 300 64841S004 01.00

Keywords: MACRO

One-line description:
Conditional instr. .IF with rational oper. in Macro creates bad code

Problem:
The use of the conditional instruction, .IF, with rational operator (.EQ.,.NE.,.LT.,.GT.,.LE.,.GE.) in a macro functions incorrectly. The following program demonstrates this problem:

```

      BUG          MACRO          &VAR
                .IF &VAR .LE. 0 SUB&&&&
                NOP
      SUB&&&&      NOP
                NOP
                MEND

                BUG -3
                BUG 1
                BUG 0
                END

```

Passing a 3 appears to create correct code, but 0 causes a ML error. Passing -1 to the MACRO creates code which doesn't call the subroutine. This is incorrect since -1 is less than 0. This same problem occurred with all the rational operators on all processors. The problem was consistent on the 64000, VAX, and 9000.

Signed off 08/25/86 in release 401.10

Number: D200053314 Product: 6800/2 ASSEMB 300 64841S004 01.00

One-line description:
Macro def. including .IF, within a IF causes assembler to stop code gen.

Problem:
If you have a ".IF" in a macro definition and that macro definition is within a conditional assembly "IF" then no code is generated. The program provided demonstrates the problem (see submitter text).

Temporary solution:
Pull the macro definition outside of the conditional if. No code will be generated for the definition.

"processor name"

```

ESSAI          EQU          0

MACRO          MACRO
                .IF          ESSAI.EQ.0    FIN
LABEL          LD           A,0
FIN            MEND

```

```

                IF          ESSAI
                MAC
                ENDIF

```

START LD A,3

Signed off 08/25/86 in release 401.10

Number: D200049197 Product: 6800/2 ASSEMB 300 64841S004 00.00

One-line description:
Linker output file should use alternate file extension.

Signed off 08/25/86 in release 401.10

Number: D200031096 Product: 6800/2 ASSEMB VAX 64841S003 01.20

One-line description:

Assembler flagging out of range error when it should not.

Problem:

There is a discrepancy on how out of range errors are handled. The below line will load the lower sixteen bits into register D (this seems appropriate):

```
LDD      #10000000H
```

While the following line will flag an out of range error:

```
LDA      #10000000H
```

Temporary solution:

And the operand with 0FFH. This will force it to eight bits. "6800"

```
LDA      (#10000000H).AN.OFFH
```

Signed off 08/25/86 in release 301.50

Number: D200046813 Product: 6800/2 ASSEMB VAX 64841S003 01.20

One-line description:

Assembler should denote an error on non-absolute .SET expressions.

Signed off 08/25/86 in release 301.50

Number: D200048207 Product: 6800/2 ASSEMB VAX 64841S003 01.40

Keywords: MACRO

One-line description:

Conditional instr. .IF with rational oper. in Macro creates bad code

Problem:

The use of the conditional instruction, .IF, with rational operator (.EQ.,.NE.,.LT.,.GT.,.LE.,.GE.) in a macro functions incorrectly. The following program demonstrates this problem:

```

BUG      MACRO      &VAR
          .IF &VAR .LE. 0 SUB&&&&
          NOP
          NOP
SUB&&&&   NOP
          NOP
          MEND

          BUG 3
          BUG -1
          BUG 0
          END

```

- 6800/2 ASSEMB -

Passing a 3 appears to create correct code, but 0 causes a ML error. Passing -1 to the MACRO creates code which doesn't call the subroutine. This is incorrect since -1 is less than 0. This same problem occurred with all the rational operators on all processors. The problem was consistent on the 64000, VAX, and 9000.

Signed off 08/25/86 in release 301.50

Number: D200053306 Product: 6800/2 ASSEMB VAX 64841S003 01.40

One-line description:

Macro def. including .IF, within a IF causes assembler to stop code gen.

Problem:

If you have a ".IF" in a macro definition and that macro definition is within a conditional assembly "IF" then no code is generated. The program provided demonstrates the problem (see submitter text).

Temporary solution:

Pull the macro definition outside of the conditional if. No code will be generated for the definition.

"processor name"

```

ESSAI    EQU      0

MAC      MACRO
          .IF      ESSAI.EQ.0  FIN
LABEL    LD       A,0
FIN      MEND

          IF      ESSAI
          MAC
          ENDIF

START    LD       A,3

```

Signed off 08/25/86 in release 301.50

Number: D200049189 Product: 6800/2 ASSEMB VAX 64841S003 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 301.50

- 6800/2 ASSEMB -

Number: 5000126516 Product: 68000 C 64819 01.07

One-line description:

Incorrect code when hex values are bit or-ed and passed as parameters.

Problem:

When two hex values are bit or-ed together, and at least one of the values is greater than or equal to 0x8000, the compiler interprets the passed value as a long word instead of a word. The following code demonstrates the problem:

```
"C"
"68000"
$FAR$
$CALL_ABS_LONG$
$LIB_ABS_LONG$
extern sample();
main()
{
  sample(0x8000); (*Generates correct code*)
  sample(0x0080 | 0x1000 | 0x7fff); (*Generates correct code*)
  sample(0x0080 | 0x1000 | 0x8000); (*Generates incorrect code*)
}
```

Temporary solution:

There are two possible temporary solutions.

1. Use an explicit type cast.

```
main()
{
  sample((int)(0x0080 | 0x1000 | 0x8000)); (*Both expressions
  sample(0x0080 | 0x1000 | (int)0x8000); generate correct
  } code *)
```

2. Use a temporary variable.

```
main()
{
  int j;
  j = 0x8000;
  sample (0x0080 | 0x1000 | j);
}
```

Signed off 08/25/86 in release 901.09

Number: 5000136234 Product: 68000 C 64819 01.00

Keywords: PASS 3

One-line description:

Pass 3 error flagged when 143-146 external functions are declared.

Problem:

Pass three error is generated when using a 'for' statement after many external declarations.

- 68000 C -

```
"C"
"68000"
```

```
$ASM_FILE$
extern FUNC_1();
extern FUNC_2();
.
.
.
)Cnnnd
extern FUNC_143();
main() {
  int i;
  for(i=0; i<=7; i++)
  ;
}
```

Temporary solution:

It appears that the error is flagged only if you have 143-146 external functions declared (inclusive). The problem may be resolved if you declare some dummy functions which will bring the total number above 146.

Signed off 08/25/86 in release 901.09

Number: D200008870 Product: 68000 C 64819 00.56

Keywords: CODE GENERATOR

One-line description:

Station reboot or bad code, statements of the form: x += (*ptr)*(ptr);

Problem:

When the += or -= operators (or the long form) are used to assign to an integer compatible variable the result of an integer compatible variable taken indirect operating on itself, the station may reboot or bad code may be produced. For example, the following result in a reboot.

```
char i, *j; int *p_1;
main() long *p_2;
{ i += (*j)*(j); } main()
{ *p_2 = *p_2 - (*p_1)*(p_1); }
```

Operators resulting in a reboot are: *, +, -, &, and |.
The % and / operators produce bad code, as in:

```
int *x, *y;
main()
{ *x -= (*y)%(*y); }
```

The xor function (^) appears to work correctly.

Temporary solution:

Use a temporary to hold the result of the operation on the indirects.

- 68000 C -

Then assign the temporary (via += or -=) to final destination.

```
char *p_1, p_2, temp;
main()
{ temp = (*p_1)*(*p_1);
  p_2 += temp;
}
```

Signed off 08/25/86 in release 901.09

Number: D200013938 Product: 68000 C 64819 01.07

Keywords: PASS 1

One-line description:

No warning or error: taking the sizeof a struct var. not declared.

Problem:

The compiler should generate an error in the following code.

```
"C"
"68000"
main () {
  int y;
  y = sizeof(struct x);
}
```

If x is not declared or is declared as anything other than a structure, the program compiles with no error messages or warnings. It stores as the size zero bytes.

Signed off 08/25/86 in release 901.09

Number: D200014282 Product: 68000 C 64819 01.07

Keywords: CODE GENERATOR

One-line description:

Comparing a variable to zero in a "for" statement often fails.

Problem:

When comparing a variable to zero in a test condition the instruction TST.W is used. This compares the operand with zero, storing no results, but setting condition codes according to the results of the test. The Carry and Overflow bits are always cleared by the TST instruction. The BCC instruction following the TST uses the carry and overflow bits when evaluating the branch condition thus resulting in the wrong branch. The following code is one example of this.

```
"C"
"68000"
main ()
{
  unsigned int i, count = 2;
  for ( i=count-1; i>=0; i--);
}
```

- 68000 C -

This code uses the BCS (branch if carry is set) instruction. This condition will never be satisfied and the loop will continue indefinitely.

Temporary solution:

Avoid comparing to the constant zero.

Signed off 08/25/86 in release 901.09

Number: D200014993 Product: 68000 C 64819 01.07

Keywords: CODE GENERATOR

One-line description:

Argument of a switch is sign-extended to long when it should remain int.

Problem:

Any case expression which has bit #15 set will never be selected due to the sign extension of the switch argument. The following is an example of this:

```
"C"
"68000"
int x;
main () {
  switch (x) {
    case 0xFFFF:
      break;
    default:
      break;
  }
}
```

The compiler first generates code to extend the argument x from a word to a long word using the "EXT.L" instruction. Then a word comparison is made to the case expressions using the "CMPI.L" instruction without sign extending the case expression's value. In the above program data register D7 contains the sign extended value of "x" when the following instruction is executed: CMPI.L #0000FFFFH, D7. Therefore, the case of x equaling 0xFFFF will never occur.

Temporary solution:

If a negative number is used as one of the case expressions, all of the comparisons are changed to CMPI.W from CMPI.L.

Signed off 08/25/86 in release 901.09

Number: D200015883 Product: 68000 C 64819 01.07

One-line description:

No error generated when an interrupt routine is explicitly called.

Problem:

The compiler fails to give an error message in a situation where an interrupt function is called from code (rather than via an interrupt vector). The following example illustrates.

- 68000 C -

```
"C"
"68000"
$INTERRUPT ON$
inter() {}
$INTERRUPT OFF$

main() {
    int i;
    i = inter();    /* This line should generate error #1104 */
}
```

Signed off 08/25/86 in release 901.09

Number: D200015990 Product: 68000 C 64819 01.07

Keywords: CODE GENERATOR

One-line description:

Wrong addressing mode used with \$BASE_PAGE\$ on in ASM68000 file.

Problem:

In the ASM68000 source generated by the \$ASM_FILE\$, the wrong addressing mode is used when the \$BASE_PAGE\$ directive is on.

Signed off 08/25/86 in release 901.09

Number: D200016014 Product: 68000 C 64819 01.07

Keywords: CODE GENERATOR

One-line description:

The wrong byte is accessed when a union is defined within a struct.

Problem:

```
"C"
"68000"
struct {
    char ch;
    union {
        char ch1;
        char ch2;
    } um;
} *str;
main() {
    str->um.ch1=1;
    str->um.ch2=2;
}
```

The variables "ch1" and "ch2" in the above example should be at um + 1. Although, in the expanded listing you see they are accessed at um + 2 as if the field "ch" was a 16 bit datatype.

Signed off 08/25/86 in release 901.09

Number: D200016592 Product: 68000 C 64819 01.07

Keywords: CODE GENERATOR

One-line description:

Structure with an odd-numbered char or short array gens. wrong code.

Problem:

The following code uses an incorrect offset from A0:

```
"C"
"68000"
struct { char name[3];
        char ext; } *ptr;

sub()
{
    ptr->ext = 'a';
}
```

The offset generated is 4[A0] when assigning 'a' to "ext" when it should be 3[A0]. This is not a problem with an even sized array or with an integer array.

Signed off 08/25/86 in release 901.09

Number: D200027714 Product: 68000 C 64819 01.07

One-line description:

No form feed between the expanded listing and the cross reference table.

Problem:

During compilation, with XREF option on, the compiler does not provide a form feed (FF) in the listing file. The XREF starts on the same page as the end of the listing. Also, the page number says 535 when it should be page 2.

Temporary solution:

After compiling with the xref option, edit the expanded listing file and insert a "control L" before the beginning of the cross reference listing.

Signed off 08/25/86 in release 901.09

Number: D200028621 Product: 68000 C 64819 01.07

One-line description:

Comp_symb file not being loaded on user specified disc.

Problem:

When over two logical units are present the comp_symb file is not being generated where specified. For example, if a file is compiled with the comp_symb option and the location of the output files is specified as LU1 the comp_symb file will be loaded onto LU0. If you later link with the comp_db option the link fails because comp_symb cannot be found.

Signed off 08/25/86 in release 901.09

Number: D200030734 Product: 68000 C 64819 01.07

Keywords: CODE GENERATOR

One-line description:
Incorrect code generated if fields are defined in a structure.

Problem:
The assembly code generated for the below C source is not correct. If any field of the structure is referenced the wrong offset is generated by the assembler.

```
"C"
"68000"
main ()
  struct{
    short int a;
    unsigned : 4;
    unsigned fl 1;} s;

    (*s).a=1;                /* this line causes incorrect offset
                             to be generated. */
  }
```

Temporary solution:
Declare the bit fields first.

```
"C"
"68000"
main()
{
  struct {
    unsigned fl :1;
    unsigned :4;
    short a ;
  } s;
```

Signed off 08/25/86 in release 901.09

Number: D200030742 Product: 68000 C 64819 01.07

Keywords: CODE GENERATOR

One-line description:
Variable may not be defined before an array in a structure.

Problem:
In a structure which includes an array(s) the array(s) must be defined before any other variable. If the other variable is declared before the array incorrect code will be generated when the array is dereferenced.

```
"C"
"68000"
struct a{
  char *p;
  char i[2];
```

- 68000 C -

```
    }
main()
{
  a *ad;
  ad->i =1;                /*Incorrect code will generated. */
}
```

Temporary solution:
Declare all arrays first.

```
"C"
"68000"
struct a{
  char i[2];
  char *p;
}
```

```
main()
{
  struct a *ad;
  ad->i=1;
}
END
**
```

Signed off 08/25/86 in release 901.09

Number: D200031328 Product: 68000 C 64819 01.07

One-line description:
++ and -- operators evaluated with improper precedence.

Problem:
According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

Example 1: array[index++] = 1;

Example 2: array[index] = 1;
index++;

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:
Separate the expression as shown in example 2.

Signed off 08/25/86 in release 901.09

Number: D200032052 Product: 68000 C 64819 01.07

Keywords: PASS 2

One-line description:
Stations jumps to PV when compiling file with syntax error.

Problem:

- 68000 C -

The file below will not compile on the 64000 or the 9000. On the 64K the station jumps into PV; the 9000 and VAX report a pass two error. If the syntax error is removed, the file will compile.

```
"C"
"68000"

enum boolean{true,false};
main()
{ enum boolean variable;
  proc(4,(enum boolean) &variable); /* BOOLEAN IS MISSING 'E' */
}
proc(parm1,parm2)
int parm1;
enum boolean *parm2;
{ *parm2 = true;
}

```

Signed off 08/25/86 in release 901.09

Number: D200033134 Product: 68000 C 64819 01.07

One-line description:

Comparing character to zero in while loop generates incorrect code.

Problem:

If you compare a character variable to zero in a while loop, incorrect code is generated. The following code demonstrates the problem.

```
"C"
"6809"

proc()
{
  char timeout = 10;
  while(timeout--); /* Code generated here causes infinite loop.
}

```

The code generated for the while loop clears the A register and then compares the D register to -1. Therefore the condition is never met.

Temporary solution:

Declare the variable used in the test condition as an integer.

```
"C"
"6809"

proc()
{
  int timeout = 10;
  while (timeout--);
}

```

Signed off 08/25/86 in release 901.09

Number: D200033449 Product: 68000 C

64819

01.07

One-line description:

Case statement involving double indirection is not generating right code

Problem:

In the special case outlined below the 68000 C compiler generates incorrect code. The conditions are as follows: If you have a parameter which is a function, which points to a function, which points to an integer (double indirection is the key) improper code is generated for a case statement. See code below.

```
"C"
"68000"

extern fun1(),fun2();

bug(instr)
int (**instr)();
{
  int b;

  switch(b); {
    case 0: *instr = fun1; /* Code for this case is correct/
      break;
    case1: *instr = fun2; /* Here, because register A0 was loaded
      break; /* with a pointer to instr in case 0 the
      /* compiler does not bother reloading A0.
      /* So, if case 0 is not executed reg A0
      /* contains garbage.*/
  }
}

Also, any case after the first one has this problem.

```

Temporary solution:

Place a default case at the top of the case statement. This statement will always be executed and the compiler will "fall through" to the next test case. See below example.

```
"C"
"68000"

extern fun1(),fun2();

dummy(){} /*Declare dummy function. */

bug(instr)
int (**instr)();
{
  int b;

  switch(b) {
    default: *instr = dummy;
    case 0 : *instr = fun1;
      break;
    case 1 : *instr = fun2;
      break;
  }
}

```

```
}
}
```

The important thing here is that there is no "break" statement in the default case. This allows the compiler to test subsequent cases.

Signed off 08/25/86 in release 901.09

Number: D200033613 Product: 68000 C 64819 01.07

One-line description:
RTS rather than RTE generated to return from interrupt routine.

Problem:
Turning \$Interrupt on\$ does not generate a "return from exception" as specified in the manual.

```
"C"
"68000"

main()
{
  int j;
}

$INTERRUPT ON$
int_func()
{
  int l;          /* A RTS, rather than the specified RTE
  l = 5;          instruction will be generated. */
  return(1);
}
```

Temporary solution:
You can generate an assembly source file using the \$ASM_FILE ON\$ directive and then change the incorret RTS instructions to RTE instructions

Signed off 08/25/86 in release 901.09

Number: D200035816 Product: 68000 C 64819 01.07

Keywords: CODE GENERATOR

One-line description:
16 bit comparison on a 8 bit unsigned short field.

Problem:
IMPROPER CODE GENERATED FOR STATEMENT INVOLVING unsigned short VARIABLE UNLESS EXPLICITLY RE-CAST AS unsigned short.

```
main()
{
  static unsigned short digit_index;
  static unsigned short digit[12];
  int a,b;
  if (digit[digit_index]--){
  a=4;
```

- 68000 C -

```
b=4;}
else{
a=5;
b=5;}
}
IMPROPER CODE IS GENERATED FOR THE COMPARISON (ie THE COMPARISON IS DONE
ON 16 BITS (8 OF WHICH HAVE BEEN CLEARED) AGAINST #0FFFFH.
12/10/85: The problem also arises if you compare a constant against
an unsigned short. For example if you declared:
#define constant ~0
unsigned short var;
and later compared these two the compiler will zero out the upper byte
of the variable var and then compare it to FFFFH. Thus, the condition
is never met.
```

12/16/85: Another example of incorrect code being generated when a char variable is used in a test condition is as follows:

```
char a;
main()
{
  a = -1;
  if(a == -1)
    a = 'A';
}
```

Temporary solution:
IF THE LINE IN QUESTION IS CHANGED TO:

```
if ((unsigned short)digit[digit_index]--){
```

CORRECT CODE IS GENERATED ALTHOUGH digit[] HAS ALREADY BEEN DECLARED unsigned short.

12/10/85: Declare the constant as a short. In other words:

```
#define constant 0FFH.
```

12/16/85: If only 128 valid characters are required the variable can be declared as a short int.

Signed off 08/25/86 in release 901.09

Number: D200036624 Product: 68000 C 64819 01.07

One-line description:
Passing a complicated expression as a parameter may generate bad code.

Problem:
Type casting an address to a long, then anding or oring it with a constant value and passing the expression as a parameter to a function generates incorrect code. The following code demonstrates this problem:

```
"C"
"68000"
extern int extvar;
extern f();
badandor() {
  f(((long) &extvar & -2); /*Generates call to Zunsmult (unsigned mult)
                          instead of AND*/
```

- 68000 C -


```

} f((long) &extvar | -2); /*Generates long add instead of OR*/
}

```

Temporary solution:

Assign the expression to a temporary variable and pass the temporary to the function:

```

badandor() {
long temp;
temp = &extvar;
temp &= -2;
f(temp);
}

```

Signed off 08/25/86 in release 901.09

Number: D200036939 Product: 68000 C 64819 01.07

Keywords: PASS 1

One-line description:

Multiple warning's may cause messages to be intermixed.

Problem:

It appears the buffer for writing out warning messages is not cleared after a message is written. In the below program two warning messages are generated with the second containing information from the first.

```

"C"
"68000"

```

```

#define PETER 0
#define PETER 1
main(){
func();
}

```

The following warning messages are printed out.

```

511: Warning: variable assumed to be function returning integer.
513: Warning: duplicate macro name; new definition holds nteger.

```

Signed off 08/25/86 in release 901.09

Number: D200040667 Product: 68000 C 64819 01.07

Keywords: PASS 3

One-line description:

Pass 3 fails to detect relative jump address out-of-range.

Problem:

Pass 3 of the compilation process may fail to detect a relative jump which is out of range. In the test program submitted the relative jump is generated for an IF..THEN statement while the compiler option OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:

As a temporary work around disable the compiler option OPTIMIZE

- 68000 C -

around those sections of code which are suspect.

Signed off 08/25/86 in release 901.09

Number: D200041228 Product: 68000 C 64819 01.07

One-line description:

Problem with integer pointer in conditional statement.

Problem:

In the following example, two loads are performed, but no other code is generated to check for zero value.

```

"C"
"processor name"
#define NULL 0
fct(parm)
int *parm;
{
if (parm - NULL)
parm = 10;
}

```

Signed off 08/25/86 in release 901.09

Number: D200041830 Product: 68000 C 64819 01.07

One-line description:

Compiler calculating wrong offset to parameter.

Problem:

The following program generates incorrect code:

```

"C"
"Z8002"
dummy(output)
int (*output)();
{
int a;
(*output)(a);
}

rummy(output)
int (*output)();
{
(*output)(); /* the offset used into the stack does not */
/* point to the passed parameter */
}

```

Signed off 08/25/86 in release 901.09

Number: D200043943 Product: 68000 C 64819 01.07

Keywords: PASS 3

One-line description:

ASM reloc. and compiler reloc differ.

- 68000 C -

Problem:
Same as submitter.

Signed off 08/25/86 in release 901.09

Number: D200047514 Product: 68000 C 64819 01.07

One-line description:
TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 901.09

Number: D200043422 Product: 68000 C 64819 01.07

One-line description:
Compiler generating inefficient code for certain "switch" statements.

Signed off 08/25/86 in release 901.09

Number: D200048728 Product: 68000 C 300 64819S004 01.00

One-line description:
Incorrect code when hex values are bit or-ed and passed as parameters.

Problem:
When two hex values are bit or-ed together, and at least one of the values is greater than or equal to 0x8000, the compiler interprets the passed value as a long word instead of a word. The following code demonstrates the problem:

```
"C"
"68000"
$FAR$
$CALL_ABS_LONG$
$LIB_ABS_LONG$
extern sample();
main()
{
  sample(0x8000); (*Generates correct code*)
  sample(0x0080 | 0x1000 | 0x7fff); (*Generates correct code*)
  sample(0x0080 | 0x1000 | 0x8000); (*Generates incorrect code*)
}
```

Temporary solution:
There are two possible temporary solutions.

1. Use an explicit type cast.

```
main()
{
  sample((int)(0x0080 | 0x1000 | 0x8000)); (*Both expressions
  sample(0x0080 | 0x1000 | (int)0x8000); generate correct
  } code *)
```

2. Use a temporary variable.

```
main()
{
  int j;
  j = 0x8000;
  sample (0x0080 | 0x1000 | j);
}
```

Signed off 08/25/86 in release 401.10

Number: D200051193 Product: 68000 C 300 64819S004 01.00

Keywords: CODE GENERATOR

One-line description:
Incorrect code generated if fields are defined in a structure.

Problem:
The assembly code generated for the below C source is not correct. If any field of the structure is referenced the wrong offset is generated

by the assembler.

```
"C"
"68000"

main ()
  struct{
    short int a;
    unsigned : 4;
    unsigned fl 1;} s;

    (*s).a=1;          /* this line causes incorrect offset
                       to be generated. */
  }
```

Temporary solution:

Declare the bit fields first.

```
"C"
"68000"
main()
{
  struct {
    unsigned fl :1;
    unsigned   :4;
    short   a   ;
  } s;
```

Signed off 08/25/86 in release 401.10

Number: D200051243 Product: 68000 C 300 64819S004 01.00

One-line description:

++ and -- operators evaluated with improper precedence.

Problem:

According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

Example 1: array[index++] = 1;

Example 2: array[index] = 1;
index++;

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:

Separate the expression as shown in example 2.

Signed off 08/25/86 in release 401.10

Number: D200052266 Product: 68000 C 300 64819S004 00.00

Keywords: CODE GENERATOR

One-line description:

Incorrect opcode "MOV A,ACC" allowed by our assembler

Problem:

The instruction "MOV A,ACC" was assemble and emulated by our products; however, the Intel 8051 goes into the weeds at this instrction. At first glance the machine code in the assembler listing appears valid (MOV A,ACC ->0000 E5E0), but the bottom of page 8-35 in Intel's microcontroller handbook states: *MOV A,ACC is not a valid instruction.

Neither our manuals nor AMD's user manual mention this instruction.

Signed off 08/25/86 in release 401.10

Number: D200058966 Product: 68000 C 300 64819S004 01.00

One-line description:

Host compilers do not put absolute pats specifications in relocatables

Problem:

Host compilers do not specify the full path name in the relocatable file.

Signed off 08/25/86 in release 401.10

Number: D200048926 Product: 68000 C 300 64819S004 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 401.10

Number: 1650007054 Product: 68000 C 500 64819S001 01.40

One-line description:

Declaring 128 external functions causes compiler to bomb in code.

Signed off 08/25/86 in release 101.50

Number: D200015891 Product: 68000 C 500 64819S001 01.00

One-line description:

No error generated when an interrupt routine is explicitly called.

Signed off 08/25/86 in release 101.50

Number: D200016030 Product: 68000 C 500 64819S001 01.00

Keywords: CODE GENERATOR

One-line description:

Wrong addressing mode used with \$BASE_PAGE\$ on in ASM68000 file.

Problem:

In the ASM68000 source generated by the \$ASM_FILE\$, the wrong addressing mode is used when the \$BASE_PAGE\$ directive is on.

Signed off 08/25/86 in release 101.50

Number: D200016071 Product: 68000 C 500 64819S001 01.00

Keywords: CODE GENERATOR

One-line description:

The wrong byte is accessed when a union is defined within a structure.

Problem:

```
"C"
"68000"
struct {
    char ch;
    union {
        char ch1;
        char ch2;
    } un;
} *str;
main() {
    str->un.ch1=1;
    str->un.ch2=2;
}
```

The variables "ch1" and "ch2" in the above example should be at un + 1. Although, in the expanded listing you see they are accessed at un + 2 as if the field "ch" was a 16 bit datatype.

Signed off 08/25/86 in release 101.50

Number: D200016600 Product: 68000 C 500 64819S001 01.10

Keywords: CODE GENERATOR

One-line description:

Structure with an odd-numbered char or short array gens. wrong code.

Problem:

The following code uses an incorrect offset from A0:

```
"C"
"68000"
struct { char name[3];
        char ext; } *ptr;
sub()
{
    ptr->ext = 'a';
}
```

The offset generated is 4[A0] when assigning 'a' to "ext" when it should be 3[A0]. This is not a problem with an even sized array or with an integer array.

Signed off 08/25/86 in release 101.50

Number: D200031013 Product: 68000 C 500 64819S001 01.10

Keywords: CODE GENERATOR

One-line description:

Incorrect code generated if fields are defined in a structure.

Problem:

The assembly code generated for the below C source is not correct. If any field of the structure is referenced the wrong offset is generated by the assembler.

```
"C"
"68000"
main ()
    struct{
        short int a;
        unsigned : 4;
        unsigned f1 1;} s;

    (*s).a=1; /* this line causes incorrect offset
              to be generated. */
}
```

Temporary solution:

Declare the bit fields first.

```
"C"
"68000"
main()
{
    struct {
        unsigned f1 :1;
        unsigned :4;
        short a ;
    }
```

```
    } s;
```

Signed off 08/25/86 in release 101.50

```
Number: D200031039 Product: 68000 C      500 64819S001      01.10
```

Keywords: CODE GENERATOR

One-line description:

Variable may not be defined before an array in a structure.

Problem:

In a structure which includes an array(s) the array(s) must be defined before any other variable. If the other variable is declared before the array incorrect code will be generated when the array is dereferenced.

```
"C"
"68000"
```

```
struct a{
    char *p;
    char i[2];
}
main()
{
    a *ad;
    ad->i =1;                /*Incorrect code will generated. */
}

```

Temporary solution:

Declare all arrays first.

```
"C"
"68000"
```

```
struct a{
    char i[2];
    char *p;
}

```

```
main()
{
    struct a *ad;
    ad->i=1;
}
END
**
```

Signed off 08/25/86 in release 101.50

```
Number: D200031336 Product: 68000 C      500 64819S001      01.10
```

One-line description:

++ and -- operators evaluated with improper precedence.

Problem:

According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

- 68000 C -

```
Example 1: array[index++] = 1;
```

```
Example 2: array[index] = 1;
```

```
           index++;
```

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:

Separate the expression as shown in example 2.

Signed off 08/25/86 in release 101.50

```
Number: D200033142 Product: 68000 C      500 64819S001      01.10
```

One-line description:

Comparing character to zero in while loop generates incorrect code.

Problem:

If you compare a character variable to zero in a while loop, incorrect code is generated. The following code demonstrates the problem.

```
"C"
"6809"
```

```
proc()
{
    char timeout = 10;

    while(timeout--);    /* Code generated here causes infinite loop.
}

```

The code generated for the while loop clears the A register and then compares the D register to -1. Therefore the condition is never met.

Temporary solution:

Declare the variable used in the test condition as an integer.

```
"C"
"6809"
```

```
proc()
{
    int timeout = 10;

    while (timeout--);
}

```

Signed off 08/25/86 in release 101.50

```
Number: D200035824 Product: 68000 C      500 64819S001      01.10
```

Keywords: CODE GENERATOR

One-line description:

16 bit comparison on a 8 bit unsigned short field.

- 68000 C -

Problem:
IMPROPER CODE GENERATED FOR STATEMENT INVOLVING unsigned short
VARIABLE UNLESS EXPLICITLY RE-CAST AS unsigned short.

```
main()
{
static unsigned short digit_index;
static unsigned short digit[12];
int a,b;
if (digit[digit_index]--){
a=4;
b=4;}
else{
a=5;
b=5;}
}
IMPROPER CODE IS GENERATED FOR THE COMPARISON (ie THE COMPARISON IS DONE
ON 16 BITS (8 OF WHICH HAVE BEEN CLEARED) AGAINST #0FFFFH.
12/10/85: The problem also arises if you compare a constant against
an unsigned short. For example if you declared:
#define constant ~0
unsigned short var;
and later compared these two the compiler will zero out the upper byte
of the variable var and then compare it to FFFFH. Thus, the condition
is never met.
```

12/16/85: Another example of incorrect code being generated when a
char variable is used in a test condition is as follows:

```
char a;
main()
{
a = -1;
if(a == -1)
a = 'A';
}
```

Temporary solution:
IF THE LINE IN QUESTION IS CHANGED TO:

```
if ((unsigned short)digit[digit_index]--){
CORRECT CODE IS GENERATED ALTHOUGH digit[] HAS ALREADY BEEN
DECLARED unsigned short.
12/10/85: Declare the constant as a short. In other words:
#define constant OFFH.
12/16/85: If only 128 valid characters are required the variable can
be declared as a short int.
```

Signed off 08/25/86 in release 101.50

Number: D200036632 Product: 68000 C 500 64819S001 01.20

One-line description:
Passing a complicated expression as a parameter may generate bad code.

Problem:

- 68000 C -

Type casting an address to a long, then anding or oring it with a
constant value and passing the expression as a parameter to a function
generates incorrect code. The following code demonstrates this problem:

```
"C"
"68000"
extern int extvar;
extern f();
badandor() {
f((long) &extvar & -2); /*Generates call to Zunsmult (unsigned mult)
instead of AND*/
f((long) &extvar | -2); /*Generates long add instead of OR*/
}
```

Temporary solution:
Assign the expression to a temporary variable and pass the temporary
to the function:

```
badandor() {
long temp;
temp = &extvar;
temp &= -2;
f(temp);
}
```

Signed off 08/25/86 in release 101.50

Number: D200037077 Product: 68000 C 500 64819S001 01.20

Keywords: PASS 3

One-line description:
Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:
Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect
data being output to the list file. In selected cases, machine code
will be incorrectly listed. For example, consider the following
Pascal program.

```
$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;

VAR
a, b : BOOLEAN;

PROCEDURE one;

BEGIN
a := b;
END;
```

In the example listed above, the output file will denote machine code
of the form FFFFC0001 for one of the generated assembly statements.
The correct value should be C8000001. This problem is caused by an

- 68000 C -

incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 101.50

Number: D200040675 Product: 68000 C 500 64819S001 01.20

Keywords: PASS 3

One-line description:

Pass 3 fails to detect relative jump address out-of-range.

Problem:

Pass 3 of the compilation process may fail to detect a relative jump which is out of range. In the test program submitted the relative jump is generated for an IF..THEN statement while the compiler option OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:

As a temporary work around disable the compiler option OPTIMIZE around those sections of code which are suspect.

Signed off 08/25/86 in release 101.50

Number: D200041236 Product: 68000 C 500 64819S001 01.20

One-line description:

Problem with integer pointer in conditional statement.

Problem:

In the following example, two loads are performed, but no other code is generated to check for zero value.

```
"C"
"processor name"
#define NULL 0
fct(parm)
int *parm;
{
    if (parm - NULL)
        parm = 10;
}
```

Signed off 08/25/86 in release 101.50

Number: D200041848 Product: 68000 C 500 64819S001 01.20

One-line description:

Compiler calculating wrong offset to parameter.

Problem:

The following program generates incorrect code:

```
"C"
"Z8002"
dummy(output)
```

- 68000 C -

```
int (*output)();
{
    int a;
    (*output)(a);
}

rummy(output)
int (*output)();
{
    (*output)(); /* the offset used into the stack does not */
                /* point to the passed parameter */
}
}
```

Signed off 08/25/86 in release 101.50

Number: D200044032 Product: 68000 C 500 64819S001 01.20

Keywords: PASS 3

One-line description:

ASM reloc. and compiler reloc differ.

Problem:

Same as submitter.

Signed off 08/25/86 in release 101.50

Number: D200047522 Product: 68000 C 500 64819S001 01.20

One-line description:

TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 101.50

Number: D200048702 Product: 68000 C 500 64819S001 01.40

One-line description:

Incorrect code when hex values are bit or-ed and passed as parameters.

Problem:

When two hex values are bit or-ed together, and at least one of the values is greater than or equal to 0x8000, the compiler interprets the passed value as a long word instead of a word. The following code demonstrates the problem:

```
"C"
"68000"
$FAR$
$CALL_ABS_LONG$
$LIB_ABS_LONG$
extern sample();
main()
{
    sample(0x8000); (*Generates correct code*)
    sample(0x0080 | 0x1000 | 0x7fff); (*Generates correct code*)
    sample(0x0080 | 0x1000 | 0x8000); (*Generates incorrect code*)
}
```

- 68000 C -

Temporary solution:
There are two possible temporary solutions.

1. Use an explicit type cast.

```
main()
{
  sample((int)(0x0080 | 0x1000 | 0x8000));  (*Both expressions
  sample(0x0080 | 0x1000 | (int)0x8000);    generate correct
}                                             code *)
```

2. Use a temporary variable.

```
main()
{
  int j;
  j = 0x8000;
  sample (0x0080 | 0x1000 | j);
}
```

Signed off 08/25/86 in release 101.50

Number: D200049650 Product: 68000 C 500 64819S001 00.00

One-line description:
NO CROSS REFERENCE TABLE IS GENERATED

Problem:
"C" COMPILERS DO NOT GENERATE A CROSS REFERENCE TABLE ON THE
VAX.

Temporary solution:
NONE KNOWN AT PRESENT

Signed off 04/18/86 in release 101.50

Number: D200058941 Product: 68000 C 500 64819S001 01.40

One-line description:
Host compilers do not put absolute pats specifications in relocatables

Problem:
Host compilers do not specify the full path name in the
relocatable file.

Signed off 08/25/86 in release 101.50

Number: D200048900 Product: 68000 C 500 64819S001 00.00

One-line description:
Linker output file should use alternate file extension.

Signed off 08/25/86 in release 101.50

Number: D200015909 Product: 68000 C VAX 64819S003 01.00

One-line description:
No error code generated when an interrupt is explicitly called.

Signed off 08/25/86 in release 301.80

Number: D200016022 Product: 68000 C VAX 64819S003 01.00

Keywords: CODE GENERATOR

One-line description:
Wrong addressing mode used with \$BASE_PAGE\$ on in ASM68000 file.

Problem:
In the ASM68000 source generated by the \$ASM_FILE\$, the wrong address-
ing mode is used when the \$BASE_PAGE\$ directive is on.

Signed off 08/25/86 in release 301.80

Number: D200016063 Product: 68000 C VAX 64819S003 01.00

Keywords: CODE GENERATOR

One-line description:
The wrong byte is accessed when a union is defined within a structure.

Problem:
"C"
"68000"
struct {
 char ch;
 union {
 char ch1;
 char ch2;
 } un;
} *str;
main() {
 str->un.ch1=1;
 str->un.ch2=2;
}

The variables "ch1" and "ch2" in the above example should be at un + 1.
Although, in the expanded listing you see they are accessed at un + 2 as
if the field "ch" was a 16 bit datatype.

Signed off 08/25/86 in release 301.80

Number: D200016618 Product: 68000 C VAX 64819S003 01.10

Keywords: CODE GENERATOR

One-line description:
Structure with an odd-numbered char or short array gens. wrong code.

Problem:
The following code uses an incorrect offset from A0:
"C"


```
"68000"
struct { char name[3];
        char ext; } *ptr;
sub()
{
    ptr->ext = 'a';
}
```

The offset generated is 4[A0] when assigning 'a' to "ext" when it should be 3[A0]. This is not a problem with an even sized array or with an integer array.

Signed off 08/25/86 in release 301.80

Number: D200031021 Product: 68000 C VAX 64819S003 01.20

Keywords: CODE GENERATOR

One-line description:
Incorrect code generated if fields are defined in a structure.

Problem:
The assembly code generated for the below C source is not correct. If any field of the structure is referenced the wrong offset is generated by the assembler.

"C"
"68000"

```
main ()
{
    struct{
        short int a;
        unsigned : 4;
        unsigned f1 i;} s;

    (*s).a=1;                /* this line causes incorrect offset
                               to be generated. */
}
```

Temporary solution:
Declare the bit fields first.

```
"C"
"68000"
main()
{
    struct {
        unsigned f1 :1;
        unsigned :4;
        short a ;
    } s;
```

Signed off 08/25/86 in release 301.80

Number: D200031047 Product: 68000 C VAX 64819S003 01.20

Keywords: CODE GENERATOR

One-line description:
Variable may not be defined before an array in a structure.

Problem:
In a structure which includes an array(s) the array(s) must be defined before any other variable. If the other variable is declared before the array incorrect code will be generated when the array is dereferenced.

"C"
"68000"

```
struct a{
    char *p;
    char i[2];
}
main()
{
    a *ad;
    ad->i =1;                /*Incorrect code will generated. */
}
```

Temporary solution:
Declare all arrays first.

"C"
"68000"

```
struct a{
    char i[2];
    char *p;
}
main()
{
    struct a *ad;
    ad->i=1;
}
END
**
```

Signed off 08/25/86 in release 301.80

Number: D200031344 Product: 68000 C VAX 64819S003 01.20

One-line description:
++ and -- operators evaluated with improper precedence.

Problem:
According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

```
Example 1: array[index++] = 1;
Example 2: array[index] = 1;
           index++;
```

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements

are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:
Separate the expression as shown in example 2.

Signed off 08/25/86 in release 301.80

Number: D200033159 Product: 68000 C VAX 64819S003 01.20

One-line description:
Comparing character to zero in while loop generates incorrect code.

Problem:
If you compare a character variable to zero in a while loop, incorrect code is generated. The following code demonstrates the problem.

```
"C"
"6809"
```

```
proc()
{
    char timeout = 10;

    while(timeout--);    /* Code generated here causes infinite loop.
}

```

The code generated for the while loop clears the A register and then compares the D register to -1. Therefore the condition is never met.

Temporary solution:
Declare the variable used in the test condition as an integer.

```
"C"
"6809"
```

```
proc()
{
    int timeout = 10;

    while (timeout--);
}

```

Signed off 08/25/86 in release 301.80

Number: D200035832 Product: 68000 C VAX 64819S003 01.20

Keywords: CODE GENERATOR

One-line description:
16 bit comparison on a 8 bit unsigned short field.

Problem:
IMPROPER CODE GENERATED FOR STATEMENT INVOLVING unsigned short VARIABLE UNLESS EXPLICITLY RE-CAST AS unsigned short.

```
main()
{

```

- 68000 C -

```
static unsigned short digit_index;
static unsigned short digit[12];
int a,b;
if (digit[digit_index]--){
    a=4;
    b=4;}
else{
    a=5;
    b=5;}
}

```

IMPROPER CODE IS GENERATED FOR THE COMPARISON (ie THE COMPARISON IS DONE ON 16 BITS (8 OF WHICH HAVE BEEN CLEARED) AGAINST #0FFFFH.

12/10/85: The problem also arises if you compare a constant against an unsigned short. For example if you declared:
#define constant ~0
unsigned short var;
and later compared these two the compiler will zero out the upper byte of the variable var and then compare it to FFFFH. Thus, the condition is never met.

12/16/85: Another example of incorrect code being generated when a char variable is used in a test condition is as follows:

```
char a;
main()
{
    a = -1;
    if(a == -1)
        a = 'A';
}

```

Temporary solution:
IF THE LINE IN QUESTION IS CHANGED TO:

```
if ((unsigned short)digit[digit_index]--){

```

CORRECT CODE IS GENERATED ALTHOUGH digit[] HAS ALREADY BEEN DECLARED unsigned short.

12/10/85: Declare the constant as a short. In other words:
#define constant 0FFH.

12/16/85: If only 128 valid characters are required the variable can be declared as a short int.

Signed off 08/25/86 in release 301.80

Number: D200036640 Product: 68000 C VAX 64819S003 01.20

One-line description:
Passing a complicated expression as a parameter may generate bad code.

Problem:
Type casting an address to a long, then anding or oring it with a constant value and passing the expression as a parameter to a function generates incorrect code. The following code demonstrates this problem:

```
"C"
"68000"
```

- 68000 C -

```
extern int extvar;
extern f();
badandor() {
    f((long) &extvar & -2); /*Generates call to Zunsmult (unsigned mult)
                           instead of AND*/
    f((long) &extvar | -2); /*Generates long add instead of OR*/
}
```

Temporary solution:
Assign the expression to a temporary variable and pass the temporary to the function:

```
badandor() {
    long temp;
    temp = &extvar;
    temp &= -2;
    f(temp);
}
```

Signed off 08/25/86 in release 301.80

Number: D200037085 Product: 68000 C VAX 64819S003 01.20

Keywords: PASS 3

One-line description:
Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:
Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect data being output to the list file. In selected cases, machine code will be incorrectly listed. For example, consider the following Pascal program.

```
$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;

VAR
    a, b : BOOLEAN;

PROCEDURE one;

BEGIN
    a := b;
END;
```

In the example listed above, the output file will denote machine code of the form FFFFC0001 for one of the generated assembly statements. The correct value should be C8000001. This problem is caused by an incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 301.80

- 68000 C -

Number: D200040683 Product: 68000 C VAX 64819S003 01.20

Keywords: PASS 3

One-line description:
Pass 3 fails to detect relative jump address out-of-range.

Problem:
Pass 3 of the compilation process may fail to detect a relative jump which is out of range. In the test program submitted the relative jump is generated for an IF..THEN statement while the compiler option OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:
As a temporary work around disable the compiler option OPTIMIZE around those sections of code which are suspect.

Signed off 08/25/86 in release 301.80

Number: D200041244 Product: 68000 C VAX 64819S003 01.20

One-line description:
Problem with integer pointer in conditional statement.

Problem:
In the following example, two loads are performed, but no other code is generated to check for zero value.

```
"C"
"processor name"
#define NULL 0
fct(parm)
int *parm;
{
    if (parm - NULL)
        parm = 10;
}
```

Signed off 08/25/86 in release 301.80

Number: D200041855 Product: 68000 C VAX 64819S003 01.20

One-line description:
Compiler calculating wrong offset to parameter.

Problem:
The following program generates incorrect code:
"C"
"Z8002"
dummy(output)
int (*output)();
{
 int a;
 (*output)(a);
}

- 68000 C -

```
rummy(output)
int (*output)();
{
    (*output)(); /* the offset used into the stack does not */
                /* point to the passed parameter          */
}
```

Signed off 08/25/86 in release 301.80

```
Number: D200044040 Product: 68000 C VAX 64819S003 01.20
```

Keywords: PASS 3

One-line description:
ASM reloc. and compiler reloc differ.

Problem:
Same as submitter.

Signed off 08/25/86 in release 301.80

```
Number: D200045856 Product: 68000 C VAX 64819S003 01.20
```

One-line description:
Title description is incorrect.

Signed off 08/25/86 in release 301.80

```
Number: D200045922 Product: 68000 C VAX 64819S003 01.20
```

One-line description:
Title description is incorrect.

Signed off 08/25/86 in release 301.80

```
Number: D200047530 Product: 68000 C VAX 64819S003 01.20
```

One-line description:
TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 301.80

```
Number: D200047811 Product: 68000 C VAX 64819S003 01.20
```

One-line description:
Illegal instruction being generated by compiler.

Problem:
The following program will cause the C compiler to generate an illegal assembly instruction.

```
"C"
"68000"
```

```
proc(s)
char s[];
{
    int i;
```

- 68000 C -

```
s[i] = "\0"; /* A MOVE.B A3,... will be generated. Cannot
              use .B with address register as the source. */
```

Temporary solution:
Do use a string assignment (ie use single quotes.)

```
"C"
"68000"
```

```
proc(s)
char s[];
```

```
{
    int i;
    s[i] = '\0';
}
```

Signed off 08/25/86 in release 301.80

```
Number: D200048710 Product: 68000 C VAX 64819S003 01.50
```

One-line description:
Incorrect code when hex values are bit or-ed and passed as parameters.

Problem:
When two hex values are bit or-ed together, and at least one of the values is greater than or equal to 0x8000, the compiler interprets the passed value as a long word instead of a word. The following code demonstrates the problem:

```
"C"
"68000"
$FAR$
$CALL_ABS_LONG$
$LIB_ABS_LONG$
extern sample();
main()
{
    sample(0x8000); /*Generates correct code*
    sample(0x0080 | 0x1000 | 0x7fff); /*Generates correct code*
    sample(0x0080 | 0x1000 | 0x8000); /*Generates incorrect code*
}
```

Temporary solution:
There are two possible temporary solutions.

1. Use an explicit type cast.

```
main()
{
    sample((int)(0x0080 | 0x1000 | 0x8000)); /*Both expressions
    sample(0x0080 | 0x1000 | (int)0x8000); generate correct
                                                code */
}
```

2. Use a temporary variable.

```
main()
```

- 68000 C -

```

{
  int j;
  j = 0x8000;
  sample (0x0080 | 0x1000 | j);
}

```

Signed off 08/25/86 in release 301.80

Number: D200055137 Product: 68000 C VAX 64819S003 01.50

One-line description:

Compilation on the VAX using batch mode generates incorrect listing file

Problem:

The test files can be found on the VAX750 under user\$disk:[robin.hughes.rgalo.test]. The following test files were used:

1. MTINHST_C. - File which contains one error- a missing '}' on line 70
2. TMTINHST_C. - Error-free version of MTINHST_C.
3. MTOPNDF_C. - File which contains one error - missing declaration for integer 'j'
4. MTOPNDF_T_C. - Error-free version of MTOPNDF_C.

One logical name must be defined as follows to access the include files referenced by the test programs:

```
$define BSLN user$disk:[robin.hughes.wsbsln.baseline]
```

When the four files were compiled interactively, the two error-free versions generated correct listings. The first file (MTINHST_C.) generated an incomplete and incorrect listing file. The listing showed the include files inserted first, followed by "C", "8086" and a few other lines of the program. The output displayed on the screen looked like:

```

In pass1.
  70 else
    ^25
  136
    ^408
In C Nocode.
comp: C Nocode cannot recover from errors.

```

When the third file (MTOPNDF_C.) was compiled, the listing appeared fine except for the insertion a some strange control charaters.

These last two files were compiled in batch mode (file: user\$disk:[robin.hughes.rgalo.test]hughes.com). The first file (MTINHST_C.) generated a complete but incorrect listing. Although two errors were found (25 & 408) the line at the bottom stated that errors = 0. The include file expansion preceeded the "C" and "8086" in the listing, and lines like, #include filename, were still in the file. The error message was at line 72 of the listing instead of line 2472 were the '}' was actual missing. Finally the last

100 lines had useless numbers in the left margin.

When the third file (MTOPNDF_C.) was compiled, an incomplete listing was generated with the include file expansions listed first.

All of these tests were done on the VAX750 with the /e/v/o options.

This problem also occurs on the 68000.

Temporary solution:

No temporary solution available

Signed off 08/25/86 in release 301.80

Number: D200058958 Product: 68000 C VAX 64819S003 01.50

One-line description:

Host compilers do not put absolute pats specifications in relocatables

Problem:

Host compilers do not specify the full path name in the relocatable file.

Signed off 08/25/86 in release 301.80

Number: D200048918 Product: 68000 C VAX 64819S003 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 301.80

Number: D200054635 Product: 68000 C VAX 64819S003 01.50

Keywords: ENHANCEMENT

One-line description:

68010 directive not supported on the 9000.

Signed off 08/25/86 in release 301.80

Number: D200051011 Product: 68000 PASCAL 300 64815S004 01.00

One-line description:
Program causes compiler to hang up.

Problem:

A program containing a complicated expression causes the compiler to hang up in pass 2. No listing file is created and no error message is generated.

Temporary solution:

Break the complicated expression up into two or more simpler expressions.

Signed off 08/25/86 in release 401.10

Number: D200051110 Product: 68000 PASCAL 300 64815S004 01.00

Keywords: BOOLEAN

One-line description:
NOT(function) as boolean expression in "IF" statement doesn't work.

Problem:

```
"68000"
PROGRAM TEST;
FUNCTION X : BOOLEAN;EXTERNAL;
BEGIN
IF NOT X THEN ;    {THE RETURN VALUE IS NEVER TESTED.}
                  {COMPARE THE CODE TO:}
IF X THEN;
END.
```

Temporary solution:

Assign the function to an intermediate variable and test the variable.

Signed off 08/25/86 in release 401.10

Number: D200051508 Product: 68000 PASCAL 300 64815S004 01.00

Keywords: CODE GENERATOR

One-line description:
B := ABS(B) fails to write to the data area.

Problem:

```
VAR I : INTEGER; B : BYTE;

BEGIN
I := B;
IF I < 0 THEN
I := ABS(I);
  ^ Although I is complimented here, it is kept in the register
  and not rewritten to the data area.
```

- 68000 PASCAL -

Temporary solution:
IF I < 0 THEN I := -(I);

Signed off 08/25/86 in release 401.10

Number: D200051631 Product: 68000 PASCAL 300 64815S004 01.00

Keywords: PASS 2

One-line description:

K := K + K + K; causes too many pass 2 errors to continue.

Problem:

```
PROCEDURE TEST (VAR K : SIGNED_16);
BEGIN
K := K + K + K;    Causes 64000 to hang in pass 2. Causes the HOST to
                  abort in pass 2 with too many errors.
```

Temporary solution:

Use a multiply operator instead of 'n' adds.

"68000"

```
PROGRAM HANGS;

VAR PARAM : SIGNED_16;

PROCEDURE TEST(VAR K : SIGNED_16);

BEGIN
  K = 3*K;
END;

BEGIN { HANGS }
END. { HANGS }
```

Signed off 08/25/86 in release 401.10

Number: D200052597 Product: 68000 PASCAL 300 64815S004 01.00

One-line description:
Missing semicolon causes compiler to hang in Pass 1.

Problem:

The following code causes the 64000 to hang in pass 1. An error is generated on the hosts stating that parsing has stopped at a particular line number.

"68000"

```
PROGRAM MAIN;
TYPE
STRUCTURED= RECORD
  INT1:INTEGER;
  INT2:INTEGER;
END;
```

```
PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
```

- 68000 PASCAL -

```

VAR I:INTEGER;
BEGIN
I:=P1      <--This missing semicolon causes the problem
I:=P1.2;
I:=P2;
END;

```

```

BEGIN
END.

```

Temporary solution:
 If the compiler hangs, look for a statement without a semicolon.
 On the 64000, the status line will show which line of code it
 stopped on. On the hosts, the error message generated indicates
 which line of code parsing stopped on.

Signed off 08/25/86 in release 401.10

Number: D200058792 Product: 68000 PASCAL 300 64815S004 01.00

Keywords: PREPROCESSOR

One-line description:
 Preprocessor reports errors when symbols hp64000, vms or hpux w #if

Signed off 08/25/86 in release 401.10

Number: D200059220 Product: 68000 PASCAL 300 64815S004 01.00

One-line description:
 Host compilers do not put absolute pats specifications in relocatables

Problem:
 Host compilers do not specify the full path name in the
 relocatable file.

Signed off 08/25/86 in release 401.10

Number: D200048835 Product: 68000 PASCAL 300 64815S004 00.00

One-line description:
 Linker output file should use alternate file extension.

Signed off 08/25/86 in release 401.10

Number: 5000095687 Product: 68000 PASCAL 500 64815S001 01.10

Keywords: CASE STATEMENT

One-line description:
 Different code generated between Host and 64000 for case statement.

```

Problem:
VAR I : INTEGER;
CASE I OF
  1 : ;
  2 : ;
  32000 :
  END;
END.

```

This program generates a 3 line comparison on the 64000, but a 32000
 line lookup on the Host.

Temporary solution:
 None at this time.

Signed off 08/25/86 in release 101.40

Number: D200027664 Product: 68000 PASCAL 500 64815S001 01.10

One-line description:
 No form feed between the expanded listing and the cross reference table.

Problem:
 During compilation, with XREF option on, the compiler does not provide
 a form feed (FF) in the listing file. The XREF starts on the same page
 as the end of the listing. Also, the page number says 535 when it
 should be page 2.

Temporary solution:
 After compiling with the xref option, edit the expanded listing file
 and insert a "control L" before the beginning of the cross reference
 listing.

Signed off 08/25/86 in release 101.40

Number: D200030627 Product: 68000 PASCAL 500 64815S001 01.10

Keywords: BOOLEAN

One-line description:
 NOT(function) as boolean expression in "IF" statement doesn't work.

```

Problem:
"68000"
PROGRAM TEST;
FUNCTION X : BOOLEAN;EXTERNAL;
BEGIN
IF NOT X THEN ;      {THE RETURN VALUE IS NEVER TESTED.}
                    {COMPARE THE CODE TO;}
IF X THEN;

```

END.

Temporary solution:

Assign the function to an intermediate variable and test the variable.

Signed off 08/25/86 in release 101.40

 Number: D200034207 Product: 68000 PASCAL 500 64815S001 01.10

Keywords: CODE GENERATOR

One-line description:

B := ABS(B) fails to write to the data area.

Problem:

VAR I : INTEGER; B : BYTE;

BEGIN

I := B;

IF I < 0 THEN

I := ABS(I);

^ Although I is complimented here, it is kept in the register
and not rewritten to the data area.

Temporary solution:

IF I < 0 THEN I := -(I);

Signed off 08/25/86 in release 101.40

 Number: D200036947 Product: 68000 PASCAL 500 64815S001 01.20

Keywords: PASS 2

One-line description:

K := K + K + K; causes too many pass 2 errors to continue.

Problem:

PROCEDURE TEST (VAR K : SIGNED_16);

BEGIN

K := K + K + K; Causes 64000 to hang in pass 2. Causes the HOST to
abort in pass 2 with too many errors.

Temporary solution:

None at this time.

Signed off 08/25/86 in release 101.40

 Number: D200037010 Product: 68000 PASCAL 500 64815S001 01.20

Keywords: PASS 3

One-line description:

Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:

Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect
data being output to the list file. In selected cases, machine code

- 68000 PASCAL -

will be incorrectly listed. For example, consider the following
Pascal program.

```
$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;
```

VAR

a, b : BOOLEAN;

PROCEDURE one;

BEGIN

a := b;

END;

In the example listed above, the output file will denote machine code
of the form FFFFC00001 for one of the generated assembly statements.
The correct value should be C8000001. This problem is caused by an
incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 101.40

 Number: D200047431 Product: 68000 PASCAL 500 64815S001 01.20

One-line description:

TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 101.40

 Number: D200052571 Product: 68000 PASCAL 500 64815S001 01.30

One-line description:

Missing semicolon causes compiler to hang in Pass 1.

Problem:

The following code causes the 64000 to hang in pass 1. An error
is generated on the hosts stating that parsing has stopped at
a particular line number.

"68000"

PROGRAM MAIN;

TYPE

STRUCTURED= RECORD

INT1:INTEGER;

INT2:INTEGER;

END;

PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);

VAR I:INTEGER;

BEGIN

I:=P1 <--This missing semicolon causes the problem

I:=P1.2;

- 68000 PASCAL -


```
I:=P2;
END;
```

```
BEGIN
END.
```

Temporary solution:

If the compiler hangs, look for a statement without a semicolon. On the 64000, the status line will show which line of code it stopped on. On the hosts, the error message generated indicates which line of code parsing stopped on.

Signed off 08/25/86 in release 101.40

```
Number: D200058776 Product: 68000 PASCAL 500 64815S001 01.30
```

Keywords: PREPROCESSOR

One-line description:

Preprocessor reports errors when symbols hp64000, vms or hpux w #if

Signed off 08/25/86 in release 101.40

```
Number: D200059204 Product: 68000 PASCAL 500 64815S001 01.30
```

One-line description:

Host compilers do not put absolute pats specifications in relocatables

Problem:

Host compilers do not specify the full path name in the relocatable file.

Signed off 08/25/86 in release 101.40

```
Number: D200048819 Product: 68000 PASCAL 500 64815S001 00.00
```

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 101.40

```
Number: D200027672 Product: 68000 PASCAL VAX 64815S003 01.20
```

One-line description:

No form feed between the expanded listing and the cross reference table.

Problem:

During compilation, with XREF option on, the compiler does not provide a form feed (FF) in the listing file. The XREF starts on the same page as the end of the listing. Also, the page number says 535 when it should be page 2.

Temporary solution:

After compiling with the xref option, edit the expanded listing file and insert a "control L" before the beginning of the cross reference listing.

Signed off 08/25/86 in release 301.60

```
Number: D200030635 Product: 68000 PASCAL VAX 64815S003 01.20
```

Keywords: BOOLEAN

One-line description:

NOT(function) as boolean expression in "IF" statement doesn't work.

Problem:

```
"68000"
PROGRAM TEST;
FUNCTION X : BOOLEAN;EXTERNAL;
BEGIN
IF NOT X THEN ;    {THE RETURN VALUE IS NEVER TESTED.}
                  {COMPARE THE CODE TO:}

IF X THEN;
END.
```

Temporary solution:

Assign the function to an intermediate variable and test the variable.

Signed off 08/25/86 in release 301.60

```
Number: D200034215 Product: 68000 PASCAL VAX 64815S003 01.20
```

Keywords: CODE GENERATOR

One-line description:

B := ABS(B) fails to write to the data area.

Problem:

VAR I : INTEGER; B : BYTE;

BEGIN

I := B;

IF I < 0 THEN

I := ABS(I);

^ Although I is complimented here, it is kept in the register and not rewritten to the data area.

Temporary solution:
IF I < 0 THEN I := -(I);

Signed off 08/25/86 in release 301.60

Number: D200036954 Product: 68000 PASCAL VAX 64815S003 01.20

Keywords: PASS 2

One-line description:

K := K + K + K; causes too many pass 2 errors to continue.

Problem:

PROCEDURE TEST (VAR K : SIGNED_16);

BEGIN

K := K + K + K; Causes 64000 to hang in pass 2. Causes the HOST to abort in pass 2 with too many errors.

Temporary solution:

None at this time.

Signed off 08/25/86 in release 301.60

Number: D200037028 Product: 68000 PASCAL VAX 64815S003 01.20

Keywords: PASS 3

One-line description:

Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:

Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect data being output to the list file. In selected cases, machine code will be incorrectly listed. For example, consider the following Pascal program.

```
$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;

VAR
  a, b : BOOLEAN;

PROCEDURE one;

  BEGIN
    a := b;
  END;
```

In the example listed above, the output file will denote machine code of the form FFFFC00001 for one of the generated assembly statements. The correct value should be C8000001. This problem is caused by an incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

- 68000 PASCAL -

Signed off 08/25/86 in release 301.60

Number: D200047449 Product: 68000 PASCAL VAX 64815S003 01.20

One-line description:

TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 301.60

Number: D200050922 Product: 68000 PASCAL VAX 64815S003 01.30

One-line description:

Program causes compiler to hang up.

Problem:

A program containing a complicated expression causes the compiler to hang up in pass 2. No listing file is created and no error message is generated.

Temporary solution:

Break the complicated expression up into two or more simpler expressions.

Signed off 08/25/86 in release 301.60

Number: D200050955 Product: 68000 PASCAL VAX 64815S003 01.30

One-line description:

Compiler generates illegal 68000 instruction LEAMOVEM.L

Problem:

The following code causes the compiler to generate an illegal 68000 instruction:

```
"68000"
PROGRAM TEST;
CONST
  event_size = 8;
TYPE
  event_type = (cmd_msg, rsp_msg, data_msg);
  event_msg_type =
    RECORD
      CASE event_type OF
        cmd_msg : (cmd : ARRAY[0..event_size-1] OF BYTE);
        rsp_msg : (rsp : ARRAY[0..event_size-1] OF BYTE);
        data_msg : (data : UNSIGNED_32);
      END;
  event =
    RECORD
      type : BYTE;
      qualifier : BYTE;
      msg : event_msg_type;
      send_task : BYTE;
```

- 68000 PASCAL -

```

END;

VAR
event1 : event;
BEGIN
event1 := event(0);
LEAMOVEM.L00000H,A0 (* This is the expanded code showing
LEA DTEST,A1 the illegal instruction LEAMOVEM *)
MOVE.L [A0]+,[A1]+
MOVE.L [A0]+,[A1]+
MOVE.L [A0]+,[A1]+
END.

```

Temporary solution:
No known work around at this time.

Signed off 08/25/86 in release 301.60

Number: D200052589 Product: 68000 PASCAL VAX 64815S003 01.30

One-line description:
Missing semicolon causes compiler to hang in Pass 1.

Problem:
The following code causes the 64000 to hang in pass 1. An error is generated on the hosts stating that parsing has stopped at a particular line number.

```

"68000"
PROGRAM MAIN;
TYPE
STRUCTURED= RECORD
INT1:INTEGER;
INT2:INTEGER;
END;

```

```

PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
I:=P1 <--This missing semicolon causes the problem
I:=P1.2;
I:=P2;
END;

BEGIN
END.

```

Temporary solution:
If the compiler hangs, look for a statement without a semicolon. On the 64000, the status line will show which line of code it stopped on. On the hosts, the error message generated indicates which line of code parsing stopped on.

Signed off 08/25/86 in release 301.60

Number: D200058784 Product: 68000 PASCAL VAX 64815S003 01.30

Keywords: PREPROCESSOR

One-line description:
Preprocessor reports errors when symbols hp64000, vms or hpux w #if

Signed off 08/25/86 in release 301.60

Number: D200059212 Product: 68000 PASCAL VAX 64815S003 01.30

One-line description:
Host compilers do not put absolute pats specifications in relocatables

Problem:
Host compilers do not specify the full path name in the relocatable file.

Signed off 08/25/86 in release 301.60

Number: D200048827 Product: 68000 PASCAL VAX 64815S003 00.00

One-line description:
Linker output file should use alternate file extension.

Signed off 08/25/86 in release 301.60

Number: D200051359 Product: 68000 PASCAL VAX 64815S003 01.30

One-line description:
Request for date and time of link on linker output file.

Signed off 08/25/86 in release 301.60

Number: D200048306 Product: 6805/9 ASSEMB 300 64844S004 01.00

Keywords: MACRO

One-line description:

Conditional instr. .IF with rational oper. in Macro creates bad code

Problem:

The use of the conditional instruction, .IF, with rational operator (.EQ.,.NE.,.LT.,.GT.,.LE.,.GE.) in a macro functions incorrectly. The following program demonstrates this problem:

```

BUG          MACRO          &VAR
              .IF &VAR .LE. 0 SUB&&&&
              NOP
              NOP
SUB&&&&       NOP
              NOP
              MEND

              BUG 3
              BUG -1
              BUG 0
              END

```

Passing a 3 appears to create correct code, but 0 causes a ML error. Passing -1 to the MACRO creates code which doesn't call the subroutine. This is incorrect since -1 is less than 0. This same problem occurred with all the rational operators on all processors. The problem was consistent on the 64000, VAX, and 9000.

Signed off 08/25/86 in release 401.10

Number: D200053397 Product: 6805/9 ASSEMB 300 64844S004 01.00

One-line description:

Macro def. including .IF, within a IF causes assembler to stop code gen.

Problem:

If you have a ".IF" in a macro definition and that macro definition is within a conditional assembly "IF" then no code is generated. The program provided demonstrates the problem (see submitter text).

Temporary solution:

Pull the macro definition outside of the conditional if. No code will be generated for the definition.

"processor name"

```

ESSAI      EQU      0

MAC        MACRO
          .IF      ESSAI.EQ.0  FIN
LABEL     LD       A,0
FIN       MEND

```

```

IF      ESSAI
MAC
ENDIF

```

START LD A,3

Signed off 08/25/86 in release 401.10

Number: D200049288 Product: 6805/9 ASSEMB 300 64844S004 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 401.10

Number: 5000115097 Product: 6805/9 ASSEMB 500 64844S001 01.10

One-line description:

Passing an undefined parameter to a macro is not flagged as an error.

Problem:

Passing undefined parameters to a macro does not generate an error or warning with the hosted assemblers (VAX and 9000).

```

CONST      ORG      10H
           EQU      0
CONST_MAC  MACRO    &P1
           .IF     (&P1) .LT. 256 P_OK
           WHATEVER ;doesn't matter
           FCB     CONST,(&P1)
           MEND

           CONST_MAC  UNDEF_PARAM

```

In this example, no error will be generated for the undefined symbol UNDEF_PARAM; the 64000 assembler generates an error message.

Signed off 08/25/86 in release 101.40

Number: D200038273 Product: 6805/9 ASSEMB 500 64844S001 01.20

One-line description:

Variable declared BEXT generates incorrect record in absolute file.

Problem:

The following examples assemble and link without errors, but generate an incorrect record in the absolute file.

```

"6809"
      ORG      10H
      EXT      AAA
      BEXT     BBB
CCC   EQU      AAA+10H
      FDB     CCC
      FCB     BBB          /*Address is 0022h*/

"6809"
      ORG      20H
      GLB     AAA,BBB
AAA   FDB     1234H
BBB   FDB     5678H
      END

```

The absolute file looks like this:

```

Record# 2 size= 5
      4 bytes starting at 0010H
0030 0032 /*0032 should be 0022*/

```

```

Record# 3 size= 5
      4 bytes starting at 0020H
1234 5678

```

Temporary solution:

The absolute file will be correct if the first source file is modified in the following way:

```

"6809"
      ORG      10H
      EXT      AAA
      BEXT     BBB
      FDB     AAA+10H
      FCB     BBB
      END

```

Signed off 08/25/86 in release 101.40

Number: D200046896 Product: 6805/9 ASSEMB 500 64844S001 01.20

One-line description:

Assembler should denote an error on non-absolute .SET expressions.

Signed off 08/25/86 in release 101.40

Number: D200048280 Product: 6805/9 ASSEMB 500 64844S001 01.30

Keywords: MACRO

One-line description:

Conditional instr. .IF with rational oper. in Macro creates bad code

Problem:

The use of the conditional instruction, .IF, with rational operator (.EQ.,.NE.,.LT.,.GT.,.LE.,.GE.) in a macro functions incorrectly. The following program demonstrates this problem:

```

      BUG      MACRO    &VAR
           .IF &VAR .LE. 0 SUB&&&&
           NOP
           NOP
      SUB&&&&    NOP
           NOP
           MEND

           BUG 3
           BUG -1
           BUG 0
           END

```

Passing a 3 appears to create correct code, but 0 causes a ML error. Passing -1 to the MACRO creates code which doesn't call the subroutine. This is incorrect since -1 is less than 0. This same problem occurred with all the rational operators on all processors. The problem was consistent on the 64000, VAX, and 9000.

Signed off 08/25/86 in release 101.40

Number: D200053371 Product: 6805/9 ASSEMB 500 64844S001 01.30

One-line description:

Macro def. including .IF, within a IF causes assembler to stop code gen.

Problem:

If you have a ".IF" in a macro definition and that macro definition is within a conditional assembly "IF" then no code is generated. The program provided demonstrates the problem (see submitter text).

Temporary solution:

Pull the macro definition outside of the conditional if. No code will be generated for the definition.

"processor name"

```

ESSAI      EQU      0
MAC        MACRO
           .IF      ESSAI.EQ.0   FIN
LABEL     LD        A,0
FIN       MEND

           IF      ESSAI
           MAC
           ENDIF

START     LD        A,3

```

Signed off 08/25/86 in release 101.40

Number: D200055939 Product: 6805/9 ASSEMB 500 64844S001 01.30

One-line description:

Relative address is calculated incorrectly when macro call has null parm

Problem:

The assembler is not calculating an address correctly when a label is equated to "\$-LABEL".

"6809"

```

           PROG
           EXT      F_CMOSDOWN

WMEM     MACRO    &P1,&P2,&P3
           LDA     &P1
           .IF     "&P3" .NE. ""    WMEM2
           .GOTO   WMEM3
WMEM2    .NOP
           STA     &P2,&P3
WMEM3    .NOP
           MEND

```

```

WMEM     #0FFH,F_CMOSDOWN,,      COMMENT

```

- 6805/9 ASSEMB -

```

AUTORDST   HEX      11
L_AUTORDST EQU     $-AUTORDST
           END

```

If you call WMEM with the third parameter as a null and have a comment which is not delimited by a semi-colon the value for L_AUTORDST is incorrect.

Temporary solution:

Use "" to delimit a null parameter and/or delimit the comment with a semi-colon.

```

So, use    WMEM    #0FFH,F_CMOSDOWN,"",      ;COMMENT
instead of WMEM    #0FFH,F_CMOSDOWN,,      COMMENT

```

Signed off 08/25/86 in release 101.40

Number: D200049262 Product: 6805/9 ASSEMB 500 64844S001 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 101.40

- 6805/9 ASSEMB -

Number: D200038281 Product: 6805/9 ASSEMB VAX 64844S003 01.20

One-line description:

Variable declared BEXT generates incorrect record in absolute file.

Problem:

The following examples assemble and link without errors, but generate an incorrect record in the absolute file.

```

"6809"
      ORG      10H
      EXT      AAA
      BEXT     BBB
CCC   EQU      AAA+10H
      FDB      CCC
      FCB      BBB           /*Address is 0022h*/

```

```

"6809"
      ORG      20H
      GLB      AAA,BBB
AAA   FDB      1234H
BBB   FDB      5678H
      END

```

The absolute file looks like this:

```

Record# 2   size= 5
      4 bytes starting at 0010H
0030 0032           /*0032 should be 0022*/

```

```

Record# 3   size= 5
      4 bytes starting at 0020H
1234 5678

```

Temporary solution:

The absolute file will be correct if the first source file is modified in the following way:

```

"6809"
      ORG      10H
      EXT      AAA
      BEXT     BBB
      FDB      AAA+10H
      FCB      BBB
      END

```

Signed off 08/25/86 in release 301.60

Number: D200046904 Product: 6805/9 ASSEMB VAX 64844S003 01.20

One-line description:

Assembler should denote an error on non-absolute .SET expressions.

Signed off 08/25/86 in release 301.60

Number: D200048298 Product: 6805/9 ASSEMB VAX 64844S003 01.40

Keywords: MACRO

One-line description:

Conditional instr. .IF with rational oper. in Macro creates bad code

Problem:

The use of the conditional instruction, .IF, with rational operator (.EQ.,.NE.,.LT.,.GT.,.LE.,.GE.) in a macro functions incorrectly. The following program demonstrates this problem:

```

      BUG      MACRO      &VAR
                .IF &VAR .LE. 0 SUB&&&&
                NOP
                NOP
      SUB&&&&    NOP
                NOP
                MEND

      BUG 3
      BUG -1
      BUG 0
      END

```

Passing a 3 appears to create correct code, but 0 causes a ML error. Passing -1 to the MACRO creates code which doesn't call the subroutine. This is incorrect since -1 is less than 0. This same problem occurred with all the rational operators on all processors. The problem was consistent on the 64000, VAX, and 9000.

Signed off 08/25/86 in release 301.60

Number: D200053389 Product: 6805/9 ASSEMB VAX 64844S003 01.40

One-line description:

Macro def. including .IF, within a IF causes assembler to stop code gen.

Problem:

If you have a ".IF" in a macro definition and that macro definition is within a conditional assembly "IF" then no code is generated. The program provided demonstrates the problem (see submitter text).

Temporary solution:

Pull the macro definition outside of the conditional if. No code will be generated for the definition.

"processor name"

```

ESSAI   EQU      0

MAC     MACRO
        .IF     ESSAI.EQ.0   FIN
LABEL   LD       A,0
FIN     MEND

```

```

IF      ESSAI
MAC
ENDIF

```

```

START   LD      A,3

```

Signed off 08/25/86 in release 301.60

```

Number: D200049270 Product: 6805/9 ASSEMB VAX 64844S003 00.00

```

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 301.60

```

Number: D200013946 Product: 6809 C 64822 00.56

```

Keywords: PASS 1

One-line description:

No warning or err: taking the sizeof a struct var. not declared.

Problem:

The compiler should generate an error in the following code.

```

"C"
"6809"
main () {
    int y;
    y = sizeof(struct x);
}

```

If x is not declared or is declared as anything other than a structure, the program compiles with no error messages or warnings. It stores as the size zero bytes.

Signed off 08/25/86 in release 201.07

```

Number: D200027748 Product: 6809 C 64822 01.04

```

One-line description:

No form feed between the expanded listing and the cross reference table.

Problem:

During compilation, with XREF option on, the compiler does not provide a form feed (FF) in the listing file. The XREF starts on the same page as the end of the listing. Also, the page number says 535 when it should be page 2.

Temporary solution:

After compiling with the xref option, edit the expanded listing file and insert a "control L" before the beginning of the cross reference listing.

Signed off 08/25/86 in release 201.07

```

Number: D200029694 Product: 6809 C 64822 01.04

```

One-line description:

File fails to compile. Error 1113 is generated.

Problem:

The submitted file does not compile. In pass three error 1113 "Program counters disagree" is flagged. The file will not compile on any system.

Signed off 08/25/86 in release 201.07

Number: D200031419 Product: 6809 C 64822 01.04

One-line description:

++ and -- operators evaluated with improper precedence.

Problem:

According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

Example 1: array[index++] = 1;

Example 2: array[index] = 1;

index++;

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:

Separate the expression as shown in example 2.

Signed off 08/25/86 in release 201.07

Number: D200032391 Product: 6809 C 64822 01.04

One-line description:

Comparing character to zero in while loop generates incorrect code.

Problem:

If you compare a character variable to zero in a while loop, incorrect code is generated. The following code demonstrates the problem.

"C"
"6809"

```
proc()
{
    char timeout = 10;

    while(timeout--); /* Code generated here causes infinite loop.
}
```

The code generated for the while loop clears the A register and then compares the D register to -1. Therefore the condition is never met.

Temporary solution:

Declare the variable used in the test condition as an integer.

"C"
"6809"

```
proc()
{
    int timeout = 10;

    while (timeout--);
}
```

Signed off 08/25/86 in release 201.07

Number: D200035865 Product: 6809 C 64822 01.04

Keywords: CODE GENERATOR

One-line description:

16 bit comparison on a 8 bit unsigned short field.

Problem:

IMPROPER CODE GENERATED FOR STATEMENT INVOLVING unsigned short VARIABLE UNLESS EXPLICITLY RE-CAST AS unsigned short.

```
main()
{
    static unsigned short digit_index;
    static unsigned short digit[12];
    int a,b;
    if (digit[digit_index--]){
        a=4;
        b=4;}
    else{
        a=5;
        b=5;}
}
```

IMPROPER CODE IS GENERATED FOR THE COMPARISON (ie THE COMPARISON IS DONE ON 16 BITS (8 OF WHICH HAVE BEEN CLEARED) AGAINST #OFFFH.

12/10/85: The problem also arises if you compare a constant against an unsigned short. For example if you declared:

```
#define constant ~0
unsigned short var;
and later compared these two the compiler will zero out the upper byte of the variable var and then compare it to FFFFH. Thus, the condition is never met.
```

12/16/85: Another example of incorrect code being generated when a char variable is used in a test condition is as follows:

```
char a;
main()
{
    a = -1;
    if(a == -1)
        a = 'A';
}
```

Temporary solution:

IF THE LINE IN QUESTION IS CHANGED TO:

```
if ((unsigned short)digit[digit_index--]){
```

CORRECT CODE IS GENERATED ALTHOUGH digit[] HAS ALREADY BEEN DECLARED unsigned short.

12/10/85: Declare the constant as a short. In other words:

```
#define constant OFFH.
```

12/16/85: If only 128 valid characters are required the variable can be declared as a short int.

12/16/85: If only 128 valid characters are required the variable can

be declared as a short int.

Signed off 08/25/86 in release 201.07

Number: D200040758 Product: 6809 C 64822 01.05

Keywords: PASS 3

One-line description:

Pass 3 fails to detect relative jump address out-of-range.

Problem:

Pass 3 of the compilation process may fail to detect a relative jump which is out of range. In the test program submitted the relative jump is generated for an IF..THEN statement while the compiler option OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:

As a temporary work around disable the compiler option OPTIMIZE around those sections of code which are suspect.

Signed off 08/25/86 in release 201.07

Number: D200041327 Product: 6809 C 64822 01.05

One-line description:

Problem with integer pointer in conditional statement.

Problem:

In the following example, two loads are performed, but no other code is generated to check for zero value.

```
"C"
"processor name"
#define NULL 0
fct(parm)
int *parm;
{
    if (parm - NULL)
        parm = 10;
}
```

Signed off 08/25/86 in release 201.07

Number: D200045245 Product: 6809 C 64822 01.05

One-line description:

DIFFERENT BUT EQUAL OBJECT CODE GENERATED ON 64000 THAN IN THE UNIX ENV.

Problem:

THE 6809 COMPILER MAY GENERATE DIFFERENT BUT EQUAL CODE IN THE 64000 ENVIRONMENT THAN THE HP-UX OR VMS ENVIRONMENTS.

THIS CODE IS ACTUALLY EQUAL IN IT'S RESULTS BUT WILL SHOW DIFFERENCES IF COMPAIRED.

EXAMPLE: THIS COULD RESULT FROM MATH OPERATIONS TAKING PLACE IN A

- 6809 C -

DIFFERENT ORDER - THE RESULT WILL BE THE SAME BUT THE CODE DIFFERENT.

Signed off 08/25/86 in release 201.07

Number: D200047605 Product: 6809 C 64822 01.05

One-line description:

TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 201.07

- 6809 C -

Number: D200050278 Product: 6809 C 300 64822S004 01.00

Keywords: PASS 1

One-line description:

Incorrect code is generated when complementing a parm. in a return stmt.

Problem:

In the following program the incorrect code is generated for the complement of the parameter to be returned.

```
"C"
"6809"
unsigned short bug()
{
    return(~x);
}
```

The compiler generates a "NEGB" when it should be a "COMB"

Temporary solution:

Set up a temporary variable and assign the complement of the parameter to it and then return the temporary. For example,

```
unsigned short temp;
temp = ~x;
return temp;
```

Signed off 08/25/86 in release 401.10

Number: D200051078 Product: 6809 C 300 64822S004 01.00

One-line description:

File fails to compile. Error 1113 is generated.

Problem:

The submitted file does not compile. In pass three error 1113 "Program counters disagree" is flagged. The file will not compile on any system.

Temporary solution:

No known temporary solution

Signed off 08/25/86 in release 401.10

Number: D200051292 Product: 6809 C 300 64822S004 01.00

One-line description:

++ and -- operators evaluated with improper precedence.

Problem:

According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

```
Example 1: array[index++] = 1;
Example 2: array[index] = 1;
           index++;
```

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before

setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:

Separate the expression as shown in example 2.

Signed off 08/25/86 in release 401.10

Number: D200052290 Product: 6809 C 300 64822S004 00.00

Keywords: CODE GENERATOR

One-line description:

Incorrect opcode "MOV A,ACC" allowed by our assembler

Problem:

The instruction "MOV A,ACC" was assemble and emulated by our products; however, the Intel 8051 goes into the weeds at this instruction. At first glance the machine code in the assembler listing appears valid (MOV A,ACC ->0000 E5E0), but the bottom of page 8-35 in Intel's microcontroller handbook states: *MOV A,ACC is not a valid instruction.

Neither our manuals nor AMD's user manual mention this instruction.

Signed off 08/25/86 in release 401.10

Number: D200059055 Product: 6809 C 300 64822S004 01.00

One-line description:

Host compilers do not put absolute pats specifications in relocatables

Problem:

Host compilers do not specify the full path name in the relocatable file.

Signed off 08/25/86 in release 401.10

Number: D200049015 Product: 6809 C 300 64822S004 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 401.10

SRB detail reports as of 08/25/86

Page: 111

Number: D200049742 Product: 6809 C 500 64822S001 00.00

One-line description:
NO CROSS REFERENCE TABLE IS GENERATED

Problem:
"C" COMPILERS DO NOT GENERATE A CROSS REFERENCE TABLE ON THE
VAX.

Temporary solution:
NONE KNOWN AT PRESENT

Signed off 04/18/86 in release 101.30

- 6809 C -

SRB detail reports as of 08/25/86

Page: 112

Number: D200015651 Product: 6809 C VAX 64822S003 01.00

Keywords: PASS 1

One-line description:
Incorrect code is generated when complementing a parm. in a return stmt.

Problem:
In the following program the incorrect code is generated for the complement of the parameter to be returned.

```
"C"  
"6809"  
unsigned short bug()  
{  
    return(~x);  
}
```

The compiler generates a "NEGB" when it should be a "COMB"

Temporary solution:
Set up a temporary variable and assign the complement of the parameter to it and then return the temporary. For example,
 unsigned short temp;
 temp = ~x;
 return temp;

Signed off 08/25/86 in release 301.50

Number: D200029710 Product: 6809 C VAX 64822S003 01.00

One-line description:
File fails to compile. Error 1113 is generated.

Problem:
The submitted file does not compile. In pass three error 1113
"Program counters disagree" is flagged. The file will not compile on
any system.

Signed off 08/25/86 in release 301.50

Number: D200035881 Product: 6809 C VAX 64822S003 00.00

Keywords: CODE GENERATOR

One-line description:
16 bit comparison on a 8 bit unsigned short field.

Problem:
IMPROPER CODE GENERATED FOR STATEMENT INVOLVING unsigned short
VARIABLE UNLESS EXPLICITLY RE-CAST AS unsigned short.

```
main()  
{  
    static unsigned short digit_index;  
    static unsigned short digit[12];  
    int a,b;
```

- 6809 C -

```
if (digit[digit_index]--){
a=4;
b=4;}
else{
a=5;
b=5;}
}
```

IMPROPER CODE IS GENERATED FOR THE COMPARISON (ie THE COMPARISON IS DONE ON 16 BITS (8 OF WHICH HAVE BEEN CLEARED) AGAINST #0FFFFH.

12/10/85: The problem also arises if you compare a constant against an unsigned short. For example if you declared:

```
#define constant ~0
unsigned short var;
and later compared these two the compiler will zero out the upper byte of the variable var and then compare it to FFFFH. Thus, the condition is never met.
```

12/16/85: Another example of incorrect code being generated when a char variable is used in a test condition is as follows:

```
char a;
main()
{
  a = -1;
  if(a == -1)
    a = 'A';
}
```

Temporary solution:
 IF THE LINE IN QUESTION IS CHANGED TO:

```
if ((unsigned short)digit[digit_index]--){
```

CORRECT CODE IS GENERATED ALTHOUGH digit[] HAS ALREADY BEEN DECLARED unsigned short.

12/10/85: Declare the constant as a short. In other words:

```
#define constant 0FFH.
```

12/16/85: If only 128 valid characters are required the variable can be declared as a short int.

Signed off 08/25/86 in release 301.50

Number: D200037143 Product: 6809 C VAX 64822S003 00.00

Keywords: PASS 3

One-line description:
 Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Signed off 08/25/86 in release 301.50

Number: D200040774 Product: 6809 C VAX 64822S003 00.00

Keywords: PASS 3

One-line description:
 Pass 3 fails to detect relative jump address out-of-range.

Problem:

Pass 3 of the compilation process may fail to detect a relative jump which is out of range. In the test program submitted the relative jump is generated for an IF..THEN statement while the compiler option OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:

As a temporary work around disable the compiler option OPTIMIZE around those sections of code which are suspect.

Signed off 08/25/86 in release 301.50

Number: D200041343 Product: 6809 C VAX 64822S003 00.00

One-line description:
 Problem with integer pointer in conditional statement.

Problem:

In the following example, two loads are performed, but no other code is generated to check for zero value.

```
"C"
"processor name"
#define NULL 0
fct(parm)
int *parm;
{
  if (parm - NULL)
    parm = 10;
}
```

Signed off 08/25/86 in release 301.50

Number: D200045989 Product: 6809 C VAX 64822S003 00.00

One-line description:
 Title description is incorrect.

Signed off 08/25/86 in release 301.50

Number: D200047621 Product: 6809 C VAX 64822S003 00.00

One-line description:
 TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 301.50

Number: D200051284 Product: 6809 C VAX 64822S003 01.20

One-line description:
 ++ and -- operators evaluated with improper precedence.

Problem:

According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

Example 1: array[index++] = 1;

Example 2: `array[index] = 1;`
`index++;`

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:

Separate the expression as shown in example 2.

Signed off 08/25/86 in release 301.50

Number: D200055160 Product: 6809 C VAX 64822S003 01.20

One-line description:

Compilation on the VAX using batch mode generates incorrect listing file

Problem:

The test files can be found on the VAX750 under user\$disk:[robin.hughes.rgalo.test]. The following test files were used:

1. MTINHST_C. - File which contains one error- a missing '}' on line 70
2. TMTINHST_C. - Error-free version of MTINHST_C.
3. MTOPNDF_C. - File which contains one error - missing declaration for integer 'j'
4. MTOPNDF_T_C. - Error-free version of MTOPNDF_C.

One logical name must be defined as follows to access the include files referenced by the test programs:

```
$define BSLN user$disk:[robin.hughes.wsbsln.baseline]
```

When the four files were compiled interactively, the two error-free versions generated correct listings. The first file (MTINHST_C.) generated an incomplete and incorrect listing file. The listing showed the include files inserted first, followed by "C", "8086" and a few other lines of the program. The output displayed on the screen looked like:

```
In pass1.
  70 else
      ^25
  136
      ^408
```

```
In C Nocode.
comp: C Nocode cannot recover from errors.
```

When the third file (MTOPNDF_C.) was compiled, the listing appeared fine except for the insertion a some strange control charaters.

These last two files were compiled in batch mode (file: user\$disk:[robin.hughes.rgalo.test]hughes.com).

The first file (MTINHST_C.) generated a complete but incorrect listing. Although two errors were found (25 & 408) the line at the bottom stated that errors = 0. The include file expansion preceeded the "C" and "8086" in the listing, and lines like, #include filename, were still in the file. The error message was at line 72 of the listing instead of line 2472 were the '}' was actual missing. Finally the last 100 lines had useless numbers in the left margin.

When the third file (MTOPNDF_C.) was compiled, an incomplete listing was generated with the include file expansions listed first.

All of these tests were done on the VAX750 with the /e/v/o options.

This problem also occurs on the 68000.

Temporary solution:

No temporary solution available

Signed off 08/25/86 in release 301.50

Number: D200059048 Product: 6809 C VAX 64822S003 01.20

One-line description:

Host compilers do not put absolute pats specifications in relocatables

Problem:

Host compilers do not specify the full path name in the relocatable file.

Signed off 08/25/86 in release 301.50

Number: D200049007 Product: 6809 C VAX 64822S003 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 301.50

Number: 5000096594 Product: 6809 PASCAL 64813 01.08

Keywords: ENHANCEMENT

One-line description:

Superfluous code generated for bounds checking in FOR loop with consts.

Problem:

```
CONST C1, C2 = anyvalue;
VAR V1 : anytype;
```

BEGIN

```
FOR V1 := C1 TO C2 DO; This generates boundary checking code prior to
executing the loop even though they are known
at compile time.
```

```
FOR V1 := 10 TO 20 DO; This does the same thing;
```

Temporary solution:

None at this time.

Signed off 08/25/86 in release 301.10

Number: 5000114777 Product: 6809 PASCAL 64813 01.08

Keywords: CODE GENERATOR

One-line description:

SHIFT funct. used as an array reference creates incorrect code.

Problem:

Incorrect code is generated when a reference to an array member uses a SHIFT operation for the index:

```
TYPE
  SET8 = SET OF BIT8;
  TAB8 = ARRAY [0..3] OF SET8;
```

```
VAR
  T : TAB8;
  S : SET8;
```

BEGIN

```
T[1] := S;
T[SHIFT(11,-3)] := S;      {generates incorrect code}
END.
```

Temporary work around:

Store SHIFT result in a temporary variable, then use variable as array index.

Note: Code generated on the 9000/vax is different from that generated on the HP64000, but both are incorrect.

Signed off 08/25/86 in release 301.10

Number: 5000119925 Product: 6809 PASCAL 64813 01.08

Keywords: CODE GENERATOR

One-line description:

An automat. BYTE to INT. conversion within a WITH statmnt. - gen. bad cd

Problem:

When the \$RANGE ON \$ compiler option is used, an automatic BYTE to INTEGER conversion being performed on a record field within a WITH statement generates 1006 (Call HP) error message on the 64100. On the 9000 and VAX the following message is created: "comp failed: too many errors in pass2". If the element referenced is the first record field, or if a functional type change is made (even if same as declared), the correct code is generated.

The following program demonstrates this problem:

"6809"

PROGRAM TEST;

\$EXTENSIONS ON, RANGE ON\$

```
VAR I : -1000..1000;
    REC : RECORD
      PLACE : BYTE;
      B : BYTE;
    END;
```

BEGIN

```
WITH REC DO I := B;      {generates error -1006}
WITH REC DO I := BYTE (B); {work around}
END.
```

The problem occurs when the variable I (range -1000..1000) and the variable B (range -128..127) have different ranges. If I is changed to have a range within -128..127 no error occurs, or if B is changed to have a range greater than or equal to -1000..1000 (i.e. signed_16, integer) no error occurs.

Temporary Workaround:

- 1) Make the element referenced in this manner the first element in the record declaration, or do a functional type change around the record field (see above example).
- 2) Turn \$RANGE OFF\$.

Signed off 08/25/86 in release 301.10

Number: D200036772 Product: 6809 PASCAL 64813 01.08

Keywords: INCLUDE

One-line description:

Nested INCLUDE files 3 or more deep cause 64000 to "hang" in pass 3.

Problem:
Nested INCLUDE files 3 or more deep cause 64000 to hang in pass 3.

Temporary solution:
None at this time.

Signed off 08/25/86 in release 301.10

Number: D200045237 Product: 6809 PASCAL 64813 01.08

One-line description:
DIFFERENT BUT EQUAL OBJECT CODE GENERATED ON 64000 THAN IN THE UNIX ENV.

Problem:
THE 6809 COMPILER MAY GENERATE DIFFERENT BUT EQUAL CODE IN THE 64000 ENVIRONMENT THAN THE HP-UX OR VMS ENVIRONMENTS.

THIS CODE IS ACTUALLY EQUAL IN IT'S RESULTS BUT WILL SHOW DIFFERENCES IF COMPAIRED.

EXAMPLE: THIS COULD RESULT FROM MATH OPERATIONS TAKING PLACE IN A DIFFERENT ORDER - THE RESULT WILL BE THE SAME BUT THE CODE DIFFERENT.

Signed off 08/25/86 in release 301.10

Number: D200047365 Product: 6809 PASCAL 64813 01.08

One-line description:
TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 301.10

Number: D200052480 Product: 6809 PASCAL 64813 01.09

One-line description:
Missing semicolon causes compiler to hang in Pass 1.

Problem:
The following code causes the 64000 to hang in pass 1. An error is generated on the hosts stating that parsing has stopped at a particular line number.

```
"processor name"
PROGRAM MAIN;
TYPE
STRUCTURED= RECORD
    INT1:INTEGER;
    INT2:INTEGER;
END;
```

```
PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
I:=P1      <--This missing semicolon causes the problem
I:=P1.2;
I:=P2;
```

- 6809 PASCAL -

END;

BEGIN
END.

Temporary solution:
If the compiler hangs, look for a statement without a semicolon. On the 64000, the status line will show which line of code it stopped on. On the hosts, the error message generated indicates which line of code parsing stopped on.

Signed off 08/25/86 in release 301.10

- 6809 PASCAL -

Number: D200048660 Product: 6809 PASCAL 300 64813S004 01.00

Keywords: CODE GENERATOR

One-line description:
SHIFT funct. used as an array reference creates incorrect code.

Problem:
Incorrect code is generated when a reference to an array member uses a SHIFT operation for the index:

```

TYPE
  SET8 = SET OF BIT8;
  TAB8 = ARRAY [0..3] OF SET8;

VAR
  T : TAB8;
  S : SET8;

BEGIN
  T[1] := S;
  T[SHIFT(11,-3)] := S;      {generates incorrect code}
END.
```

Temporary work around:

Store SHIFT result in a temporary variable, then use variable as array index.

Note: Code generated on the 9000/vax is different from that generated on the HP64000, but both are incorrect.

Signed off 08/25/86 in release 401.10

Number: D200052514 Product: 6809 PASCAL 300 64813S004 01.00

One-line description:
Missing semicolon causes compiler to hang in Pass 1.

Problem:
The following code causes the 64000 to hang in pass 1. An error is generated on the hosts stating that parsing has stopped at a particular line number.

```

"6809"
PROGRAM MAIN;
TYPE
STRUCTURED= RECORD
  INT1:INTEGER;
  INT2:INTEGER;
END;

PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
  I:=P1      <--This missing semicolon causes the problem
  I:=P1.2;
```

- 6809 PASCAL -

I:=P2;
END;

BEGIN
END.

Temporary solution:
If the compiler hangs, look for a statement without a semicolon. On the 64000, the status line will show which line of code it stopped on. On the hosts, the error message generated indicates which line of code parsing stopped on.

Signed off 08/25/86 in release 401.10

Number: D200058735 Product: 6809 PASCAL 300 64813S004 01.00

Keywords: PREPROCESSOR

One-line description:
Preprocessor reports errors when symbols hp64000, vms or hpux w #if

Signed off 08/25/86 in release 401.10

Number: D200059162 Product: 6809 PASCAL 300 64813S004 01.00

One-line description:
Host compilers do not put absolute paths specifications in relocatables

Problem:
Host compilers do not specify the full path name in the relocatable file.

Signed off 08/25/86 in release 401.10

Number: D200048777 Product: 6809 PASCAL 300 64813S004 00.00

One-line description:
Linker output file should use alternate file extension.

Signed off 08/25/86 in release 401.10

- 6809 PASCAL -

Number: D200034181 Product: 6809 PASCAL 500 64813S001 01.00

Keywords: ENHANCEMENT

One-line description:
Superfluous code generated for bounds checking in FOR loop with consts.

Problem:
CONST C1, C2 = anyvalue;
VAR V1 : anytype;

```
BEGIN
FOR V1 := C1 TO C2 DO; This generates boundary checking code prior to
                        executing the loop even though they are known
                        at compile time.
```

```
FOR V1 := 10 TO 20 DO; This does the same thing;
```

Temporary solution:
None at this time.

Signed off 08/25/86 in release 101.20

Number: D200036988 Product: 6809 PASCAL 500 64813S001 01.00

Keywords: PASS 3

One-line description:
Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:
Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect data being output to the list file. In selected cases, machine code will be incorrectly listed. For example, consider the following Pascal program.

```
$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;
```

```
VAR
  a, b : BOOLEAN;
```

```
PROCEDURE one;
```

```
  BEGIN
    a := b;
  END;
```

In the example listed above, the output file will denote machine code of the form FFFFC00001 for one of the generated assembly statements. The correct value should be C8000001. This problem is caused by an incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 101.20

Number: D200047373 Product: 6809 PASCAL 500 64813S001 01.00

One-line description:
TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 101.20

Number: D200048645 Product: 6809 PASCAL 500 64813S001 01.10

Keywords: CODE GENERATOR

One-line description:
SHIFT funct. used as an array reference creates incorrect code.

Problem:
Incorrect code is generated when a reference to an array member uses a SHIFT operation for the index:

```
TYPE
  SET8 = SET OF BIT8;
  TAB8 = ARRAY [0..3] OF SET8;
```

```
VAR
  T : TAB8;
  S : SET8;
```

```
BEGIN
  T[1] := S;
  T[SHIFT(11,-3)] := S;      {generates incorrect code}
END.
```

Temporary work around:

Store SHIFT result in a temporary variable, then use variable as array index.

Note: Code generated on the 9000/vax is different from that generated on the HP64000, but both are incorrect.

Signed off 08/25/86 in release 101.20

Number: D200052498 Product: 6809 PASCAL 500 64813S001 01.10

One-line description:
Missing semicolon causes compiler to hang in Pass 1.

Problem:
The following code causes the 64000 to hang in pass 1. An error is generated on the hosts stating that parsing has stopped at a particular line number.

```
"processor name"
PROGRAM MAIN;
TYPE
```

```
STRUCTURED= RECORD
  INT1:INTEGER;
  INT2:INTEGER;
END;
```

```
PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
  I:=P1      <--This missing semicolon causes the problem
  I:=P1.2;
  I:=P2;
END;

BEGIN
END.
```

Temporary solution:
If the compiler hangs, look for a statement without a semicolon.
On the 64000, the status line will show which line of code it
stopped on. On the hosts, the error message generated indicates
which line of code parsing stopped on.

Signed off 08/25/86 in release 101.20

Number: D200058719 Product: 6809 PASCAL 500 64813S001 01.10

Keywords: PREPROCESSOR

One-line description:
Preprocessor reports errors when symbols hp64000, vms or hpux w #if

Signed off 08/25/86 in release 101.20

Number: D200059147 Product: 6809 PASCAL 500 64813S001 01.10

One-line description:
Host compilers do not put absolute pats specifications in relocatables

Problem:
Host compilers do not specify the full path name in the
relocatable file.

Signed off 08/25/86 in release 101.20

Number: D200048751 Product: 6809 PASCAL 500 64813S001 00.00

One-line description:
Linker output file should use alternate file extension.

Signed off 08/25/86 in release 101.20

Number: D200034199 Product: 6809 PASCAL VAX 64813S003 01.00

Keywords: ENHANCEMENT

One-line description:
Superfluous code generated for bounds checking in FOR loop with consts.

Problem:
CONST C1, C2 = anyvalue;
VAR V1 : anytype;

```
BEGIN
FOR V1 := C1 TO C2 DO; This generates boundary checking code prior to
                        executing the loop even though they are known
                        at compile time.
```

```
FOR V1 := 10 TO 20 DO; This does the same thing;
```

Temporary solution:
None at this time.

Signed off 08/25/86 in release 301.30

Number: D200036996 Product: 6809 PASCAL VAX 64813S003 01.00

Keywords: PASS 3

One-line description:
Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:
Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect
data being output to the list file. In selected cases, machine code
will be incorrectly listed. For example, consider the following
Pascal program.

```
$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;

  VAR
    a, b : BOOLEAN;

  PROCEDURE one;

    BEGIN
      a := b;
    END;
```

In the example listed above, the output file will denote machine code
of the form FFFFC00001 for one of the generated assembly statements.
The correct value should be C8000001. This problem is caused by an
incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 301.30

Number: D200043372 Product: 6809 PASCAL VAX 64813S003 01.00

One-line description:
COMPILER ASSIGNS INCORRECT TEMP STORAGE SOMETIMES BYTE TO REAL.

Signed off 08/25/86 in release 301.30

Number: D200047381 Product: 6809 PASCAL VAX 64813S003 01.00

One-line description:
TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 301.30

Number: D200048652 Product: 6809 PASCAL VAX 64813S003 01.10

Keywords: CODE GENERATOR

One-line description:
SHIFT funct. used as an array reference creates incorrect code.

Problem:
Incorrect code is generated when a reference to an array member uses
a SHIFT operation for the index:

```

TYPE
  SET8 = SET OF BIT8;
  TAB8 = ARRAY [0..3] OF SET8;

VAR
  T : TAB8;
  S : SET8;

BEGIN
  T[1] := S;
  T[SHIFT(11,-3)] := S;      {generates incorrect code}
END.

```

Temporary work around:

Store SHIFT result in a temporary variable, then use variable as
array index.

Note: Code genrated on the 9000/vax is different from that generated
on the HP64000, but both are incorrect.

Signed off 08/25/86 in release 301.30

Number: D200052506 Product: 6809 PASCAL VAX 64813S003 01.10

One-line description:
Missing semicolon causes compiler to hang in Pass 1.

Problem:

- 6809 PASCAL -

The following code causes the 64000 to hang in pass 1. An error
is generated on the hosts stating that parsing has stopped at
a particular line number.

```

"processor name"
PROGRAM MAIN;
TYPE
STRUCTURED= RECORD
  INT1:INTEGER;
  INT2:INTEGER;
END;

```

```

PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
I:=P1      <--This missing semicolon causes the problem
I:=P1.2;
I:=P2;
END;

```

```

BEGIN
END.

```

Temporary solution:

If the compiler hangs, look for a statement without a semicolon.
On the 64000, the status line will show which line of code it
stopped on. On the hosts, the error message generated indicates
which line of code parsing stopped on.

Signed off 08/25/86 in release 301.30

Number: D200058727 Product: 6809 PASCAL VAX 64813S003 01.10

Keywords: PREPROCESSOR

One-line description:
Preprocessor reports errors when symbols hp64000, vms or hpux w #if

Signed off 08/25/86 in release 301.30

Number: D200059154 Product: 6809 PASCAL VAX 64813S003 01.10

One-line description:
Host compilers do not put absolute pats specifications in relocatables

Problem:
Host compilers do not specify the full path name in the
relocatable file.

Signed off 08/25/86 in release 301.30

Number: D200048769 Product: 6809 PASCAL VAX 64813S003 00.00

One-line description:
Linker output file should use alternate file extension.

Signed off 08/25/86 in release 301.30

- 6809 PASCAL -

Number: 2700005900 Product: 8085 B PASCAL 64825 00.00

One-line description:
Incorrect code generated for WHILE construct.

Temporary solution:
There are two possible work-arounds for this problem:

- (1) alter the order of comparisons, or
- (2) change the TYPE of a to something other than SIGNED_16.

Signed off 08/25/86 in release 501.03

Number: D200019307 Product: 8085 B PASCAL 64825 01.01

Keywords: PASS 2

One-line description:
Program re-BOOTS 64000 station.

Problem:
Program will re-BOOT the 64000 station when compiled using the 64000 cross compiler. NOTE: This problem exists ONLY with the 64000 compiler.

Signed off 08/25/86 in release 501.03

Number: D200020131 Product: 8085 B PASCAL 64825 01.01

Keywords: STRING ARRAYS

One-line description:
Multidimensional arrays of packed string arrays cannot be assigned to.

Problem:

```
PROGRAM TEST;
TYPE STRING_40 = PACKED ARRAY [0..15] OF CHAR;
VAR ARRAY1 : ARRAY[1..2,1..2] OF STRING_40;
```

```
BEGIN
ARRAY1[1,1] := 'HELLO'
****Pass 2 error ?? 1006 => Contact HP
END.
```

Signed off 08/25/86 in release 501.03

Number: D200022434 Product: 8085 B PASCAL 64825 01.01

Keywords: CODE GENERATOR

One-line description:
Incorrect code generated for IF statement.

Problem:
Compiling the following program demonstrates a code generation problem for the IF statement.

```
PROGRAM test;
$EXTENSIONS$
```

```
VAR
SCAN_TYPE : BYTE;
```

```
BEGIN
IF (SCAN_TYPE > 6) OR (SCAN_TYPE = 2) THEN
END.
```

After determining the result of (SCAN_TYPE > 6) the compiler overwrites the result (stored in the accumulator) with other data. Thus, the only comparison made is (SCAN_TYPE = 2).

Temporary solution:

Divide the IF statement into two separate statements.

Signed off 08/25/86 in release 501.03

Number: D200022491 Product: 8085 B PASCAL 64825 01.01

Keywords: CODE GENERATOR

One-line description:

Incorrect code generated for SET inclusion statement.

Problem:

The following program demonstrates a code generation problem for the SET inclusion statement.

```
PROGRAM test;
$EXTENSIONS$
```

```
TYPE
BYTE_SET = SET OF (B0, B1, B2, B3, B4, B5, B6, B7);
```

```
VAR
status_byte : BYTE_SET;
```

```
BEGIN
IF [B0] <= status_byte THEN
END.
```

In the example listed, the compiler generates code which OR's and CP's (compare) rather than an AND operation.

Temporary solution:

Use the set inclusion statement: IF B0 IN status_byte THEN ...

Signed off 08/25/86 in release 501.03

Number: D200026500 Product: 8085 B PASCAL 64825 01.01

One-line description:

Defining TRUE and FALSE as global may result in duplicate symbol names.

Problem:

- 8085 B PASCAL -

Defining the variables (constants) TRUE and FALSE to be global may result in a duplicate symbol error during a link. These variables are incorrectly defined as global in the Zwordcmp routine located in 'Zlibrary'.

NOTE: Redefining the values of TRUE and/or FALSE is not a legal Pascal operation. Redefinition of these constants is therefore not supported when using the HP 64000 compiler.

Temporary solution:

Obtain the source to Zwordcmp from your local HP Systems Engineer.

Signed off 08/25/86 in release 501.03

Number: D200034157 Product: 8085 B PASCAL 64825 01.01

Keywords: STRING

One-line description:

Pointers to STRINGS cannot be assigned a string of length one.

Problem:

```
TYPE STR_ARR : PACKED ARRAY [0..7] OF CHAR; {I.E., A STRING}
ARR_PTR : ^STR_ARR;
```

```
VAR PTR : ARR_PTR;
```

```
BEGIN
```

```

:
:
PTR^ := "1234567"; {WORKS FINE}
PTR^ := "1"; {GENERATES THE FOLLOWING INCORRECT CODE}
LD A,001H {THIS WILL BE THE STRING LENGTH}
LD HL,[PTR]
LD [HL], A {SO FAR SO GOOD, WE'VE LOADED THE BYTE COUNT IN
STR_ARR[0]}
LD HL,[PTR+001H]{THIS IS THE MISTAKE. WE SHOULD HAVE DONE A
LD HL,[PTR] INC HL}
LD [HL], 031H
```

Temporary solution:

None at this time.

Signed off 08/25/86 in release 501.03

Number: D200036814 Product: 8085 B PASCAL 64825 01.01

Keywords: INCLUDE

One-line description:

Nested INCLUDE files 3 or more deep cause 64000 to "hang" in pass 3.

Problem:

Nested INCLUDE files 3 or more deep cause 64000 to hang in pass 3.

- 8085 B PASCAL -

Temporary solution:
None at this time.

Signed off 08/25/86 in release 501.03

Number: D200037796 Product: 8085 B PASCAL 64825 01.01

One-line description:
Bad code generated for assignment statement.

Problem:
Bad code is generated for the following two Pascal statements.

```

$SEPARATE ON$
$EXTENSIONS ON$
PROGRAM test;

```

```

PROCEDURE one (a : BYTE; VAR b : SIGNED_16);

```

```

  VAR
    c : SIGNED_16;
  BEGIN
    c := SIGNED_16 (a) + b;
    c := SIGNED_16 (a) - b;
  END.

```

In the first statement an 'XCHG' assembly instruction is missing. In the second statement 4 extra lines are generated and the code generated is incorrect.

Temporary solution:
Reverse the order of the two "operands" in the addition statement. In other words use the expression

```

c := b + SIGNED_16 (a);

```

Signed off 08/25/86 in release 501.03

Number: D200040261 Product: 8085 B PASCAL 64825 01.01

Keywords: SETS

One-line description:
SUPERSET or SUBSET checking doesn't work.

Problem:
TYPE SET TYPE = SET OF (B0,B1,B2,B3,B4,B5,B6,B7);
VAR X : SET_TYPE;
BEGIN
IF X <= [B3,B4] THEN; {GENERATES INCORRECT CODE}
IF X >= [B3,B4] THEN; {GENERATES INCORRECT CODE}

Temporary solution:
None at this time.

Signed off 08/25/86 in release 501.03

Number: D200041145 Product: 8085 B PASCAL 64825 01.01

One-line description:
Bad code generated for IF.. statement (including WITH).

Signed off 08/25/86 in release 501.03

Number: D200044735 Product: 8085 B PASCAL 64825 01.01

Keywords: FOR LOOP

One-line description:
FOR Signed8 := 0 TO 2 DO REAL1 := REAL1/REAL2 overwrites the A-register.

Temporary solution:
Use the compiler option \$AMNESIA +\$

Signed off 08/25/86 in release 501.03

Number: D200047696 Product: 8085 B PASCAL 64825 01.01

One-line description:
TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 501.03

Number: D200052381 Product: 8085 B PASCAL 64825 01.02

One-line description:
Incorrect code generated when a CHAR is converted to an UNSIGNED_16.

Problem:
Incorrect code is generated when a CHAR variable is converted to an UNSIGNED_16. The following code is an example:

```

"processor name"
$EXTENSIONS ON$
$RECURSIVE OFF$
PROGRAM PASCALTEST;
TYPE
  BUG_TYPE = UNSIGNED_16; (*There is no problem if this is
                           SIGNED_16*)

```

```

PROCEDURE BUGGY(COUNT:BUG_TYPE);EXTERNAL;
FUNCTION OPEN:SIGNED_16;
VAR
  COUNT : BUG_TYPE;
  LEN: CHAR;
BEGIN
  (*THE NEXT TWO STATEMENTS GENERATE INCORRECT CODE*)
  COUNT := BUG_TYPE(LEN);
  (* LD A,001H *)
  (* LD [Dopen+00002H],A *)
  (* LD A,[Dopen+00004H] *)
  (* LD [Dopen+00003H],A *)

```

```
BUGGY(BUG_TYPE(LEN));
```

```
(* LD A,001H *)
(* LD [Dopen+00005H],BC*)
(* LD A,[Dopen+00004H] *)
(* LD HL,[Dopen+00005H]*)
(* PUSH HL *)
(* CALL BUGGY *)
(* INC SP *)
(* INC SP *)
```

```
END;
```

Something very strange occurs when the same code is compiled with \$RECURSIVE ON\$. The statement BUGGY(BUG_TYPE(LEN)); generates the following code:

```
LD A,001H
LD [IX-11],A
LD [IX-10],WHAT??
LD A,[IX-5]
LD L,A
LD H,[IX-10]
PUSH HL
CALL BUGGY
INC SP
INC SP
```

Temporary solution:

No known temporary solution.

Signed off 08/25/86 in release 501.03

Number: D200052670 Product: 8085 B PASCAL 64825 01.02

One-line description:

Missing semicolon causes compiler to hang in Pass 1.

Problem:

The following code causes the 64000 to hang in pass 1. An error is generated on the hosts stating that parsing has stopped at a particular line number.

```
"BZ80"
PROGRAM MAIN;
TYPE
STRUCTURED= RECORD
    INT1:INTEGER;
    INT2:INTEGER;
END;
```

```
PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
I:=P1    <--This missing semicolon causes the problem
I:=P1.2;
I:=P2;
```

- 8085 B PASCAL -

```
END;
```

```
BEGIN
END.
```

Temporary solution:

If the compiler hangs, look for a statement without a semicolon. On the 64000, the status line will show which line of code it stopped on. On the hosts, the error message generated indicates which line of code parsing stopped on.

Signed off 08/25/86 in release 501.03

- 8085 B PASCAL -

Number: D200052084 Product: 8085 B PASCAL 300 64825S004 01.00

One-line description:

Bad code generated for IF.. statement (including WITH).

Problem:

The following program demonstrates a code generation problem. The compiler loads the accumulator with the constant value, then overwrites the value when an indirect load (LDAX) is performed.

```
PROGRAM test;
$EXTENSIONS ON$
$RECURSIVE ON$
```

TYPE

```
codeblk = RECORD
  id: BYTE;
  base: SIGNED_16;
END;
pointer = ^codeblk;
```

```
PROCEDURE one (fac_ptr: pointer);
```

```
BEGIN
  WITH fac_ptr^ DO
    IF (id >= 25) AND (id <= 29) THEN
END;
```

In addition, if the WITH statement is commented out, the compiler also generates incorrect code. In this case, the compiler loads the value of "id" and "25" and then calls a run-time library routine which compares the two values. After returning from the comparison routine, the compiler destroys the value in the HL register pair (id), and then later assumes the value in HL is still valid.

Temporary solution:

No known temporary solution.

Signed off 08/25/86 in release 401.10

Number: D200052415 Product: 8085 B PASCAL 300 64825S004 01.00

One-line description:

Incorrect code generated when a CHAR is converted to an UNSIGNED_16.

Problem:

Incorrect code is generated when a CHAR variable is converted to an UNSIGNED_16. The following code is an example:

```
"processor name"
$EXTENSIONS ON$
$RECURSIVE OFF$
PROGRAM PASCALTEST;
TYPE
```

```
  BUG_TYPE = UNSIGNED_16; (*There is no problem if this is
                           SIGNED_16*)
```

```
PROCEDURE BUGGY(COUNT:BUG_TYPE);EXTERNAL;
FUNCTION OPEN:SIGNED_16;
VAR
```

```
  COUNT : BUG_TYPE;
  LEN: CHAR;
BEGIN
  (*THE NEXT TWO STATEMENTS GENERATE INCORRECT CODE*)
  COUNT := BUG_TYPE(LEN);
  (* LD  A,001H          *)
  (* LD  [Dopen+00002H],A *)
  (* LD  A,[Dopen+00004H] *)
  (* LD  [Dopen+00003H],A *)
  BUGGY(BUG_TYPE(LEN));
  (* LD  A,001H          *)
  (* LD  [Dopen+00005H],BC *)
  (* LD  A,[Dopen+00004H] *)
  (* LD  HL,[Dopen+00005H] *)
  (* PUSH HL             *)
  (* CALL BUGGY          *)
  (* INC  SP             *)
  (* INC  SP             *)
END;
```

Something very strange occurs when the same code is compiled with \$RECURSIVE ON\$. The statement BUGGY(BUG_TYPE(LEN)); generates the following code:

```
LD  A,001H
LD  [IX-11],A
LD  [IX-10],WHAT???
LD  A,[IX-5]
LD  L,A
LD  H,[IX-10]
PUSH HL
CALL BUGGY
INC  SP
INC  SP
```

Temporary solution:

No known temporary solution.

Signed off 08/25/86 in release 401.10

Number: D200052704 Product: 8085 B PASCAL 300 64825S004 01.00

One-line description:

Missing semicolon causes compiler to hang in Pass 1.

Problem:

The following code causes the 64000 to hang in pass 1. An error is generated on the hosts stating that parsing has stopped at a particular line number.

```
"BZ80"
PROGRAM MAIN;
```

```

TYPE
STRUCTURED= RECORD
  INT1:INTEGER;
  INT2:INTEGER;
END;

```

```

PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
  I:=P1      <--This missing semicolon causes the problem
  I:=P1.2;
  I:=P2;
END;

```

```

BEGIN
END.

```

Temporary solution:

If the compiler hangs, look for a statement without a semicolon. On the 64000, the status line will show which line of code it stopped on. On the hosts, the error message generated indicates which line of code parsing stopped on.

Signed off 08/25/86 in release 401.10

```

Number: D200058883  Product: 8085 B PASCAL  300 64825S004  01.00

```

Keywords: PREPROCESSOR

One-line description:

Preprocessor reports errors when symbols hp64000, vms or hpux w #if

Signed off 08/25/86 in release 401.10

```

Number: D200059287  Product: 8085 B PASCAL  300 64825S004  01.00

```

One-line description:

Host compilers do not put absolute pats specifications in relocatables

Problem:

Host compilers do not specify the full path name in the relocatable file.

Temporary solution:

No known temporary solution.

Signed off 08/25/86 in release 401.10

```

Number: D200049106  Product: 8085 B PASCAL  300 64825S004  00.00

```

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 401.10

- 8085 B PASCAL -

```

Number: 5000107888  Product: 8085 B PASCAL  500 64825S001  01.10

```

Keywords: PASS 2

One-line description:

Array element as argument of CASE statement causes compile to fail.

Problem:

The following program causes the error "comp failed; too many errors in pass 2" to be generated:

```

"processor name"
$EXTENSIONS ON$
PROGRAM TEST;
VAR
  I: INTEGER;
  T: ARRAY[1..3] OF BYTE;

```

```

BEGIN
  CASE T[I] OF;
  END;
END.

```

Signed off 08/25/86 in release 101.40

```

Number: D200020149  Product: 8085 B PASCAL  500 64825S001  01.10

```

Keywords: STRING ARRAYS

One-line description:

Multidimensional arrays of packed string arrays cannot be assigned to.

Problem:

```

PROGRAM TEST;
TYPE STRING_40 = PACKED ARRAY [0..15] OF CHAR;
VAR ARRAY1 : ARRAY[1..2,1..2] OF STRING_40;

```

```

BEGIN
  ARRAY1[1,1] := 'HELLO'
  ****Pass 2 error ?? 1006 => Contact HP
END.

```

Temporary solution:

Put the assignment statement within a procedure and call the procedure when necessary. The array may be accessed by either global or local variables.

Signed off 08/25/86 in release 101.40

```

Number: D200022442  Product: 8085 B PASCAL  500 64825S001  01.10

```

Keywords: CODE GENERATOR

One-line description:

Incorrect code generated for IF statement.

- 8085 B PASCAL -

Problem:

Compiling the following program demonstrates a code generation problem for the IF statement.

```
PROGRAM test;
$EXTENSIONS$
```

```
VAR
  SCAN_TYPE : BYTE;
```

```
BEGIN
  IF (SCAN_TYPE > 6) OR (SCAN_TYPE = 2) THEN
  END.
```

After determining the result of (SCAN_TYPE > 6) the compiler overwrites the result (stored in the accumulator) with other data. Thus, the only comparison made is (SCAN_TYPE = 2).

Temporary solution:

Divide the IF statement into two separate statements.

Signed off 08/25/86 in release 101.40

Number: D200022509 Product: 8085 B PASCAL 500 64825S001 01.10

Keywords: CODE GENERATOR

One-line description:

Incorrect code generated for SET inclusion statement.

Problem:

The following program demonstrates a code generation problem for the SET inclusion statement.

```
PROGRAM test;
$EXTENSIONS$
```

```
TYPE
  BYTE_SET = SET OF (B0, B1, B2, B3, B4, B5, B6, B7);
```

```
VAR
  status_byte : BYTE_SET;
```

```
BEGIN
  IF [B0] <= status_byte THEN
  END.
```

In the example listed, the compiler generates code which OR's and CP's (compare) rather than an AND operation.

Temporary solution:

Use the set inclusion statement: IF B0 IN status_byte THEN ...

Signed off 08/25/86 in release 101.40

Number: D200026518 Product: 8085 B PASCAL 500 64825S001 01.10

One-line description:

Defining TRUE and FALSE as global may result in duplicate symbol names.

Problem:

Defining the variables (constants) TRUE and FALSE to be global may result in a duplicate symbol error during a link. These variables are incorrectly defined as global in the Zwordcmp routine located in 'Zlibrary'.

NOTE: Redefining the values of TRUE and/or FALSE is not a legal Pascal operation. Redefinition of these constants is therefore not supported when using the HP 64000 compiler.

Temporary solution:

Obtain the source to Zwordcmp from your local HP Systems Engineer.

Signed off 08/25/86 in release 101.40

Number: D200027789 Product: 8085 B PASCAL 500 64825S001 01.10

One-line description:

No form feed between the expanded listing and the cross reference table.

Problem:

During compilation, with XREF option on, the compiler does not provide a form feed (FF) in the listing file. The XREF starts on the same page as the end of the listing. Also, the page number says 535 when it should be page 2.

Temporary solution:

After compiling with the xref option, edit the expanded listing file and insert a "control L" before the beginning of the cross reference listing.

Signed off 08/25/86 in release 101.40

Number: D200028852 Product: 8085 B PASCAL 500 64825S001 01.10

One-line description:

Incorrect code generated for WHILE construct.

Temporary solution:

There are two possible work-arounds for this problem:

- (1) alter the order of comparisons, or
- (2) change the TYPE of a to something other than SIGNED_16.

Signed off 08/25/86 in release 101.40

Number: D200034165 Product: 8085 B PASCAL 500 64825S001 01.10

Keywords: STRING

One-line description:

Pointers to STRINGS cannot be assigned a string of length one.

Problem:

```
TYPE STR_ARR : PACKED ARRAY [0..7] OF CHAR; {I.E., A STRING}
   ARR_PTR : ^STR_ARR;
```

```
VAR PTR : ARR_PTR;
```

```
BEGIN
```

```
  .
  .
PTR^ := "1234567"; {WORKS FINE}
PTR^ := "1";      {GENERATES THE FOLLOWING INCORRECT CODE}
LD A,001H        {THIS WILL BE THE STRING LENGTH}
LD HL,[PTR]
LD [HL], A       {SO FAR SO GOOD, WE'VE LOADED THE BYTE COUNT IN
                  STR_ARR[0]}
LD HL,[PTR+001H]{THIS IS THE MISTAKE. WE SHOULD HAVE DONE A
                  LD HL,[PTR]   INC HL}
LD [HL], 031H
```

Temporary solution:
None at this time.

Signed off 08/25/86 in release 101.40

Number: D200037192 Product: 8085 B PASCAL 500 64825S001 01.20

Keywords: PASS 3

One-line description:

Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:

Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect data being output to the list file. In selected cases, machine code will be incorrectly listed. For example, consider the following Pascal program.

```
$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;

VAR
  a, b : BOOLEAN;

PROCEDURE one;

BEGIN
  a := b;
END;
```

- 8085 B PASCAL -

In the example listed above, the output file will denote machine code of the form FFFFC00001 for one of the generated assembly statements. The correct value should be C8000001. This problem is caused by an incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 101.40

Number: D200037804 Product: 8085 B PASCAL 500 64825S001 01.20

One-line description:

Bad code generated for assignment statement.

Problem:

Bad code is generated for the following two Pascal statements.

```
$SEPARATE ON$
$EXTENSIONS ON$
PROGRAM test;
```

```
PROCEDURE one (a : BYTE; VAR b : SIGNED_16);
```

```
VAR
```

```
  c : SIGNED_16;
```

```
BEGIN
```

```
  c := SIGNED_16 (a) + b;
```

```
  c := SIGNED_16 (a) - b;
```

```
END.
```

In the first statement an 'XCHG' assembly instruction is missing. In the second statement 4 extra lines are generated and the code generated is incorrect.

Temporary solution:

Reverse the order of the two "operands" in the addition statement. In other words use the expression

```
  c := b + SIGNED_16 (a);
```

Signed off 08/25/86 in release 101.40

Number: D200040279 Product: 8085 B PASCAL 500 64825S001 01.20

Keywords: SETS

One-line description:

SUPERSET or SUBSET checking doesn't work.

Problem:

```
TYPE SET_TYPE = SET OF (B0,B1,B2,B3,B4,B5,B6,B7);
VAR X : SET_TYPE;
BEGIN
```

- 8085 B PASCAL -

```
IF X <= [B3,B4] THEN; {GENERATES INCORRECT CODE}
IF X >= [B3,B4] THEN; {GENERATES INCORRECT CODE}
```

Temporary solution:
None at this time.

Signed off 08/25/86 in release 101.40

```
Number: D200041749 Product: 8085 B PASCAL 500 64825S001 01.20
```

One-line description:
Bad code generated for IF.. statement (including WITH).

Signed off 08/25/86 in release 101.40

```
Number: D200044743 Product: 8085 B PASCAL 500 64825S001 01.20
```

Keywords: FOR LOOP

One-line description:
FOR Signed8 := 0 TO 2 DO REAL1 := REAL1/REAL2 overwrites the A-register.

Temporary solution:
Use the compiler option \$AMNESIA +\$

Signed off 08/25/86 in release 101.40

```
Number: D200047704 Product: 8085 B PASCAL 500 64825S001 01.20
```

One-line description:
TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 101.40

```
Number: D200052399 Product: 8085 B PASCAL 500 64825S001 01.30
```

One-line description:
Incorrect code generated when a CHAR is converted to an UNSIGNED_16.

Problem:
Incorrect code is generated when a CHAR variable is converted to an UNSIGNED_16. The following code is an example:

```
"processor name"
$EXTENSIONS ON$
$RECURSIVE OFF$
PROGRAM PASCALTEST;
TYPE
  BUG_TYPE = UNSIGNED_16; (*There is no problem if this is
                          SIGNED_16*)
```

```
PROCEDURE BUGGY(COUNT:BUG_TYPE);EXTERNAL;
FUNCTION OPEN:SIGNED_16;
VAR
  COUNT : BUG_TYPE;
  LEN: CHAR;
BEGIN
```

```
(*THE NEXT TWO STATEMENTS GENERATE INCORRECT CODE*)
COUNT := BUG_TYPE(LEN);
```

```
(* LD A,001H *)
(* LD [Dopen+00002H],A *)
(* LD A,[Dopen+00004H] *)
(* LD [Dopen+00003H],A *)

BUGGY(BUG_TYPE(LEN));

(* LD A,001H *)
(* LD [Dopen+00005H],BC*)
(* LD A,[Dopen+00004H] *)
(* LD HL,[Dopen+00005H]*)
(* PUSH HL *)
(* CALL BUGGY *)
(* INC SP *)
(* INC SP *)
```

END;

Something very strange occurs when the same code is compiled with \$RECURSIVE ON\$. The statement BUGGY(BUG_TYPE(LEN)); generates the following code:

```
LD A,001H
LD [IX-11],A
LD [IX-10],WHAT??
LD A,[IX-5]
LD L,A
LD H,[IX-10]
PUSH HL
CALL BUGGY
INC SP
INC SP
```

Temporary solution:
No known temporary solution.

Signed off 08/25/86 in release 101.40

```
Number: D200052688 Product: 8085 B PASCAL 500 64825S001 01.30
```

One-line description:
Missing semicolon causes compiler to hang in Pass 1.

Problem:
The following code causes the 64000 to hang in pass 1. An error is generated on the hosts stating that parsing has stopped at a particular line number.

```
"BZ80"
PROGRAM MAIN;
TYPE
STRUCTURED= RECORD
  INT1:INTEGER;
  INT2:INTEGER;
END;
```

```
PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
I:=P1      <--This missing semicolon causes the problem
I:=P1.2;
I:=P2;
END;
```

```
BEGIN
END.
```

Temporary solution:

If the compiler hangs, look for a statement without a semicolon. On the 64000, the status line will show which line of code it stopped on. On the hosts, the error message generated indicates which line of code parsing stopped on.

Signed off 08/25/86 in release 101.40

```
Number: D200058867 Product: 8085 B PASCAL 500 64825S001 01.30
```

Keywords: PREPROCESSOR

One-line description:

Preprocessor reports errors when symbols hp64000, vms or hpux w #if

Signed off 08/25/86 in release 101.40

```
Number: D200059261 Product: 8085 B PASCAL 500 64825S001 01.30
```

One-line description:

Host compilers do not put absolute paths specifications in relocatables

Problem:
Host compilers do not specify the full path name in the relocatable file.

Temporary solution:

No known temporary solution.

Signed off 08/25/86 in release 101.40

```
Number: D200049080 Product: 8085 B PASCAL 500 64825S001 00.00
```

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 101.40

```
Number: D200020156 Product: 8085 B PASCAL VAX 64825S003 01.10
```

Keywords: STRING ARRAYS

One-line description:

Multidimensional arrays of packed string arrays cannot be assigned to.

Problem:

```
PROGRAM TEST;
TYPE STRING_40 = PACKED ARRAY [0..15] OF CHAR;
VAR ARRAY1 : ARRAY[1..2,1..2] OF STRING_40;
```

```
BEGIN
ARRAY1[1,1] := 'HELLO'
****Pass 2 error ?? 1006 => Contact HP
END.
```

Temporary solution:

Put the assignment statement within a procedure and call the procedure when necessary. The array may be accessed by either global or local variables.

Signed off 08/25/86 in release 301.60

```
Number: D200022459 Product: 8085 B PASCAL VAX 64825S003 01.10
```

Keywords: CODE GENERATOR

One-line description:

Incorrect code generated for IF statement.

Problem:

Compiling the following program demonstrates a code generation problem for the IF statement.

```
PROGRAM test;
$EXTENSIONS$
```

```
VAR
SCAN_TYPE : BYTE;
```

```
BEGIN
IF (SCAN_TYPE > 6) OR (SCAN_TYPE = 2) THEN
END.
```

After determining the result of (SCAN_TYPE > 6) the compiler overwrites the result (stored in the accumulator) with other data. Thus, the only comparison made is (SCAN_TYPE = 2).

Temporary solution:

Divide the IF statement into two separate statements.

Signed off 08/25/86 in release 301.60

Number: D200022517 Product: 8085 B PASCAL VAX 64825S003 01.10

Keywords: CODE GENERATOR

One-line description:

Incorrect code generated for SET inclusion statement.

Problem:

The following program demonstrates a code generation problem for the SET inclusion statement.

```
PROGRAM test;
$EXTENSIONS$
```

```
TYPE
  BYTE_SET = SET OF (B0, B1, B2, B3, B4, B5, B6, B7);
```

```
VAR
  status_byte : BYTE_SET;
```

```
BEGIN
  IF [B0] <= status_byte THEN
  END.
```

In the example listed, the compiler generates code which OR's and CP's (compare) rather than an AND operation.

Temporary solution:

Use the set inclusion statement: IF B0 IN status_byte THEN ...

Signed off 08/25/86 in release 301.60

Number: D200026526 Product: 8085 B PASCAL VAX 64825S003 01.10

One-line description:

Defining TRUE and FALSE as global may result in duplicate symbol names.

Problem:

Defining the variables (constants) TRUE and FALSE to be global may result in a duplicate symbol error during a link. These variables are incorrectly defined as global in the Zwordcmp routine located in 'Zlibrary'.

NOTE: Redefining the values of TRUE and/or FALSE is not a legal Pascal operation. Redefinition of these constants is therefore not supported when using the HP 64000 compiler.

Temporary solution:

Obtain the source to Zwordcmp from your local HP Systems Engineer.

Signed off 08/25/86 in release 301.60

Number: D200027797 Product: 8085 B PASCAL VAX 64825S003 01.20

One-line description:

No form feed between the expanded listing and the cross reference table.

Problem:

During compilation, with XREF option on, the compiler does not provide a form feed (FF) in the listing file. The XREF starts on the same page as the end of the listing. Also, the page number says 535 when it should be page 2.

Temporary solution:

After compiling with the xref option, edit the expanded listing file and insert a "control L" before the beginning of the cross reference listing.

Signed off 08/25/86 in release 301.60

Number: D200028860 Product: 8085 B PASCAL VAX 64825S003 01.20

One-line description:

Incorrect code generated for WHILE construct.

Temporary solution:

There are two possible work-arounds for this problem:

- (1) alter the order of comparisons, or
- (2) change the TYPE of a to something other than SIGNED_16.

Signed off 08/25/86 in release 301.60

Number: D200034173 Product: 8085 B PASCAL VAX 64825S003 01.20

Keywords: STRING

One-line description:

Pointers to STRINGS cannot be assigned a string of length one.

Problem:

```
TYPE STR_ARR : PACKED ARRAY [0..7] OF CHAR; {I.E., A STRING}
  ARR_PTR : ^STR_ARR;
```

```
VAR PTR : ARR_PTR;
```

```
BEGIN
```

```
  .
  .
  PTR^ := "1234567"; {WORKS FINE}
  PTR^ := "1"; {GENERATES THE FOLLOWING INCORRECT CODE}
  LD A,001H {THIS WILL BE THE STRING LENGTH}
  LD HL,[PTR]
  LD [HL], A {SO FAR SO GOOD, WE'VE LOADED THE BYTE COUNT IN
  STR_ARR[0]}
  LD HL,[PTR+001H]{THIS IS THE MISTAKE. WE SHOULD HAVE DONE A
  LD HL,[PTR] INC HL}
  LD [HL], 031H
```

Temporary solution:
None at this time.

Signed off 08/25/86 in release 301.60

Number: D200037200 Product: 8085 B PASCAL VAX 64825S003 01.20

Keywords: PASS 3

One-line description:
Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:
Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect data being output to the list file. In selected cases, machine code will be incorrectly listed. For example, consider the following Pascal program.

```
$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;

VAR
  a, b : BOOLEAN;

PROCEDURE one;

  BEGIN
    a := b;
  END;
```

In the example listed above, the output file will denote machine code of the form FFFFC00001 for one of the generated assembly statements. The correct value should be C8000001. This problem is caused by an incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 301.60

Number: D200037812 Product: 8085 B PASCAL VAX 64825S003 01.20

One-line description:
Bad code generated for assignment statement.

Problem:
Bad code is generated for the following two Pascal statements.

```
$SEPARATE ON$
$EXTENSIONS ON$
PROGRAM test;

  PROCEDURE one (a : BYTE; VAR b : SIGNED_16);
```

- 8085 B PASCAL -

```
VAR
  c : SIGNED_16;

BEGIN
  c := SIGNED_16 (a) + b;
  c := SIGNED_16 (a) - b;
END.
```

In the first statement an 'XCHG' assembly instruction is missing. In the second statement 4 extra lines are generated and the code generated is incorrect.

Temporary solution:
Reverse the order of the two "operands" in the addition statement. In other words use the expression

```
c := b + SIGNED_16 (a);
```

Signed off 08/25/86 in release 301.60

Number: D200040287 Product: 8085 B PASCAL VAX 64825S003 01.20

Keywords: SETS

One-line description:
SUPERSET or SUBSET checking doesn't work.

```
Problem:
TYPE SET TYPE = SET OF (B0,B1,B2,B3,B4,B5,B6,B7);
VAR X : SET_TYPE;
BEGIN
  IF X <= [B3,B4] THEN; {GENERATES INCORRECT CODE}
  IF X >= [B3,B4] THEN; {GENERATES INCORRECT CODE}
```

Temporary solution:
None at this time.

Signed off 08/25/86 in release 301.60

Number: D200041756 Product: 8085 B PASCAL VAX 64825S003 01.20

One-line description:
Bad code generated for IF.. statement (including WITH).

Signed off 08/25/86 in release 301.60

Number: D200044750 Product: 8085 B PASCAL VAX 64825S003 01.20

Keywords: FOR LOOP

One-line description:
FOR Signed8 := 0 TO 2 DO REAL1 := REAL1/REAL2 overwrites the A-register.

Temporary solution:
Use the compiler option \$AMNESIA +\$

Signed off 08/25/86 in release 301.60

- 8085 B PASCAL -

Number: D200047712 Product: 8085 B PASCAL VAX 64825S003 01.20

One-line description:
TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 301.60

Number: D200052407 Product: 8085 B PASCAL VAX 64825S003 01.50

One-line description:
Incorrect code generated when a CHAR is converted to an UNSIGNED_16.

Problem:
Incorrect code is generated when a CHAR variable is converted to an UNSIGNED_16. The following code is an example:

```
"processor name"
$EXTENSIONS ON$
$RECURSIVE OFF$
PROGRAM PASCALTEST;
TYPE
    BUG_TYPE = UNSIGNED_16;    (*There is no problem if this is
                               SIGNED_16*)
```

```
PROCEDURE BUGGY(COUNT:BUG_TYPE);EXTERNAL;
FUNCTION OPEN:SIGNED_16;
VAR
```

```
    COUNT : BUG_TYPE;
    LEN: CHAR;
BEGIN
    (*THE NEXT TWO STATEMENTS GENERATE INCORRECT CODE*)
    COUNT := BUG_TYPE(LEN);
        (* LD    A,001H          *)
        (* LD    [Dopen+00002H],A *)
        (* LD    A,[Dopen+00004H] *)
        (* LD    [Dopen+00003H],A *)
    BUGGY(BUG_TYPE(LEN));
        (* LD    A,001H          *)
        (* LD    [Dopen+00005H],BC*)
        (* LD    A,[Dopen+00004H] *)
        (* LD    HL,[Dopen+00005H] *)
        (* PUSH  HL              *)
        (* CALL  BUGGY           *)
        (* INC   SP              *)
        (* INC   SP              *)
END;
```

Something very strange occurs when the same code is compiled with \$RECURSIVE ON\$. The statement BUGGY(BUG_TYPE(LEN)); generates the following code:

```
LD    A,001H
LD    [IX-11],A
LD    [IX-10],WHAT???
```

- 8085 B PASCAL -

```
LD    A,[IX-5]
LD    L,A
LD    H,[IX-10]
PUSH  HL
CALL  BUGGY
INC   SP
INC   SP
```

Temporary solution:
No known temporary solution.

Signed off 08/25/86 in release 301.60

Number: D200052696 Product: 8085 B PASCAL VAX 64825S003 01.50

One-line description:
Missing semicolon causes compiler to hang in Pass 1.

Problem:
The following code causes the 64000 to hang in pass 1. An error is generated on the hosts stating that parsing has stopped at a particular line number.

```
"BZ80"
PROGRAM MAIN;
TYPE
STRUCTURED= RECORD
    INT1:INTEGER;
    INT2:INTEGER;
END;
```

```
PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
    I:=P1    <--This missing semicolon causes the problem
    I:=P1.2;
    I:=P2;
END;
```

```
BEGIN
END.
```

Temporary solution:
If the compiler hangs, look for a statement without a semicolon. On the 64000, the status line will show which line of code it stopped on. On the hosts, the error message generated indicates which line of code parsing stopped on.

Signed off 08/25/86 in release 301.60

Number: D200058875 Product: 8085 B PASCAL VAX 64825S003 01.50

Keywords: PREPROCESSOR

One-line description:
Preprocessor reports errors when symbols hp64000, vms or hpux w #if

- 8085 B PASCAL -

Signed off 08/25/86 in release 301.60

Number: D200059279 Product: 8085 B PASCAL VAX 64825S003 01.50

One-line description:

Host compilers do not put absolute pats specifications in relocatables

Signed off 08/25/86 in release 301.60

Number: D200049098 Product: 8085 B PASCAL VAX 64825S003 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 301.60

Number: 5000135780 Product: 8085 C 64826 01.02

One-line description:

Function return address is incorrect and program returns to wrong place.

Problem:

When a pointer is passed to a function with \$RECURSIVE ON\$, the return address is incorrect, causing the program to return to the wrong address. This problem occurs when the function call is not part of an assignment statement.

Temporary solution:

Assign the return value of the function call to a dummy variable. This will cause the compiler to generate the correct return address.

Signed off 08/25/86 in release 601.03

Number: D200013995 Product: 8085 C 64826 01.01

Keywords: PASS 1

One-line description:

No warning or error: taking the sizeof a struct var. not declared.

Problem:

The compiler should generate an error in the following code.

```
"C"
"8085"
main () {
    int y;
    y = sizeof(struct x);
}
```

If x is not declared or is declared as anything other than a structure, the program compiles with no error messages or warnings. It stores as the size zero bytes.

Signed off 08/25/86 in release 601.03

Number: D200025387 Product: 8085 C 64826 01.01

Keywords: CODE GENERATOR

One-line description:

Dereferenced and incremented 2nd field of structure fails when parameter

Problem:

When the second pointer field of a structure is dereferenced and incremented and passed as a parameter, the code generated puts the result in the data area instead of back on the stack for the calling routine. This does not occur with any other field in the structure, only the second one.

Example:

"C"

```
"8085"
struct strct { char *ptr1; char *ptr2; };
func(strct_ptr)
struct strct *strct_ptr;
{
  ++strct_ptr -> ptr1;
  ++strct_ptr -> ptr2; /* This expression causes the problem */
}
```

Temporary solution:

Assign the dereferenced field to a temporary variable of the appropriate type, then increment the temporary variable. Finally, assign the temporary variable to the dereferenced structure field:

```
struct strct { char *ptr1; char *ptr2; };
func(strct_ptr)
struct strct *strct_ptr;
{
  int temp1;
  ++strct_ptr -> ptr1;
  temp1 = strct_ptr -> ptr2;
  ++temp1;
  strct_ptr -> ptr2 = temp1;
}
```

Signed off 08/25/86 in release 601.03

Number: D200026781 Product: 8085 C 64826 01.01

One-line description:

Incorrect code gen by assignment to deref'd 8 bit field of structure.

Problem:

When an 8 bit field of a structure is dereferenced and used as the left hand side of an assignment statement using the += operator, incorrect code is generated. This does not occur with the first field in the structure. The incorrect code is an LHLD Dmain instruction which loads H and L with garbage since Dmain is uninitialized. The following code is an example of this:

```
"C"
"processor name"
$RECURSIVE OFF$
main() {
  extern char KEY,X1();
  struct ROW {
    char A;
    char B;
  } *PTR;
```

```
PTR->B+=X1(KEY); /*This instruction generates an incorrect
                 LHLD Dmain instruction*/
```

If the = operator is used instead of the += operator in the assignment statement, the problem does not occur.

Temporary solution:

Use a temporary variable:
temp = PTR->B;

```
temp+=X1(KEY);
PTR->B = temp;
```

Signed off 08/25/86 in release 601.03

Number: D200027805 Product: 8085 C 64826 01.01

One-line description:

No form feed between the expanded listing and the cross reference table.

Problem:

During compilation, with XREF option on, the compiler does not provide a form feed (FF) in the listing file. The XREF starts on the same page as the end of the listing. Also, the page number says 535 when it should be page 2.

Temporary solution:

After compiling with the xref option, edit the expanded listing file and insert a "control L" before the beginning of the cross reference listing.

Signed off 08/25/86 in release 601.03

Number: D200027912 Product: 8085 C 64826 01.01

One-line description:

Addition of dereferenced pointers to structures may fail.

Problem:

Adding two operands that are dereferenced pointers to structures may fail because the compiler forgets to store the H and L registers and overwrites them. The following code is an example of this:

```
"C"
"processor name"
struct tree {
  int distance;
  int x_start;
  int x_range;
};
trees(treex)
struct tree *treex;
{
  treex->distance=treex->x_start+treex->x_range; /*This line
                                                generates an ADD HL,DE instruction to index
                                                into the structure tree, but overwrites H and L
                                                in the next instruction instead of storing it*/
```

Temporary solution:

Use local temporary variables of the appropriate types to store the values of the dereferenced structure pointers before using them in a complex expression. Depending on the complexity of the expression, more than one temporary variable may have to be used.

```
trees(treex)
struct tree *treex;
{
```

```
int x;
x = treex->x_start;
treex->distance= x + treex->x_range;
}
```

Signed off 08/25/86 in release 601.03

Number: D200031104 Product: 8085 C 64826 01.01

One-line description:

++ and -- operators evaluated with improper precedence.

Problem:

According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

```
Example 1: array[index++] = 1;
Example 2: array[index] = 1;
           index++;
```

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:

Separate the expression as shown in example 2.

Signed off 08/25/86 in release 601.03

Number: D200033258 Product: 8085 C 64826 01.01

One-line description:

Comparing character to zero in while loop generates incorrect code.

Problem:

If you compare a character variable to zero in a while loop, incorrect code is generated. The following code demonstrates the problem.

```
"C"
"6809"

proc()
{
char timeout = 10;

while(timeout--); /* Code generated here causes infinite loop.
}
```

The code generated for the while loop clears the A register and then compares the D register to -1. Therefore the condition is never met.

Temporary solution:

Declare the variable used in the test condition as an integer.

```
"C"
"6809"
```

```
proc()
{
int timeout = 10;

while (timeout--);
}
```

Signed off 08/25/86 in release 601.03

Number: D200034298 Product: 8085 C 64826 01.01

Keywords: CODE GENERATOR

One-line description:

A shift assignment operation (<<=) generates incorrect code.

Problem:

If a shift assignment is used instead of a shift within an assignment, the compiler uses the high byte of the variable to be used as the shift counter instead of the low byte. The following is an example:

```
"C"
"processor name"
char data=1;
int shift=4;
main () {
data=data<<shift; /* works correctly */
data<<=shift; /* uses higher order byte of "shift" */
}
```

Temporary solution:

```
Use
data=data<<shift;
instead of
data<<=shift;
```

Signed off 08/25/86 in release 601.03

Number: D200035923 Product: 8085 C 64826 01.01

Keywords: CODE GENERATOR

One-line description:

16 bit comparison on a 8 bit unsigned short field.

Problem:

Improper code is generated for a statement involving unsigned short variables unless they are explicitly cast as unsigned short.

```
main()
{
static unsigned short digit_index;
static unsigned short digit[12];
int a,b;
if (digit[digit_index--]){
a=4;
b=4;}
else{
```

```
a=5;
b=5;}
}
```

Improper code is generated for the comparison (ie The comparison is done on 16 bits (8 of which have been cleared) Against #0FFFFH.

12/10/85: The problem also arises if you compare a constant against an unsigned short. For example if you declared:

```
#define constant ~0
unsigned short var;
and later compared these two the compiler will zero out the upper byte of the variable var and then compare it to FFFFH. Thus, the condition is never met.
```

12/16/85: Another example of incorrect code being generated when a char variable is used in a test condition is as follows:

```
char a;
main()
{
  a = -1;
  if(a == -1)
    a = 'A';
}
```

Temporary solution:

Correct code is generated if the line in question is changed to the following although digit[] has already been declared unsigned short.

```
if ((unsigned short)digit[digit_index]--){
12/10/85: Declare the constant as a short. In other words:
#define constant 0FFH.
12/16/85: If only 128 valid characters are required the variable can be declared as a short int.
```

Signed off 08/25/86 in release 601.03

Number: D200037465 Product: 8085 C 64826 01.01

One-line description:

Run time UNDERFLOW error using ZDSBSUB library if result has even parity

Problem:

Byte subtraction with \$DEBUG ON\$ will cause an underflow error if the result has even parity. An underflow will be incorrectly flagged if the result has even parity. No error will be indicated, even if one exists, if the result has odd parity. The problem is in ZDsbsub (Debug signed byte subtraction). The 8085 interprets PE exclusively as a parity bit, while the library is anticipating that the bit can be interpreted as an overflow bit.

SAMPLE CODE:

```
"C"
"8085"
$DEBUG ON$ /*This is required for the error to occur*/
main()
{
```

- 8085 C -

```
short small;
short zero;
small = -128;
zero = small - small; /* causes error */
}
```

This problem affects 8085 C and Pascal compilers on 64000 and hosts.

Temporary solution:

Turn \$DEBUG OFF\$ around signed byte subtractions.

Signed off 08/25/86 in release 601.03

Number: D200040816 Product: 8085 C 64826 01.01

Keywords: PASS 3

One-line description:

Pass 3 fails to detect relative jump address out-of-range.

Problem:

Pass 3 of the compilation process may fail to detect a relative jump which is out of range. In the test program submitted the relative jump is generated for an IF..THEN statement while the compiler option OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:

As a temporary work around disable the compiler option OPTIMIZE around those sections of code which are suspect.

Signed off 08/25/86 in release 601.03

Number: D200041376 Product: 8085 C 64826 01.01

One-line description:

Problem with integer pointer in conditional statement.

Problem:

In the following example, two loads are performed, but no other code is generated to check for zero value.

```
"C"
"processor name"
#define NULL 0
fct(parm)
int *parm;
{
  if (parm - NULL)
    parm = 10;
}
```

Signed off 08/25/86 in release 601.03

- 8085 C -

Number: D200046037 Product: 8085 C 64826 01.01

One-line description:
Post increment of pointer results in incorrect code.

Problem:
Post increment of a pointer value will cause incorrect code to be generated. First, the pointer is pre-incremented rather than post incremented. Secondly, the result is stored in the wrong location.

```
"C"
"8085"
$SHORT_ARITH +$
$RECURSIVE OFF$
$SEPARATE ON$
```

```
main()
{
  long ai[2], *aiptr, a1, a2;
  ai[0]=0L;
  ai[1]=1L;
  aiptr=ai;
  ai=*aiptr++; /* Problem Statement. *aiptr is pre-incremented
               and the result is stored in wrong location. */
}
```

Temporary solution:
Increment the pointer after the assignment is made.
Use: a1=*aiptr;
*aiptr++;

Rather than:
a1=*aiptr++;

Signed off 08/25/86 in release 601.03

Number: D200047720 Product: 8085 C 64826 01.01

One-line description:
TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 601.03

Number: D200053777 Product: 8085 C 64826 01.02

One-line description:
Incorrect code for multiplication dependent on order of operands.

Problem:
The following example generates incorrect code:

```
"C"
"8085"
int count;
char cnt_buf[0];
main()
{
  cnt_buf[0] = count - cnt_buf[2]*100 - cnt_buf[1]*10;
```

- 8085 C -

}

The result of the second multiplication, cnt_buf[1]*10, is stored in a temporary location and never retrieved. Also, just before storing what the compiler thinks is the result of the entire expression, it subtracts part of the address of one of the temporary locations from the result of count - cnt_buf[2]*100.

Temporary solution:
This problem is dependent on the order of the operands that are multiplied. By changing the order as shown below, the problem does not occur.

```
"C"
"8085"
int count;
char cnt_buf[0];
main()
{
  cnt_buf[0] = count - 100*cnt_buf[2] - 10*cnt_buf[1];
}
```

Signed off 08/25/86 in release 601.03

Number: D200055277 Product: 8085 C 64826 01.02

One-line description:
Compiler loses track of array index.

Problem:
With \$RECURSIVE ON\$, the compiler loses track of where on the stack it has put certain variables. The following code is an example of this problem:

```
"C"
"processor name"
$RECURSIVE ON$
index()
{
  int xdigit[80];
  short i;
  i = 9;
  (*LXI H, -(Iindex+00001H) *)
  (*DAD SP *)
  (*MVI M, 009H *)

  xdigit[i++] = 10;
  (*MOV A, M *)
  (*INR A (*another defect, D200031104*)*)
  (*MOV M, A *)
  (*LXI H, -(Iindex+000A1H) *)
  (*DAD SP *)
  (*XCHG *)
  (*LXI H, -(Iindex+000A2H) *) wrong!
  (* ..... *)
```

- 8085 C -

}

Temporary solution:
No known temporary solution.

Signed off 08/25/86 in release 601.03

Number: D200050757 Product: 8085 C 300 64826S004 01.00

One-line description:
Defining TRUE and FALSE as global may result in duplicate symbol names.

Problem:
Defining the variables (constants) TRUE and FALSE to be global may result in a duplicate symbol error during a link. These variables are incorrectly defined as global in the Zwordcmp routine located in 'Zlibrary'.

NOTE: Redefining the values of TRUE and/or FALSE is not a legal Pascal operation. Redefinition of these constants is therefore not supported when using the HP 64000 compiler.

Temporary solution:
Obtain the source to Zwordcmp from your local HP Systems Engineer.

Signed off 08/25/86 in release 401.10

Number: D200051318 Product: 8085 C 300 64826S004 01.00

One-line description:
++ and -- operators evaluated with improper precedence.

Problem:
According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

Example 1: array[index++] = 1;

Example 2: array[index] = 1;
index++;

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:
Separate the expression as shown in example 2.

Signed off 08/25/86 in release 401.10

Number: D200052001 Product: 8085 C 300 64826S004 01.00

One-line description:
Run time UNDERFLOW error using ZDSBSUB library if result has even parity

Problem:
Byte subtraction with \$DEBUG ON\$ will cause an underflow error if the result has even parity. An underflow will be incorrectly flagged if the result has even parity. No error will be indicated, even if one exists, if the result has odd parity. The problem is in ZDsbsub (Debug signed byte subtraction). The 8085 interprets PE exclusively as a parity bit, while the library is anticipating that the bit can be interpreted as an overflow bit.

SAMPLE CODE:

```
"C"
"8085"
$DEBUG ON$ /*This is required for the error to occur*/
main()
{
    short small;
    short zero;
    small = -128;
    zero = small - small; /* causes error */
}
```

This problem affects 8085 C and Pascal compilers on 64000 and hosts.

Temporary solution:

Turn \$DEBUG OFF\$ around signed byte subtractions.

Signed off 08/25/86 in release 401.10

Number: D200055293 Product: 8085 C 300 64826S004 01.00

One-line description:

Compiler loses track of array index.

Problem:

With \$RECURSIVE ON\$, the compiler loses track of where on the stack it has put certain variables. The following code is an example of this problem:

```
"C"
"processor name"
$RECURSIVE ON$
index()
{
    int xdigit[80];
    short i;
    i = 9;
        (*LXI H, -(Iindex+00001H) *)
        (*DAD SP *)
        (*MVI M, 009H *)

    xdigit[i++] = 10;
        (*MOV A, M *)
        (*INR A (*another defect, D200031104*)*)
        (*MOV M, A *)
        (*LXI H, -(Iindex+000A1H) *)
        (*DAD SP *)
        (*XCHG *)
        (*LXI H, -(Iindex+000A2H) *) wrong!
        (* ..... *)
}
```

Temporary solution:

No known temporary solution.

Signed off 08/25/86 in release 401.10

- 8085 C -

Number: D200059113 Product: 8085 C 300 64826S004 01.00

One-line description:

Host compilers do not put absolute pats specifications in relocatables

Problem:

Host compilers do not specify the full path name in the relocatable file.

Temporary solution:

No known temporary solution.

Signed off 08/25/86 in release 401.10

Number: D200049130 Product: 8085 C 300 64826S004 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 401.10

- 8085 C -

Number: D200025692 Product: 8085 C 500 64826S001 01.10

Keywords: CODE GENERATOR

One-line description:

Dereferenced and incremented 2nd field of structure fails when parameter

Problem:

When the second pointer field of a structure is dereferenced and incremented and passed as a parameter, the code generated puts the result in the data area instead of back on the stack for the calling routine. This does not occur with any other field in the structure, only the second one.

Example:

```
"C"
"8085"
struct strt { char *ptr1; char *ptr2; };
func(strct_ptr)
struct strt *strct_ptr;
{
  ++strct_ptr -> ptr1;
  ++strct_ptr -> ptr2; /* This expression causes the problem */
}
```

Temporary solution:

Assign the dereferenced field to a temporary variable of the appropriate type, then increment the temporary variable. Finally, assign the temporary variable to the dereferenced structure field:

```
struct strt { char *ptr1; char *ptr2; };
func(strct_ptr)
struct strt *strct_ptr;
{
  int temp1;
  ++strct_ptr -> ptr1;
  temp1 = strct_ptr -> ptr2;
  ++temp1;
  strct_ptr -> ptr2 = temp1;
}
```

Signed off 08/25/86 in release 101.50

Number: D200027011 Product: 8085 C 500 64826S001 01.10

One-line description:

Incorrect code gen by assignment to deref'd 8 bit field of structure.

Problem:

When an 8 bit field of a structure is dereferenced and used as the left hand side of an assignment statement using the += operator, incorrect code is generated. This does not occur with the first field in the structure. The incorrect code is an LHLD Dmain instruction which loads H and L with garbage since Dmain is uninitialized. The following code is an example of this:

"C"

- 8085 C -

```
"processor name"
$RECURSIVE OFF$
main() {
  extern char KEY,X1();
  struct ROW {
    char A;
    char B;
  } *PTR;
  PTR->B+=X1(KEY); /*This instruction generates an incorrect
  } LHLD Dmain instruction*/
  If the = operator is used instead of the += operator in the assignment
  statement, the problem does not occur.
```

Temporary solution:

Use a temporary variable:

```
temp = PTR->B;
temp+=X1(KEY);
PTR->B = temp;
```

Signed off 08/25/86 in release 101.50

Number: D200027920 Product: 8085 C 500 64826S001 01.10

One-line description:

Addition of dereferenced pointers to structures may fail.

Problem:

Adding two operands that are dereferenced pointers to structures may fail because the compiler forgets to store the H and L registers and overwrites them. The following code is an example of this:

```
"C"
"processor name"
struct tree {
  int distance;
  int x_start;
  int x_range;
};
trees(treex)
struct tree *treex;
{
  treex->distance=treex->x_start+treex->x_range; /*This line
  } generates an ADD HL,DE instruction to index
  into the structure tree, but overwrites H and L
  in the next instruction instead of storing it*/
```

Temporary solution:

Use local temporary variables of the appropriate types to store the values of the dereferenced structure pointers before using them in a complex expression. Depending on the complexity of the expression, more than one temporary variable may have to be used.

```
trees(treex)
struct tree *treex;
{
  int x;
  x = treex->x_start;
```

- 8085 C -

```

    treex->distance= x + treex->x_range;
}

```

Signed off 08/25/86 in release 101.50

Number: D200031450 Product: 8085 C 500 64826S001 01.10

One-line description:

++ and -- operators evaluated with improper precedence.

Problem:

According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

Example 1: array[index++] = 1;

Example 2: array[index] = 1;

index++;

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:

Separate the expression as shown in example 2.

Signed off 08/25/86 in release 101.50

Number: D200033266 Product: 8085 C 500 64826S001 01.10

One-line description:

Comparing character to zero in while loop generates incorrect code.

Problem:

If you compare a character variable to zero in a while loop, incorrect code is generated. The following code demonstrates the problem.

```

"C"
"6809"

```

```

proc()
{
    char timeout = 10;

    while(timeout--); /* Code generated here causes infinite loop.
}

```

The code generated for the while loop clears the A register and then compares the D register to -1. Therefore the condition is never met.

Temporary solution:

Declare the variable used in the test condition as an integer.

```

"C"
"6809"

proc()
{
    int timeout = 10;

```

- 8085 C -

```

    while (timeout--);
}

```

Signed off 08/25/86 in release 101.50

Number: D200034306 Product: 8085 C 500 64826S001 01.10

Keywords: CODE GENERATOR

One-line description:

A shift assignment operation (<=>) generates incorrect code.

Problem:

If a shift assignment is used instead of a shift within an assignment, the compiler uses the high byte of the variable to be used as the shift counter instead of the low byte. The following is an example:

```

"C"
"procesor name"
char data=1;
int shift=4;
main () {
    data=data<<shift; /* works correctly */
    data<=<=shift; /* uses higher order byte of "shift" */
}

```

Temporary solution:

Use

```

    data=data<<shift;
instead of
    data<=<=shift;

```

Signed off 08/25/86 in release 101.50

Number: D200035931 Product: 8085 C 500 64826S001 01.10

Keywords: CODE GENERATOR

One-line description:

16 bit comparison on an 8 bit unsigned short field.

Problem:

Improper code is generated for statements involving unsigned short variables unless they are explicitly cast as unsigned shorts.

```

main()
{
    static unsigned short digit_index;
    static unsigned short digit[12];
    int a,b;
    if (digit[digit_index]--){
        a=4;
        b=4;}
    else{
        a=5;
        b=5;}
}

```

- 8085 C -

Improper code is generated for the comparison (ie The comparison is done on 16 bits (8 of which have been cleared) against #0FFFFH.

12/10/85: The problem also arises if you compare a constant against an unsigned short. For example if you declared:

```
#define constant ~0
unsigned short var;
and later compared these two the compiler will zero out the upper byte
of the variable var and then compare it to FFFFH. Thus, the condition
is never met.
```

12/16/85: Another example of incorrect code being generated when a char variable is used in a test condition is as follows:

```
char a;
main()
{
  a = -1;
  if(a == -1)
    a = 'A';
}
```

Temporary solution:

Correct code is generated if the line in question is changed to the following although digit[] has already been declared unsigned short.

```
if ((unsigned short)digit[digit_index--]){
12/10/85: Declare the constant as a short. In other words:
#define constant 0FFH.
12/16/85: If only 128 valid characters are required the variable can
be declared as a short int.
```

Signed off 08/25/86 in release 101.50

Number: D200037218 Product: 8085 C 500 64826S001 01.20

Keywords: PASS 3

One-line description:

Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:

Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect data being output to the list file. In selected cases, machine code will be incorrectly listed. For example, consider the following Pascal program.

```
$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;

VAR
  a, b : BOOLEAN;

PROCEDURE one;

BEGIN
  a := b;
```

- 8085 C -

END;

In the example listed above, the output file will denote machine code of the form FFFFC00001 for one of the generated assembly statements. The correct value should be C8000001. This problem is caused by an incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 101.50

Number: D200040618 Product: 8085 C 500 64826S001 01.20

One-line description:

Run time UNDERFLOW error using ZDSBSUB library if result has even parity

Problem:

Byte subtraction with \$DEBUG ON\$ will cause an underflow error if the result has even parity. An underflow will be incorrectly flagged if the result has even parity. No error will be indicated, even if one exists, if the result has odd parity. The problem is in ZDsbsub (Debug signed byte subtraction). The 8085 interprets PE exclusively as a parity bit, while the library is anticipating that the bit can be interpreted as an overflow bit.

SAMPLE CODE:

```
"C"
"8085"
$DEBUG ON$ /*This is required for the error to occur*/
main()
{
  short small;
  short zero;
  small = -128;
  zero = small - small; /* causes error */
}
```

This problem affects 8085 C and Pascal compilers on 64000 and hosts.

Temporary solution:

Turn \$DEBUG OFF\$ around signed byte subtractions.

Signed off 08/25/86 in release 101.50

Number: D200040824 Product: 8085 C 500 64826S001 01.20

Keywords: PASS 3

One-line description:

Pass 3 fails to detect relative jump address out-of-range.

Problem:

Pass 3 of the compilation process may fail to detect a relative jump which is out of range. In the test program submitted the relative jump is generated for an IF..THEN statement while the compiler option

- 8085 C -

OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:

As a temporary work around disable the compiler option OPTIMIZE around those sections of code which are suspect.

Signed off 08/25/86 in release 101.50

Number: D200041384 Product: 8085 C 500 64826S001 01.20

One-line description:

Problem with integer pointer in conditional statement.

Problem:

In the following example, two loads are performed, but no other code is generated to check for zero value.

```
"C"
"processor name"
#define NULL 0
fct(parm)
int *parm;
{
    if (parm - NULL)
        parm = 10;
}
```

Signed off 08/25/86 in release 101.50

Number: D200046011 Product: 8085 C 500 64826S001 01.20

One-line description:

Title description is incorrect.

Signed off 08/25/86 in release 101.50

Number: D200046201 Product: 8085 C 500 64826S001 01.20

One-line description:

Post increment of pointer results in incorrect code.

Problem:

Post increment of a pointer value will cause incorrect code to be generated. First, the pointer is pre-incremented rather than post incremented. Secondly, the result is stored in the wrong location.

```
"C"
"8085"
$SHORT ARITH +$
$RECURSIVE OFF$
$SEPARATE ON$

main()
{
    long ai[2],*aiptr,a1,a2;
    ai[0]=0L;
    ai[1]=1L;
    aiptr=ai;
```

- 8085 C -

```
ai=*aiptr++; /* Problem Statement. *aiptr is pre-incremented
and the result is stored in wrong location. */
```

Temporary solution:

Increment the pointer after the assignment is made.

```
Use: ai=*aiptr;
    *aiptr++;
```

Rather than:

```
ai=*aiptr++;
```

Signed off 08/25/86 in release 101.50

Number: D200047738 Product: 8085 C 500 64826S001 01.20

One-line description:

TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 101.50

Number: D200049809 Product: 8085 C 500 64826S001 00.00

One-line description:

NO CROSS REFERENCE TABLE IS GENERATED

Problem:

"C" COMPILERS DO NOT GENERATE A CROSS REFERENCE TABLE ON THE VAX.

Temporary solution:

NONE KNOWN AT PRESENT

Signed off 04/18/86 in release 101.50

Number: D200055251 Product: 8085 C 500 64826S001 01.40

One-line description:

Compiler loses track of array index.

Problem:

With \$RECURSIVE ON\$, the compiler loses track of where on the stack it has put certain variables. The following code is an example of this problem:

```
"C"
"processor name"
$RECURSIVE ON$
index()
{
    int xdigit[80];
    short i;
    i = 9;
    (*LXI H, -(Iindex+00001H) *)
    (*DAD SP *)
    (*MVI M, 009H *)

    xdigit[i++] = 10;
```

- 8085 C -

```

(*MOV  A,M      *)
(*INR  A, (*another defect, D200031104*)*)
(*MOV  M,A      *)
(*LXI  H,-(Iindex+000A1H) *)
(*DAD  SP       *)
(*XCHG                *)
(*LXI  H,-(Iindex+000A2H) *) wrong!
(* .....*)

```

}

Temporary solution:
No known temporary solution.

Signed off 08/25/86 in release 101.50

Number: D200059097 Product: 8085 C 500 64826S001 01.40

One-line description:
Host compilers do not put absolute pats specifications in relocatables

Problem:
Host compilers do not specify the full path name in the relocatable file.

Temporary solution:
No known temporary solution.

Signed off 08/25/86 in release 101.50

Number: D200049114 Product: 8085 C 500 64826S001 00.00

One-line description:
Linker output file should use alternate file extension.

Signed off 08/25/86 in release 101.50

Number: D200025700 Product: 8085 C VAX 64826S003 01.10

Keywords: CODE GENERATOR

One-line description:
Dereferenced and incremented 2nd field of structure fails when parameter

Problem:
When the second pointer field of a structure is dereferenced and incremented and passed as a parameter, the code generated puts the result in the data area instead of back on the stack for the calling routine. This does not occur with any other field in the structure, only the second one.

Example:
"C"
"8085"
struct strt { char *ptr1; char *ptr2; };
func(strt_ptr)
struct strt *strt_ptr;
{
++strt_ptr -> ptr1;
++strt_ptr -> ptr2; /* This expression causes the problem */
}

Temporary solution:
Assign the dereferenced field to a temporary variable of the appropriate type, then increment the temporary variable. Finally, assign the temporary variable to the dereferenced structure field:

```

struct strt { char *ptr1; char *ptr2; };
func(strt_ptr)
struct strt *strt_ptr;
{
  int temp1;
  ++strt_ptr -> ptr1;
  temp1 = strt_ptr -> ptr2;
  ++temp1;
  strt_ptr -> ptr2 = temp1;
}

```

Signed off 08/25/86 in release 301.80

Number: D200027029 Product: 8085 C VAX 64826S003 01.20

One-line description:
Incorrect code gen by assignment to deref'd 8 bit field of structure.

Problem:
When an 8 bit field of a structure is dereferenced and used as the left hand side of an assignment statement using the += operator, incorrect code is generated. This does not occur with the first field in the structure. The incorrect code is an LHLD Dmain instruction which loads H and L with garbage since Dmain is uninitialized. The following code is an example of this:

"C"

```

"processor name"
$RECURSIVE OFF$
main() {
extern char KEY,X1();
struct ROW {
    char A;
    char B;
} *PTR;
PTR->B+=X1(KEY);    /*This instruction generates an incorrect
                    LHL Dmain instruction*/
}
If the = operator is used instead of the += operator in the assignment
statement, the problem does not occur.

```

Temporary solution:
 Use a temporary variable:
 temp = PTR->B;
 temp+=X1(KEY);
 PTR->B = temp;

Signed off 08/25/86 in release 301.80

 Number: D200027938 Product: 8085 C VAX 64826S003 01.20

One-line description:
 Addition of dereferenced pointers to structures may fail.

Problem:
 Adding two operands that are dereferenced pointers to structures may fail because the compiler forgets to store the H and L registers and overwrites them. The following code is an example of this:

```

"C"
"processor name"
struct tree {
    int distance;
    int x_start;
    int x_range;
};
trees(treex)
struct tree *treex;
{
    treex->distance=treex->x_start+treex->x_range; /*This line
    generates an ADD HL,DE instruction to index
    into the structure tree, but overwrites H and L
    in the next instruction instead of storing it*/
}

```

Temporary solution:
 Use local temporary variables of the appropriate types to store the values of the dereferenced structure pointers before using them in a complex expression. Depending on the complexity of the expression, more than one temporary variable may have to be used.

```

trees(treex)
struct tree *treex;
{
    int x;
    x = treex->x_start;
}

```

```

    treex->distance= x + treex->x_range;
}

```

Signed off 08/25/86 in release 301.80

 Number: D200031468 Product: 8085 C VAX 64826S003 01.20

One-line description:
 ++ and -- operators evaluated with improper precedence.

Problem:
 According to Kernighan and Ritchie, page 43, the following expressions are equivalent:
 Example 1: array[index++] = 1;
 Example 2: array[index] = 1;
 index++;
 However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:
 Separate the expression as shown in example 2.

Signed off 08/25/86 in release 301.80

 Number: D200033274 Product: 8085 C VAX 64826S003 01.20

One-line description:
 Comparing character to zero in while loop generates incorrect code.

Problem:
 If you compare a character variable to zero in a while loop, incorrect code is generated. The following code demonstrates the problem.

```

"C"
"6809"
proc()
{
    char timeout = 10;
    while(timeout--); /* Code generated here causes infinite loop.
}

```

The code generated for the while loop clears the A register and then compares the D register to -1. Therefore the condition is never met.

Temporary solution:
 Declare the variable used in the test condition as an integer.

```

"C"
"6809"
proc()
{
    int timeout = 10;
}

```

```
while (timeout--);
}
```

Signed off 08/25/86 in release 301.80

Number: D200034314 Product: 8085 C VAX 64826S003 01.20

Keywords: CODE GENERATOR

One-line description:

A shift assignment operation (<<=) generates incorrect code.

Problem:

If a shift assignment is used instead of a shift within an assignment, the compiler uses the high byte of the variable to be used as the shift counter instead of the low byte. The following is an example:

```
"C"
"processor name"
char data=1;
int shift=4;
main () {
    data=data<<shift; /* works correctly */
    data<<=shift; /* uses higher order byte of "shift" */
}
```

Temporary solution:

Use
instead of
data<<=shift;

Signed off 08/25/86 in release 301.80

Number: D200035949 Product: 8085 C VAX 64826S003 01.20

Keywords: CODE GENERATOR

One-line description:

16 bit comparison on a 8 bit unsigned short field.

Problem:

Improper code is generated for statements involving unsigned short variables unless they are explicitly cast as unsigned shorts.

```
main()
{
    static unsigned short digit_index;
    static unsigned short digit[12];
    int a,b;
    if (digit[digit_index--]){
        a=4;
        b=4;}
    else{
        a=5;
        b=5;}
}
```

Improper code is generated for the comparison (ie the comparison is done on 16 bits (8 of which have been cleared) against #0FFFFH.

12/10/85: The problem also arises if you compare a constant against an unsigned short. For example if you declared:

```
#define constant ~0
unsigned short var;
and later compared these two the compiler will zero out the upper byte
of the variable var and then compare it to FFFFH. Thus, the condition
is never met.
```

12/16/85: Another example of incorrect code being generated when a char variable is used in a test condition is as follows:

```
char a;
main()
{
    a = -1;
    if(a == -1)
        a = 'A';
}
```

Temporary solution:

Correct code is generated if the line in question is changed to the following although digit[] has already been declared unsigned short.

```
if ((unsigned short)digit[digit_index--]){
12/10/85: Declare the constant as a short. In other words:
#define constant 0FFH.
12/16/85: If only 128 valid characters are required the variable can
be declared as a short int.
```

Signed off 08/25/86 in release 301.80

Number: D200037226 Product: 8085 C VAX 64826S003 01.20

Keywords: PASS 3

One-line description:

Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:

Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect data being output to the list file. In selected cases, machine code will be incorrectly listed. For example, consider the following Pascal program.

```
$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;

VAR
    a, b : BOOLEAN;

PROCEDURE one;

BEGIN
    a := b;
```

END;

In the example listed above, the output file will denote machine code of the form FFFFC0001 for one of the generated assembly statements. The correct value should be C800001. This problem is caused by an incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 301.80

Number: D200040626 Product: 8085 C VAX 64826S003 01.20

One-line description:

Run time UNDERFLOW error using ZDSBSUB library if result has even parity

Problem:

Byte subtraction with \$DEBUG ON\$ will cause an underflow error if the result has even parity. An underflow will be incorrectly flagged if the result has even parity. No error will be indicated, even if one exists, if the result has odd parity. The problem is in Zdsbsub (Debug signed byte subtraction). The 8085 interprets PE exclusively as a parity bit, while the library is anticipating that the bit can be interpreted as an overflow bit.

SAMPLE CODE:

```
"C"
"8085"
$DEBUG ON$ /*This is required for the error to occur*/
main()
{
    short small;
    short zero;
    small = -128;
    zero = small - small; /* causes error */
}
```

This problem affects 8085 C and Pascal compilers on 64000 and hosts.

Temporary solution:

Turn \$DEBUG OFF\$ around signed byte subtractions.

Signed off 08/25/86 in release 301.80

Number: D200040832 Product: 8085 C VAX 64826S003 01.20

Keywords: PASS 3

One-line description:

Pass 3 fails to detect relative jump address out-of-range.

Problem:

Pass 3 of the compilation process may fail to detect a relative jump which is out of range. In the test program submitted the relative jump is generated for an IF..THEN statement while the compiler option

- 8085 C -

OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:

As a temporary work around disable the compiler option OPTIMIZE around those sections of code which are suspect.

Signed off 08/25/86 in release 301.80

Number: D200041392 Product: 8085 C VAX 64826S003 01.20

One-line description:

Problem with integer pointer in conditional statement.

Problem:

In the following example, two loads are performed, but no other code is generated to check for zero value.

```
"C"
"processor name"
#define NULL 0
fct(parm)
int *parm;
{
    if (parm - NULL)
        parm = 10;
}
```

Signed off 08/25/86 in release 301.80

Number: D200046029 Product: 8085 C VAX 64826S003 01.20

One-line description:

Title description is incorrect.

Signed off 08/25/86 in release 301.80

Number: D200046219 Product: 8085 C VAX 64826S003 01.20

One-line description:

Post increment of pointer results in incorrect code.

Problem:

Post increment of a pointer value will cause incorrect code to be generated. First, the pointer is pre-incremented rather than post incremented. Secondly, the result is stored in the wrong location.

```
"C"
"8085"
$SHORT_ARITH +$
$RECURSIVE OFF$
$SEPARATE ON$
```

```
main()
{
    long ai[2],*aiptr,a1,a2;
    ai[0]=0L;
    ai[1]=1L;
    aiptr=ai;
```

- 8085 C -


```
ai=*aiptr++; /* Problem Statement. *aiptr is pre-incremented
and the result is stored in wrong location. */
```

Temporary solution:
Increment the pointer after the assignment is made.
Use: ai=*aiptr;
*aiptr++;

Rather than:
ai=*aiptr++;

Signed off 08/25/86 in release 301.80

```
Number: D200047746 Product: 8085 C VAX 64826S003 01.20
```

One-line description:
TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 301.80

```
Number: D200055186 Product: 8085 C VAX 64826S003 01.60
```

One-line description:
Compilation on the VAX using batch mode generates incorrect listing file

Problem:
The test files can be found on the VAX750 under user\$disk:[robin.hughes.rgalo.test]. The following test files were used:

1. MTINHST_C. - File which contains one error- a missing '}' on line 70
2. TMTINHST_C. - Error-free version of MTINHST_C.
3. MTOPNDF_C. - File which contains one error - missing declaration for integer 'j'
4. MTOPNDF_C. - Error-free version of MTOPNDF_C.

One logical name must be defined as follows to access the include files referenced by the test programs:

```
#define BSLN user$disk:[robin.hughes.wsbsln.baseline]
```

When the four files were compiled interactively, the two error-free versions generated correct listings. The first file (MTINHST_C.) generated an incomplete and incorrect listing file. The listing showed the include files inserted first, followed by "C", "8086" and a few other lines of the program. The output displayed on the screen looked like:

```
In pass1.
  70 else
      ^25
  136
      ^408
In C Nocode.
```

```
comp: C Nocode cannot recover from errors.
```

When the third file (MTOPNDF_C.) was compiled, the listing appeared fine except for the insertion a some strange control charaters.

These last two files were compiled in batch mode (file: user\$disk:[robin.hughes.rgalo.test]hughes.com). The first file (MTINHST_C.) generated a complete but incorrect listing. Although two errors were found (25 & 408) the line at the bottom stated that errors = 0. The include file expansion preceeded the "C" and "8086" in the listing, and lines like, #include filename, were still in the file. The error message was at line 72 of the listing instead of line 2472 were the '}' was actual missing. Finally the last 100 lines had useless numbers in the left margin.

When the third file (MTOPNDF_C.) was compiled, an incomplete listing was generated with the include file expansions listed first.

All of these tests were done on the VAX750 with the /e/v/o options.

This problem also occurs on the 68000.

Temporary solution:
No temporary solution available

Signed off 08/25/86 in release 301.80

```
Number: D200055285 Product: 8085 C VAX 64826S003 01.60
```

One-line description:
Compiler loses track of array index.

Problem:
With \$RECURSIVE ON\$, the compiler loses track of where on the stack it has put certain variables. The following code is an example of this problem:

```
"C"
"processor name"
$RECURSIVE ON$
index()
{
  int xdigit[80];
  short i;
  i = 9;
  (*LXI H, -(Iindex+00001H) *)
  (*DAD SP *)
  (*MVI M, 009H *)

  xdigit[i++] = 10;
  (*MOV A, M *)
  (*INR A (*another defect, D200031104*)*)
  (*MOV M, A *)
  (*LXI H, -(Iindex+000A1H) *)
  (*DAD SP *)
  (*XCHG *)
  (*LXI H, -(Iindex+000A2H) *) wrong!
  (* ..... *)
```

}

Temporary solution:
No known temporary solution.

Signed off 08/25/86 in release 301.80

 Number: D200059105 Product: 8085 C VAX 64826S003 01.60

One-line description:
Host compilers do not put absolute pats specifications in relocatables

Problem:
Host compilers do not specify the full path name in the relocatable file.

Temporary solution:
No known temporary solution.

Signed off 08/25/86 in release 301.80

 Number: D200049122 Product: 8085 C VAX 64826S003 00.00

One-line description:
Linker output file should use alternate file extension.

Signed off 08/25/86 in release 301.80

 Number: 5000103218 Product: 8086/8 C 64818 02.00

One-line description:
ASM file created by compiler generates errors when assembled.

Problem:
The ASM file generated by the 8086 C compiler may have errors when assembled.

Signed off 08/25/86 in release 803.01

 Number: D200013961 Product: 8086/8 C 64818 01.06

Keywords: PASS 1

One-line description:
No warning or error: taking the sizeof a struct var. not declared.

Problem:
The compiler should generate an error in the following code.

```
"C"
"8086"
main () {
    int y;
    y = sizeof(struct x);
}
```

If x is not declared or is declared as anything other than a structure, the program compiles with no error messages or warnings. It stores as the size zero bytes.

Signed off 08/25/86 in release 803.01

 Number: D200026427 Product: 8086/8 C 64818 01.06

One-line description:
No error when illegal assignment to a pointer is made.

Problem:
The native compiler on the 9000 flags an error for the following code, but the 8086/8 C compiler does not:

```
main()
{
    char *ptr;
    int i;
    char c;

    (ptr + i) +2 = c;    /*Should flag an error stating illegal
                        left hand side of expression */
}
```

Signed off 08/25/86 in release 803.01

Number: D200027706 Product: 8086/8 C 64818 02.00

One-line description:
No form feed between the expanded listing and the cross reference table.

Problem:
During compilation, with XREF option on, the compiler does not provide a form feed (FF) in the listing file. The XREF starts on the same page as the end of the listing. Also, the page number says 535 when it should be page 2.

Temporary solution:
After compiling with the xref option, edit the expanded listing file and insert a "control L" before the beginning of the cross reference listing.

Signed off 08/25/86 in release 803.01

Number: D200031294 Product: 8086/8 C 64818 02.00

One-line description:
++ and -- operators evaluated with improper precedence.

Problem:
According to Kernighan and Ritchie, page 43, the following expressions are equivalent:
Example 1: array[index++] = 1;
Example 2: array[index] = 1;
 index++;
However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:
Separate the expression as shown in example 2.

Signed off 08/25/86 in release 803.01

Number: D200033100 Product: 8086/8 C 64818 02.00

One-line description:
Comparing character to zero in while loop generates incorrect code.

Problem:
If you compare a character variable to zero in a while loop, incorrect code is generated. The following code demonstrates the problem.

```
"C"
"6809"

proc()
{
    char timeout = 10;

    while(timeout--); /* Code generated here causes infinite loop.
```

- 8086/8 C -

```
}
```

The code generated for the while loop clears the A register and then compares the D register to -1. Therefore the condition is never met.

Temporary solution:
Declare the variable used in the test condition as an integer.

```
"C"
"6809"

proc()
{
    int timeout = 10;

    while (timeout--);
}
```

Signed off 08/25/86 in release 803.01

Number: D200035782 Product: 8086/8 C 64818 02.00

Keywords: CODE GENERATOR

One-line description:
16 bit comparison on a 8 bit unsigned short field.

Problem:
Improper code is generated for statements involving unsigned short variables unless they are explicitly cast as unsigned shorts.

```
main()
{
    static unsigned short digit_index;
    static unsigned short digit[12];
    int a,b;
    if (digit[digit_index--]){
        a=4;
        b=4;}
    else{
        a=5;
        b=5;}
}

Improper code is generated for the comparison (ie the comparison is done on 16 bits (8 of which have been cleared) against #0FFFFH.
12/10/85: The problem also arises if you compare a constant against an unsigned short. For example if you declared:
#define constant ~0
unsigned short var;
and later compared these two, the compiler will zero out the upper byte of the variable var and then compare it to FFFFH. Thus, the condition is never met.
```

12/16/85: Another example of incorrect code being generated when a char variable is used in a test condition is as follows:

```
char a;
main()
```

- 8086/8 C -

```
{
  a = -1;
  if(a == -1)
    a = 'A';
}
```

Temporary solution:

Correct code is generated if the line in question is changed to the following although digit[] has already been declared unsigned short.

```
if ((unsigned short)digit[digit_index]--){
```

12/10/85: Declare the constant as a short. In other words:

```
#define constant OFFH.
```

12/16/85: If only 128 valid characters are required the variable can be declared as a short int.

Signed off 08/25/86 in release 803.01

```
Number: D200040634 Product: 8086/8 C 64818 02.00
```

Keywords: PASS 3

One-line description:

Pass 3 fails to detect relative jump address out-of-range.

Problem:

Pass 3 of the compilation process may fail to detect a relative jump which is out of range. In the test program submitted the relative jump is generated for an IF.THEN statement while the compiler option OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:

As a temporary work around disable the compiler option OPTIMIZE around those sections of code which are suspect.

Signed off 08/25/86 in release 803.01

```
Number: D200041194 Product: 8086/8 C 64818 02.00
```

One-line description:

Problem with integer pointer in conditional statement.

Problem:

In the following example, two loads are performed, but no other code is generated to check for zero value.

```
"C"
"processor name"
#define NULL 0
fct(parm)
int *parm;
{
  if (parm - NULL)
    parm = 10;
}
```

Signed off 08/25/86 in release 803.01

```
Number: D200047480 Product: 8086/8 C 64818 02.00
```

One-line description:

T00 MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 803.01

```
Number: D200049841 Product: 8086/8 C 64818 03.00
```

One-line description:

ES pushed instead of DS when POINTER SIZE = 32.

Problem:

The following code demonstrates a problem with the 8086 C compiler when \$POINTER_SIZE 32\$ is set:

```
"C"
"processor name"
$POINTER_SIZE 32$
static char aack[];
ppout()
{
  char *term;
  if (term == aack); <-- This statement generates incorrect code.
                          A PUSH ES instruction is generated
                          incorrectly.
}
```

Temporary solution:

Do not use \$POINTER_SIZE 32\$ in this manner if possible. Otherwise, create a ASM8086 file with \$ASM_FILE ON\$, correct the ASM8086 file to PUSH DS instead of PUSH ES, and assemble ASM8086.

Signed off 08/25/86 in release 803.01

Number: D200049874 Product: 8086/8 C 300 64818S004 03.00

One-line description:
ES pushed instead of DS when POINTER SIZE = 32.

Problem:
The following code demonstrates a problem with the 8086 C compiler when \$POINTER_SIZE 32\$ is set:

```
"C"
"processor name"
$POINTER_SIZE 32$
static char aack[];
ppout()
{
  char *term;
  if (term == aack);    <-- This statement generates incorrect code.
                        A PUSH ES instruction is generated
                        incorrectly.
}
```

Temporary solution:
Do not use \$POIINTER_SIZE 32\$ if possible. Otherwise, create a ASM8086 file with \$ASM_FILE ON\$, edit the ASM8086 file to PUSH DS instead of PUSH ES, adn assemble the ASM8086 file.

Signed off 08/25/86 in release 403.10

Number: D200051235 Product: 8086/8 C 300 64818S004 03.00

One-line description:
++ and -- operators evaluated with improper precedence.

Problem:
According to Kernighan and Ritchie, page 43, the following expressions are equivalent:
Example 1: array[index++] = 1;
Example 2: array[index] = 1;
 index++;
However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:
Separate the expression as shown in example 2.

Signed off 08/25/86 in release 403.10

Number: D200052258 Product: 8086/8 C 300 64818S004 00.00

Keywords: CODE GENERATOR

One-line description:
Incorrect opcode "MOV A,ACC" allowed by our assembler

- 8086/8 C -

Problem:
The instruction "MOV A,ACC" was assemble and emulated by our products; however, the Intel 8051 goes into the weeds at this instruction. At first glance the machine code in the assembler listing appears valid (MOV A,ACC ->0000 E5E0), but the bottom of page 8-35 in Intel's microcontroller handbook states: *MOV A,ACC is not a valid instruction.

Neither our manuals nor AMD's user manual mention this instruction.

Signed off 08/25/86 in release 403.10

Number: D200058933 Product: 8086/8 C 300 64818S004 03.00

One-line description:
Host compilers do not put absolute pats specifications in relocatables

Problem:
Host compilers do not specify the full path name in the relocatable file.

Signed off 08/25/86 in release 403.10

Number: D200048892 Product: 8086/8 C 300 64818S004 00.00

One-line description:
Linker output file should use alternate file extension.

Signed off 08/25/86 in release 403.10

- 8086/8 C -

Number: D200026666 Product: 8086/8 C 500 64818S001 01.10

One-line description:

No error when illegal assignment to a pointer is made.

Problem:

The native compiler on the 9000 flags an error for the following code, but the 8086/8 C compiler does not:

```
main()
{
  char *ptr;
  int i;
  char c;

  (ptr + i) +2 = c;      /*Should flag an error stating illegal
                        left hand side of expression */
}
```

Signed off 08/25/86 in release 103.20

Number: D200031302 Product: 8086/8 C 500 64818S001 02.00

One-line description:

++ and -- operators evaluated with improper precedence.

Problem:

According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

Example 1: array[index++] = 1;

Example 2: array[index] = 1;
index++;

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:

Separate the expression as shown in example 2.

Signed off 08/25/86 in release 103.20

Number: D200033118 Product: 8086/8 C 500 64818S001 02.00

One-line description:

Comparing character to zero in while loop generates incorrect code.

Problem:

If you compare a character variable to zero in a while loop, incorrect code is generated. The following code demonstrates the problem.

"C"

"6809"

```
proc()
{
  char timeout = 10;
```

- 8086/8 C -

```
    while(timeout--);      /* Code generated here causes infinite loop.
    }
```

The code generated for the while loop clears the A register and then compares the D register to -1. Therefore the condition is never met.

Temporary solution:

Declare the variable used in the test condition as an integer.

"C"

"6809"

```
proc()
{
  int timeout = 10;

  while (timeout--);
}
```

Signed off 08/25/86 in release 103.20

Number: D200035790 Product: 8086/8 C 500 64818S001 02.00

Keywords: CODE GENERATOR

One-line description:

16 bit comparison on a 8 bit unsigned short field.

Problem:

Improper code is generated for statements involving unsigned short variables unless they are explicitly cast as unsigned shorts.

```
main()
{
  static unsigned short digit_index;
  static unsigned short digit[12];
  int a,b;
  if (digit[digit_index--]){
    a=4;
    b=4;}
  else{
    a=5;
    b=5;}
}
```

Improper code is generated for the comparison (ie the comparison is done on 16 bits (8 of which have been cleared) against #0FFFFH.

12/10/85: The problem also arises if you compare a constant against an unsigned short. For example if you declared:

```
#define constant ~0
```

```
unsigned short var;
```

and later compared these two, the compiler will zero out the upper byte of the variable var and then compare it to FFFFH. Thus, the condition is never met.

12/16/85: Another example of incorrect code being generated when a char variable is used in a test condition is as follows:

- 8086/8 C -

```

char a;
main()
{
  a = -1;
  if(a == -1)
    a = 'A';
}

```

Temporary solution:

Correct code is generated if the line in question is changed to the following although digit[] has already been declared unsigned short.

```
if ((unsigned short)digit[digit_index]--){
```

12/10/85: Declare the constant as a short. In other words:

```
#define constant OFFH.
```

12/16/85: If only 128 valid characters are required the variable can be declared as a short int.

Signed off 08/25/86 in release 103.20

```
Number: D200037051 Product: 8086/8 C 500 64818S001 02.01
```

Keywords: PASS 3

One-line description:

Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:

Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect data being output to the list file. In selected cases, machine code will be incorrectly listed. For example, consider the following Pascal program.

```

$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;

```

```

VAR
  a, b : BOOLEAN;

```

```
PROCEDURE one;
```

```

  BEGIN
    a := b;
  END;

```

In the example listed above, the output file will denote machine code of the form FFFFC00001 for one of the generated assembly statements. The correct value should be C8000001. This problem is caused by an incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 103.20

- 8086/8 C -

```
Number: D200040642 Product: 8086/8 C 500 64818S001 02.01
```

Keywords: PASS 3

One-line description:

Pass 3 fails to detect relative jump address out-of-range.

Problem:

Pass 3 of the compilation process may fail to detect a relative jump which is out of range. In the test program submitted the relative jump is generated for an IF..THEN statement while the compiler option OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:

As a temporary work around disable the compiler option OPTIMIZE around those sections of code which are suspect.

Signed off 08/25/86 in release 103.20

```
Number: D200041202 Product: 8086/8 C 500 64818S001 02.01
```

One-line description:

Problem with integer pointer in conditional statement.

Problem:

In the following example, two loads are performed, but no other code is generated to check for zero value.

```

"C"
"processor name"
#define NULL 0
fct(parm)
int *parm;
{
  if (parm - NULL)
    parm = 10;
}

```

Signed off 08/25/86 in release 103.20

```
Number: D200045906 Product: 8086/8 C 500 64818S001 02.01
```

One-line description:

Title description is incorrect.

Signed off 08/25/86 in release 103.20

```
Number: D200046276 Product: 8086/8 C 500 64818S001 01.20
```

One-line description:

NULL CHARACTERS IN ASM SOURCE PRODUCED WITH \$ASM_FILE\$

Signed off 08/25/86 in release 103.20

- 8086/8 C -

Number: D200047498 Product: 8086/8 C 500 64818S001 02.01

One-line description:
TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 103.20

Number: D200049635 Product: 8086/8 C 500 64818S001 00.00

One-line description:
NO CROSS REFERENCE TABLE IS GENERATED

Problem:
"C" COMPILERS DO NOT GENERATE A CROSS REFERENCE TABLE ON THE
VAX.

Temporary solution:
NONE KNOWN AT PRESENT

Signed off 04/18/86 in release 103.20

Number: D200049858 Product: 8086/8 C 500 64818S001 03.10

One-line description:
ES pushed instead of DS when POINTER SIZE = 32.

Problem:
The following code demonstrates a problem with the 8086 C compiler
when \$POINTER_SIZE 32\$ is set:

```
"C"
"processor name"
$POINTER_SIZE 32$
static char aack[];
ppout()
{
  char *term;
  if (term == aack);    <-- This statement generates incorrect code.
                        A PUSH ES instruction is generated
                        incorrectly.
}
```

Signed off 08/25/86 in release 103.20

Number: D200058917 Product: 8086/8 C 500 64818S001 03.10

One-line description:
Host compilers do not put absolute pats specifications in relocatables

Problem:
Host compilers do not specify the full path name in the
relocatable file.

Signed off 08/25/86 in release 103.20

Number: D200048876 Product: 8086/8 C 500 64818S001 00.00

One-line description:
Linker output file should use alternate file extension.

Signed off 08/25/86 in release 103.20

Number: D200026674 Product: 8086/8 C VAX 64818S003 01.10

One-line description:

No error when illegal assignment to a pointer is made.

Problem:

The native compiler on the 9000 flags an error for the following code, but the 8086/8 C compiler does not:

```
main()
{
  char *ptr;
  int i;
  char c;

  (ptr + i) + 2 = c;    /*Should flag an error stating illegal
                        left hand side of expression */
}
```

Signed off 08/25/86 in release 303.40

Number: D200031310 Product: 8086/8 C VAX 64818S003 02.00

One-line description:

++ and -- operators evaluated with improper precedence.

Problem:

According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

Example 1: array[index++] = 1;

Example 2: array[index] = 1;
index++;

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:

Separate the expression as shown in example 2.

Signed off 08/25/86 in release 303.40

Number: D200033126 Product: 8086/8 C VAX 64818S003 02.00

One-line description:

Comparing character to zero in while loop generates incorrect code.

Problem:

If you compare a character variable to zero in a while loop, incorrect code is generated. The following code demonstrates the problem.

```
"C"
"6809"

proc()
{
  char timeout = 10;
```

- 8086/8 C -

```
    while(timeout--);    /* Code generated here causes infinite loop.
    }
```

The code generated for the while loop clears the A register and then compares the D register to -1. Therefore the condition is never met.

Temporary solution:

Declare the variable used in the test condition as an integer.

```
"C"
"6809"

proc()
{
  int timeout = 10;

  while (timeout--);
}
```

Signed off 08/25/86 in release 303.40

Number: D200035808 Product: 8086/8 C VAX 64818S003 02.00

Keywords: CODE GENERATOR

One-line description:

16 bit comparison on a 8 bit unsigned short field.

Problem:

Improper code is generated for statements involving unsigned short variables unless they are explicitly cast as unsigned shorts.

```
main()
{
  static unsigned short digit_index;
  static unsigned short digit[12];
  int a,b;
  if (digit[digit_index--]){
    a=4;
    b=4;}
  else{
    a=5;
    b=5;}
}
```

Improper code is generated for the comparison (ie the comparison is done on 16 bits (8 of which have been cleared) against #0FFFFH.

12/10/85: The problem also arises if you compare a constant against an unsigned short. For example if you declared:

```
#define constant ~0
unsigned short var;
and later compared these two the compiler will zero out the upper byte of the variable var and then compare it to FFFFH. Thus, the condition is never met.
```

12/16/85: Another example of incorrect code being generated when a character variable is used in a test condition is as follows:

- 8086/8 C -

```
char a;
main()
{
  a = -1;
  if(a == -1)
    a = 'A';
}
```

Temporary solution:

Correct code is generated if the line in question is changed to the following although digit[] has already been declared unsigned short.

```
if ((unsigned short)digit[digit_index--]){
```

12/10/85: Declare the constant as a short. In other words:

```
#define constant OFFH.
```

12/16/85: If only 128 valid characters are required the variable can be declared as a short integer.

Signed off 08/25/86 in release 303.40

Number: D200037069 Product: 8086/8 C VAX 64818S003 02.00

Keywords: PASS 3

One-line description:

Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:

Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect data being output to the list file. In selected cases, machine code will be incorrectly listed. For example, consider the following Pascal program.

```
$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;
```

```
VAR
  a, b : BOOLEAN;
```

```
PROCEDURE one;
```

```
BEGIN
  a := b;
END;
```

In the example listed above, the output file will denote machine code of the form FFFFC00001 for one of the generated assembly statements. The correct value should be C8000001. This problem is caused by an incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 303.40

- 8086/8 C -

Number: D200040659 Product: 8086/8 C VAX 64818S003 02.00

Keywords: PASS 3

One-line description:

Pass 3 fails to detect relative jump address out-of-range.

Problem:

Pass 3 of the compilation process may fail to detect a relative jump which is out of range. In the test program submitted the relative jump is generated for an IF..THEN statement while the compiler option OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:

As a temporary work around disable the compiler option OPTIMIZE around those sections of code which are suspect.

Signed off 08/25/86 in release 303.40

Number: D200041210 Product: 8086/8 C VAX 64818S003 02.00

One-line description:

Problem with integer pointer in conditional statement.

Problem:

In the following example, two loads are performed, but no other code is generated to check for zero value.

```
"C"
"processor name"
#define NULL 0
fct(parm)
int *parm;
{
  if (parm - NULL)
    parm = 10;
}
```

Signed off 08/25/86 in release 303.40

Number: D200045914 Product: 8086/8 C VAX 64818S003 02.00

One-line description:

Title description is incorrect.

Signed off 08/25/86 in release 303.40

Number: D200046607 Product: 8086/8 C VAX 64818S003 02.00

One-line description:

NULL CHARACTERS IN ASM SOURCE PRODUCED WITH \$ASM_FILE\$

Signed off 08/25/86 in release 303.40

- 8086/8 C -

Number: D200047506 Product: 8086/8 C VAX 64818S003 02.00

One-line description:
TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 303.40

Number: D200049866 Product: 8086/8 C VAX 64818S003 03.10

One-line description:
ES pushed instead of DS when POINTER SIZE = 32.

Problem:
The following code demonstrates a problem with the 8086 C compiler when \$POINTER_SIZE 32\$ is set:

```
"C"
"processor name"
$POINTER_SIZE 32$
static char aack[];
ppout()
{
  char *term;
  if (term == aack);    <-- This statement generates incorrect code.
                       A PUSH ES instruction is generated
                       incorrectly.
}
```

Temporary solution:
No known temporary solution.

Signed off 08/25/86 in release 303.40

Number: D200055129 Product: 8086/8 C VAX 64818S003 03.10

One-line description:
Compilation on the VAX using batch mode generates incorrect listing file

Problem:
The test files can be found on the VAX750 under user\$disk:[robin.hughes.rgalo.test]. The following test files were used:

1. MTINHST_C. - File which contains one error- a missing '}' on line 70
2. TMTINHST_C. - Error-free version of MTINHST_C.
3. MTOPNDF_C. - File which contains one error - missing declaration for integer 'j'
4. MTOPNDF_C. - Error-free version of MTOPNDF_C.

One logical name must be defined as follows to access the include files referenced by the test programs:

```
$define BSLN user$disk:[robin.hughes.wsbsln.baseline]
```

When the four files were compiled interactively, the two error-free versions generated correct listings. The first file (MTINHST_C.) generated an incomplete and incorrect listing file. The listing showed the include files inserted first, followed by "C", "8086" and a few other lines of the program. The output displayed on the screen looked like:

```
In pass1.
  70 else
      ^25
  136
      ^408
In C Nocode.
comp: C Nocode cannot recover from errors.
```

When the third file (MTOPNDF_C.) was compiled, the listing appeared fine except for the insertion a some strange control charaters.

These last two files were compiled in batch mode (file: user\$disk:[robin.hughes.rgalo.test]hughes.com). The first file (MTINHST_C.) generated a complete but incorrect listing. Although two errors were found (25 & 408) the line at the bottom stated that errors = 0. The include file expansion preceeded the "C" and "8086" in the listing, and lines like, #include filename, were still in the file. The error message was at line 72 of the listing instead of line 2472 were the '}' was actual missing. Finally the last 100 lines had useless numbers in the left margin.

When the third file (MTOPNDF_C.) was compiled, an incomplete listing was generated with the include file expansions listed first.

All of these tests were done on the VAX750 with the /e/v/o options.

This problem also occurs on the 68000.

Temporary solution:
No temporary solution available

Signed off 08/25/86 in release 303.40

Number: D200058925 Product: 8086/8 C VAX 64818S003 03.10

One-line description:
Host compilers do not put absolute pats specifications in relocatables

Problem:
Host compilers do not specify the full path name in the relocatable file.

Signed off 08/25/86 in release 303.40

Number: D200048884 Product: 8086/8 C VAX 64818S003 00.00

One-line description:
Linker output file should use alternate file extension.

Signed off 08/25/86 in release 303.40

Number: 5000118828 Product: 8086/8 PASCAL 64814 02.00

One-line description:

Param of WRITELN not separated by , 's cause compiler to abort.

Problem:

Compiler aborts without creating a listing file when WRITELN parameters are not delimited by commas. The following example causes the compiler to abort and a "301:no case provided for this value" message appears on the status line. Line numbers do not appear on the status line before the compiler aborts (that normally give a hint to the location of the problem).

```
"8086"
$EXTENSIONS ON$
```

```
PROGRAM TEST;
VAR FSORTIE : TEXT;
BEGIN
WRITELN(FSORTIE,'MESSAGE'XXX');
END.
```

Note: The two parameters that are not separated by commas do not have to be strings. They could be variable names.

The VAX and 9000 generate the following errors for this line:
0,4,126,139

Temporary solution:

The only temporary solution is to manually check the source file for WRITELN parameters not delimited by commas.

Pisces+:

If a Pisces+ environment is being used the file could be compiled on the host computer.

Signed off 08/25/86 in release 403.01

Number: D200015230 Product: 8086/8 PASCAL 64814 01.10

One-line description:

Only two bytes of a three byte array are passed correctly as parameters.

Problem:

Problem when passing parameters....3 byte array of type char. Only two of the parameters are passed correctly, the third parameter is passed as zero.

Temporary solution:

Problem can be resolved by using an even array.

Signed off 08/25/86 in release 403.01

Number: D200036780 Product: 8086/8 PASCAL 64814 02.01

Keywords: INCLUDE

One-line description:

Nested INCLUDE files 3 or more deep cause 64000 to "hang" in pass 3.

Problem:

Nested INCLUDE files 3 or more deep cause 64000 to hang in pass 3.

Temporary solution:

None at this time.

Signed off 08/25/86 in release 403.01

Number: D200037234 Product: 8086/8 PASCAL 64814 02.01

One-line description:

Bad "machine" code generated for LEA assembly instruction.

Temporary solution:

Use the compiler option \$ASM_FILE\$ to obtain an assembly file. Use this file as input to the assembler. The assembler generates correct code.

Signed off 08/25/86 in release 403.01

Number: D200038950 Product: 8086/8 PASCAL 64814 02.01

One-line description:

Incorrect machine code generated for LEA ... instruction.

Signed off 08/25/86 in release 403.01

Number: D200046631 Product: 8086/8 PASCAL 64814 02.01

One-line description:

Error 1102: register needed but not available.

Problem:

Signed off 08/25/86 in release 403.01

Number: D200047399 Product: 8086/8 PASCAL 64814 02.01

One-line description:

T00 MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 403.01

Number: D200052522 Product: 8086/8 PASCAL 64814 03.00

One-line description:

Missing semicolon causes compiler to hang in Pass 1.

Problem:

The following code causes the 64000 to hang in pass 1. An error

is generated on the hosts stating that parsing has stopped at a particular line number.

"processor name"

```
PROGRAM MAIN;
TYPE
STRUCTURED= RECORD
    INT1:INTEGER;
    INT2:INTEGER;
END;
```

```
PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
I:=P1      <--This missing semicolon causes the problem
I:=P1.2;
I:=P2;
END;
```

```
BEGIN
END.
```

Temporary solution:

If the compiler hangs, look for a statement without a semicolon. On the 64000, the status line will show which line of code it stopped on. On the hosts, the error message generated indicates which line of code parsing stopped on.

Signed off 08/25/86 in release 403.01

Number: D200053181 Product: 8086/8 PASCAL 64814 03.00

Keywords: CODE GENERATOR

One-line description:

Width option causes 64000 to enter PV during compilation

Problem:

THE FOLLOWING PROGRAM CAUSES THE 64000 TO JUMP INTO PERFORMANCE VERIFICATION WHEN COMPILED.

```
"80188"
$EXTENSIONS ON$
$ WIDTH 70$
PROGRAM TEST;
$GLOBPROC ON$
PROCEDURE EXAMPLE;
```

```
CONST
VAR1 = 2; VAR2 = 3; VAR3 = 4;
TYPE
SET_1 = (W,X,Y,Z); SET_2 = (O,R,Q,S);
SET1 = SET OF SET_1; SET2 = SET OF SET_2;
REC1 = RECORD
    DESC : SET1;
END;
```

VAR

```
A : INTEGER; P : UNSIGNED 8;
ARRAY1 : ARRAY [1..4] OF ARRAY [1..5] OF REC1;
ARRAY2 : ARRAY [6] OF SET2;
```

```
BEGIN
P := 10;
CASE (10 + A) OF
11: BEGIN
    IF (X IN ARRAY1[VAR1,VAR2].DESC) AND
        NOT (Q IN ARRAY2[VAR3]) THEN {THEN ends in col 70}
        P := P + 1;
    IF NOT (X IN ARRAY1[VAR1,VAR2].DESC) AND
        (Q IN ARRAY2[VAR3]) THEN {THEN ends in col 70}
        P := P + 2;
    END;
22: BEGIN
    IF (X IN ARRAY1[VAR1,VAR2].DESC) AND
        NOT (S IN ARRAY2[VAR3]) THEN {THEN ends in col 70}
        P := P + 1;
    IF NOT (X IN ARRAY1[VAR1,VAR2].DESC) AND
        (S IN ARRAY2[VAR3]) THEN {THEN ends in col 70}
        P := P + 2;
    END;
    OTHERWISE;
END;
END;
```

THE PROBLEM OCCURS ONLY WHEN THE WIDTH IS SET TO 70, 71, OR 72. ALL OTHER SETTINGS WORK. USING JUST ONE CASE CONSTANT INSTEAD OF TWO WILL NOT CREATE THE PROBLEM. IN ORDER TO CAUSE THE DEFECT THE SET MUST BE INDIRECTLY ACCESSED THROUGH A RECORD OR AN ARRAY. ALSO THE ARRAY INDEXES MUST BE VARIABLES OR CONSTANTS (I.E. ARRAY1[2,3].DESC WILL NOT JUMP INTO PV).

TEMPORARY SOLUTION:

CHANGE THE WIDTH COMPILER OPTION TO LONGER THAN THE LONGEST SOURCE LINE.

Signed off 08/25/86 in release 403.01

Number: D200053728 Product: 8086/8 PASCAL 64814 03.00

One-line description:

Register needed but not available

Problem:

An example of this problem can be found on the 9000 hpldsb under /users/robin/pass2.s. The 1102 errors do not occur if you remove all the unnecessary variables that are defined. The customer uses include files for all his declarations.

Temporary solution:

No known temporary solution.

Signed off 08/25/86 in release 403.01

Number: D200053736 Product: 8086/8 PASCAL 64814 03.00

Keywords: CODE GENERATOR

One-line description:
Variable addresses calculated incorrectly

Problem:
THE PROGRAM IN THE SUMMITER TEXT SECTION DOES NOT GENERATE THE
CORRECT ADDRESSES FOR "OPR_SLOT_SELECTED" AND "OVERRIDE_CHAN_SLOT"
WHEN COMPILED.

A COPY OF THIS PROGRAM CAN BE FFOUND ON !HPLSDSB UNDER /USERS/ROBIN/
AWABUG2.S

Temporary solution:
No known temporary solution.

Signed off 08/25/86 in release 403.01

Number: D200052555 Product: 8086/8 PASCAL 300 64814S004 03.00

One-line description:
Missing semicolon causes compiler to hang in Pass 1.

Problem:
The following code causes the 64000 to hang in pass 1. An error
is generated on the hosts stating that parsing has stopped at
a particular line number.

```
"processor name"
PROGRAM MAIN;
TYPE
STRUCTURED= RECORD
    INT1:INTEGER;
    INT2:INTEGER;
END;
```

```
PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
I:=P1      <--This missing semicolon causes the problem
I:=P1.2;
I:=P2;
END;
```

```
BEGIN
END.
```

Temporary solution:
If the compiler hangs, look for a statement without a semicolon.
On the 64000, the status line will show which line of code it
stopped on. On the hosts, the error message generated indicates
which line of code parsing stopped on.

Signed off 08/25/86 in release 403.10

Number: D200058768 Product: 8086/8 PASCAL 300 64814S004 03.00

Keywords: PREPROCESSOR

One-line description:
Preprocessor reports errors when symbols hp64000, vms or hpux w #if

Signed off 08/25/86 in release 403.10

Number: D200059196 Product: 8086/8 PASCAL 300 64814S004 03.00

One-line description:
Host compilers do not put absolute pats specifications in relocatables

Problem:
Host compilers do not specify the full path name in the
relocatable file.

Signed off 08/25/86 in release 403.10

SRB detail reports as of 08/25/86

Page: 213

Number: D200048801 Product: 8086/8 PASCAL 300 64814S004 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 403.10

SRB detail reports as of 08/25/86

Page: 214

Number: D200027649 Product: 8086/8 PASCAL 500 64814S001 02.00

One-line description:

No form feed between the expanded listing and the cross reference table.

Problem:

During compilation, with XREF option on, the compiler does not provide a form feed (FF) in the listing file. The XREF starts on the same page as the end of the listing. Also, the page number says 535 when it should be page 2.

Temporary solution:

After compiling with the xref option, edit the expanded listing file and insert a "control L" before the beginning of the cross reference listing.

Signed off 08/25/86 in release 103.10

Number: D200036871 Product: 8086/8 PASCAL 500 64814S001 02.00

Keywords: PASS 3

One-line description:

Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:

Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect data being output to the list file. In selected cases, machine code will be incorrectly listed. For example, consider the following Pascal program.

```
$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;

  VAR
    a, b : BOOLEAN;

  PROCEDURE one;

    BEGIN
      a := b;
    END;
```

In the example listed above, the output file will denote machine code of the form FFFFC00001 for one of the generated assembly statements. The correct value should be C8000001. This problem is caused by an incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 103.10

Number: D200037291 Product: 8086/8 PASCAL 500 64814S001 02.00

One-line description:

Bad "machine" code generated for LEA assembly instruction.

Signed off 08/25/86 in release 103.10

Number: D200046318 Product: 8086/8 PASCAL 500 64814S001 01.30

One-line description:

NULL CHARACTERS IN ASM SOURCE PRODUCED WITH \$ASM_FILE\$

Signed off 08/25/86 in release 103.10

Number: D200046748 Product: 8086/8 PASCAL 500 64814S001 02.00

One-line description:

Error 1102: register needed but not available.

Signed off 08/25/86 in release 103.10

Number: D200047407 Product: 8086/8 PASCAL 500 64814S001 02.00

One-line description:

TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 103.10

Number: D200052530 Product: 8086/8 PASCAL 500 64814S001 03.00

One-line description:

Missing semicolon causes compiler to hang in Pass 1.

Problem:

The following code causes the 64000 to hang in pass 1. An error is generated on the hosts stating that parsing has stopped at a particular line number.

```
"processor name"
PROGRAM MAIN;
TYPE
STRUCTURED= RECORD
    INT1:INTEGER;
    INT2:INTEGER;
END;
```

```
PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
I:=P1      <--This missing semicolon causes the problem
I:=P1.2;
I:=P2;
END;
```

```
BEGIN
END.
```

Temporary solution:

If the compiler hangs, look for a statement without a semicolon. On the 64000, the status line will show which line of code it stopped on. On the hosts, the error message generated indicates which line of code parsing stopped on.

Signed off 08/25/86 in release 103.10

Number: D200058743 Product: 8086/8 PASCAL 500 64814S001 03.00

Keywords: PREPROCESSOR

One-line description:

Preprocessor reports errors when symbols hp64000, vms or hpux w #if

Signed off 08/25/86 in release 103.10

Number: D200059170 Product: 8086/8 PASCAL 500 64814S001 03.00

One-line description:

Host compilers do not put absolute paths specifications in relocatables

Problem:

Host compilers do not specify the full path name in the relocatable file.

Signed off 08/25/86 in release 103.10

Number: D200048785 Product: 8086/8 PASCAL 500 64814S001 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 103.10

Number: D200027656 Product: 8086/8 PASCAL VAX 64814S003 02.00

One-line description:

No form feed between the expanded listing and the cross reference table.

Problem:

During compilation, with XREF option on, the compiler does not provide a form feed (FF) in the listing file. The XREF starts on the same page as the end of the listing. Also, the page number says 535 when it should be page 2.

Temporary solution:

After compiling with the xref option, edit the expanded listing file and insert a "control L" before the beginning of the cross reference listing.

Signed off 08/25/86 in release 303.20

Number: D200037002 Product: 8086/8 PASCAL VAX 64814S003 02.00

Keywords: PASS 3

One-line description:

Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:

Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect data being output to the list file. In selected cases, machine code will be incorrectly listed. For example, consider the following Pascal program.

```
$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;
```

```
VAR
  a, b : BOOLEAN;
```

```
PROCEDURE one;
```

```
BEGIN
  a := b;
END;
```

In the example listed above, the output file will denote machine code of the form FFFFC00001 for one of the generated assembly statements. The correct value should be C8000001. This problem is caused by an incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 303.20

Number: D200037309 Product: 8086/8 PASCAL VAX 64814S003 02.00

One-line description:

Bad "machine" code generated for LEA assembly instruction.

Signed off 08/25/86 in release 303.20

Number: D200046615 Product: 8086/8 PASCAL VAX 64814S003 02.00

One-line description:

NULL CHARACTERS IN ASM SOURCE PRODUCED WITH \$ASM_FILE\$

Signed off 08/25/86 in release 303.20

Number: D200046755 Product: 8086/8 PASCAL VAX 64814S003 02.00

One-line description:

Error 1102: register needed but not available.

Signed off 08/25/86 in release 303.20

Number: D200047415 Product: 8086/8 PASCAL VAX 64814S003 02.00

One-line description:

TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 303.20

Number: D200052548 Product: 8086/8 PASCAL VAX 64814S003 03.00

One-line description:

Missing semicolon causes compiler to hang in Pass 1.

Problem:

The following code causes the 64000 to hang in pass 1. An error is generated on the hosts stating that parsing has stopped at a particular line number.

```
"processor name"
PROGRAM MAIN;
TYPE
STRUCTURED= RECORD
  INT1:INTEGER;
  INT2:INTEGER;
END;
```

```
PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
  I:=P1      <--This missing semicolon causes the problem
  I:=P1.2;
  I:=P2;
END;
```

```
BEGIN
END.
```

Temporary solution:

If the compiler hangs, look for a statement without a semicolon. On the 64000, the status line will show which line of code it stopped on. On the hosts, the error message generated indicates which line of code parsing stopped on.

Signed off 08/25/86 in release 303.20

Number: D200058750 Product: 8086/8 PASCAL VAX 64814S003 03.00

Keywords: PREPROCESSOR

One-line description:

Preprocessor reports errors when symbols hp64000, vms or hpux w #if

Signed off 08/25/86 in release 303.20

Number: D200059188 Product: 8086/8 PASCAL VAX 64814S003 03.00

One-line description:

Host compilers do not put absolute pats specifications in relocatables

Problem:

Host compilers do not specify the full path name in the relocatable file.

Signed off 08/25/86 in release 303.20

Number: D200048793 Product: 8086/8 PASCAL VAX 64814S003 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 303.20

Number: D200060301 Product: F9450 EMULATION 64286 01.02

One-line description:

Intermittent PV failures occur on test 8 (IO Cycles)

Temporary solution:

Ignore failures on test 8 if they occur at a rate of approximately 2 in 100.

Signed off 08/25/86 in release 601.03

Number: D200043570 Product: OP_SYS DEC-VAX / VMS 64882 01.20

Keywords: TRANSFER

One-line description:

The wrong protection can be left on HSL0.DAT when MAPBUS completes.

Problem:

When CSIB initially runs, it spawns a sub-process (usually named SYSTEM_1) to run a MAPBUS on the 64000 cluster. When MAPBUS completes, a file called HP\$64000:HSL0.DAT is created with file protection that denies the world READ-ACCESS.

The error message that a user will receive is:

```
transfer: high speed link 0 not running
ERROR: requested high speed link is not in operation
%NONAME-E-NOMSG, Message number 0000002
```

Temporary solution:

The protection on this file must be set with the following command:
\$ SET PROTECTION=(SYSTEM:REWD,OWNER:REWD,GROUP:R,WORLD:R) HSL0.DAT

Signed off 08/25/86 in release 201.70

Number: D200043935 Product: OP_SYS DEC-VAX / VMS 64882 01.20

Keywords: HIGH SPEED LINK TRANSFER

One-line description:

TRANSFER/H/A/T from anACL controled directory does not work.

Problem:

Given a directory that denies access to a user by its file protection, but who is allowed access via an ACL, even though the user may read and copy the file via a DCL command, TRANSFER/H is not able to access the file although TRANSFER/R can.

Temporary solution:

Copy the files to be transfered out of the ACL controlled directory and then TRANSFER the copied file.
A second solution would be to change the file protection to allow access per normal file access protections.

Signed off 08/25/86 in release 201.70

Number: D200045054 Product: OP_SYS DEC-VAX / VMS 64882 01.20

Keywords: HIGH SPEED LINK

One-line description:

File list transfers may not work under certain conditions.

Problem:

Given the following transfer, "TRANSFER/HSL/LIST/ASSERTIVE/TO",

- OP_SYS DEC-VAX / VMS -

if any of the files in the list or the directory containing the files does not allow world read access, the transfer will abort at the point where access is denied and will display a status dump.

Temporary solution:

Make sure the directory containing the files and the files them selves allow (W:R) access.

Signed off 08/25/86 in release 201.70

Number: D200046110 Product: OP_SYS DEC-VAX / VMS 64882 01.20

One-line description:

Mapbus output is "hardwired" to the system console.

Signed off 08/25/86 in release 201.70

Number: D200046144 Product: OP_SYS DEC-VAX / VMS 64882 01.20

One-line description:

Debug transfers will not work when '.PAS' file extensions are used.

Signed off 08/25/86 in release 201.70

Number: D200047969 Product: OP_SYS DEC-VAX / VMS 64882 01.20

Keywords: HIGH SPEED LINK

One-line description:

The HPIB configuration on the OPA0: doesn't contain line-feeds.

Problem:

When mapbus completes when CSIB is started, all the lines of the HPIB configuration printed on the OPA0: overwrite themselves. It appears that that data to the OPA0: doesn't contain line-feeds.

When a mapbus is manually run from the OPA0:, the HPIB configuration is printed correctly.

Temporary solution:

None at this time.

Signed off 08/25/86 in release 201.70

Number: D200047985 Product: OP_SYS DEC-VAX / VMS 64882 01.20

Keywords: HIGH SPEED LINK

One-line description:

A CSIB with a pending MAPBUS, changes priority from 12 to 14 and back.

Signed off 08/25/86 in release 201.70

- OP_SYS DEC-VAX / VMS -

Number: D200048025 Product: OP_SYS DEC-VAX / VMS 64882 01.20

Keywords: HIGH SPEED LINK

One-line description:
High speed link transfer does not work from passworded userids.

Problem:
High speed link transfers don't work to/from pass-worded 64000 userids.

Temporary solution:
None at this time.

Signed off 08/25/86 in release 201.70

Number: D200053819 Product: OP_SYS DEC-VAX / VMS 64882 01.60

Keywords: TRANSFER

One-line description:
Certain length filename.extension's will not transfer.

Problem:
If the sum of the lengths of the file name and the extension exceed 17 characters, then the length of the extension cannot exceed 8 characters for the file to transfer.

Signed off 08/25/86 in release 201.70

Number: D200053892 Product: OP_SYS DEC-VAX / VMS 64882 01.60

One-line description:
Foreground signal can kill a background batch remote control job.

Problem:
A 'CNTL C', entered in foregorund work can kill a background remote control job which was started from the same terminal session. This was an unintentional RE-INTRODUCTION of the defect that was fixed and documented by SR-NO D200020263.

Temporary solution:
Add a 10 second sleep to the beginning of any remote control batch job. After submitting thi batch job, log off during that first 10 seconds. Any foreground signals generated in the future will then belong to another terminal session and have no effect on the batch job.

Signed off 08/25/86 in release 201.70

Number: D200053900 Product: OP_SYS DEC-VAX / VMS 64882 01.60

One-line description:
Hp 64000 exit message is not outputted for exits when needed

Problem:
Remote will appear not to be able to exit from the main menu if the HP 64000 was bit left in monitor mode. The message prompting the

- OP_SYS DEC-VAX / VMS -

user to enter a "yes" to reboot the HP 64000 was not outputted.

Temporary solution:
The user may enter the exit command followed by a "yes" when exiting while the HP 64000 is not in monitor mode, or the user may return the HP 64000 to monitor mode before exiting.

Signed off 08/25/86 in release 201.70

Number: D200053884 Product: OP_SYS DEC-VAX / VMS 64882 01.60

One-line description:
REMOTE CONTROL HP6400 LOCKING MECHANISM WAS MADE MORE RELIABLE

Signed off 08/25/86 in release 201.70

- OP_SYS DEC-VAX / VMS -

SRB detail reports as of 08/25/86

Page: 225

Number: D200043588 Product: OP_SYS HP-UX / 500 64880 01.20

One-line description:

High Speed Link transfer can remove files from protected directories.

Signed off 04/18/86 in release 001.60

Number: D200054320 Product: OP_SYS HP-UX / 500 64880 01.50

One-line description:

Foreground signal can kill a background batch remote control job.

Problem:

A 'CNTRL C', entered in foreground work can kill a background remote control job which was started from the same terminal session. This was an unintentional RE-INTRODUCTION of the defect that was fixed and documented by SR-NO D200020263.

Temporary solution:

Add a 10_second sleep to the beginning of any remote control batch job. After submitting this batch job, log off during that first 10 seconds. Any foreground signals generated in the future will then belong to another terminal session and have no effect on the batch job.

Signed off 08/25/86 in release 001.60

Number: D200054338 Product: OP_SYS HP-UX / 500 64880 01.50

One-line description:

HP 64000 exit message is not outputted for exits when needed

Problem:

Remote will appear not to be able to exit from the main menu if the HP 64000 was bit left in monitor mode. The message prompting the user to enter a "yes" to reboot the HP 64000 was not outputted.

Temporary solution:

The user may enter the exit command followed by a "yes" when exiting while the HP 64000 is not in monitor mode, or the user may return the HP 64000 to monitor mode before exiting.

Signed off 08/25/86 in release 001.60

Number: D200054346 Product: OP_SYS HP-UX / 500 64880 01.50

One-line description:

An escaped shell from the menu can return prematurely

Problem:

If the user escapes from the SHELL from the MENU while something is running on the HP 64000, which generates a status line update, the remote control program might return from the ESCAPED SHELL before the user exits the ESCAPED SHELL.

Terminal input will not appear normal and the user should exit

- OP_SYS HP-UX / 500 -

SRB detail reports as of 08/25/86

Page: 226

As Soon As Possible and KILL the ESCAPED SHELL - if it still exists.

Temporary solution:

DO NOT escape to a shell from the menu while something is running on the HP 64000 which might generate a STATUS LINE UPDATE.

Signed off 08/25/86 in release 001.60

Number: D200060269 Product: OP_SYS HP-UX / 500 64880 01.50

One-line description:

Problem with make utility.

Problem:

The hosted compiler doesn't return with the correct return status if the compilation has resulted in an error. The assembler returns with a non-zero result after an assembly with errors, so that "make" correctly stops the "making" process. After a compilation with errors, "make" continues with its actions, producing an incorrect absolute file.

Although the value returned by the compiler and assembler is not documented, the assembler always returns a useful value for "make" while the compiler always returns "0".

Signed off 08/25/86 in release 001.60

Number: D200060277 Product: OP_SYS HP-UX / 500 64880 01.50

One-line description:

Problems with the linker listing file and map.

Problem:

The map produced by the linker is not the same as the listing file on the 64000. It has no pages, the error information goes to the std-err. Using "pr" gives you paging, but no headers on each page. Using "2>&1" merges not only the error info, but also the unwanted copy of the "command.K" file in the output.

Signed off 08/25/86 in release 001.60

Number: 5000124040 Product: OP_SYS HP-UX / 500 64880 01.30

Keywords: LINKER

One-line description:

Linker is VERY "picky" about the use of file extensions.

Signed off 08/25/86 in release 001.60

Number: D200054312 Product: OP_SYS HP-UX / 500 64880 01.50

One-line description:

REMOTE CONTROL HP6400 LOCKING MECHANISM WAS MADE MORE RELIABLE

Signed off 08/25/86 in release 001.60

- OP_SYS HP-UX / 500 -

Number: D200042044 Product: USER DEF ASSEMB 500 64851S001 00.00

Keywords: LINKER

One-line description:
LINKER WILL NOT LINK FILENAMES STARTING WITH A NUMBER

Signed off 08/25/86 in release 101.50

Number: D200047019 Product: USER DEF ASSEMB 500 64851S001 01.20

One-line description:
Assembler should denote an error on non-absolute .SET expressions.

Signed off 08/25/86 in release 101.50

Number: D200048066 Product: USER DEF ASSEMB 500 64851S001 01.20

One-line description:
Assembler flags error on host but NOT on 64000.

Problem:
Submitted source file (for SA6801) does not correctly assemble on the host. The same file assembles without errors on the 64000.

Signed off 08/25/86 in release 101.50

Number: D200053496 Product: USER DEF ASSEMB 500 64851S001 01.30

One-line description:
Macro def. including .IF, within a IF causes assembler to stop code gen.

Problem:
If you have a ".IF" in a macro definition and that macro definition is within a conditional assembly "IF" then no code is generated. The program provided demonstrates the problem (see submitter text).

Temporary solution:
Pull the macro definition outside of the conditional if. No code will be generated for the definition.

"processor name"

ESSAI	EQU	0	
MAC	MACRO		
	.IF	ESSAI.EQ.0	FIN
LABEL	LD	A,0	
FIN	MEND		
	IF	ESSAI	
	MAC		
	ENDIF		
START	LD	A,3	

Signed off 08/25/86 in release 101.50

Number: D200055525 Product: USER DEF ASSEMB 500 64851S001 01.40

One-line description:
Comments not delimited by semi-colons appear in the assembler xref.

Problem:
If you do not delimit a comment with a semi-colon it will appear in the assembler xref.

"processor"

```
MOVE    D0,D1    COMMENT
```

COMMENT appears in the asm xref as an undefined symbol.

Temporary solution:
Delimit all comments with a semi-colon.

Signed off 08/25/86 in release 101.50

Number: D200059295 Product: USER DEF ASSEMB 500 64851S001 01.40

One-line description:
Host compilers do not put absolute pats specifications in relocatables

Problem:
Host compilers do not specify the full path name in the relocatable file.

Temporary solution:
No known temporary solution.

Signed off 08/25/86 in release 101.50

Number: D200059949 Product: USER DEF ASSEMB 500 64851S001 01.40

One-line description:
QUOTING CHARACTERS WITHIN STRINGS ARE ALL TRANSLATED TO "."

Problem:
When using quoting characters within strings ('",^) they are all translated to "." This was done to facilitate string comparisons but causes a problem when the string is to be part of the generated code.

Signed off 08/25/86 in release 101.50

Number: 1650006536 Product: USER DEF ASSEMB VAX 64851S003 01.20

Keywords: MACRO

One-line description:
string comparison does not function using conditional .if instr.

Problem:

Hosted Macro assembler on Vax does not expand macros properly. The problem is related with "String inequality comparison".

```
BEGIN      MACRO      &P1
             .IF &P1 .NE. "" FIN
             MOV     A,#0FH
             .NOP
             MEND

             BEGIN    MYLABEL
             BEGIN    ""
             END
```

The HP64100 allows checking for optional macro parameters by the above example. This method only works with the null ("") operand. If any other string is used for the operand, quotes must be placed either around the parameter at the macro call or around the &P1 in the .IF statement. However, the vax and 9000 do not produce the same code as the HP64100. Although the VAX/9000 does not generate an error message, the code generated is incorrect. For example, the call "BEGIN MYLABEL" in the above test program creates the following listing.

```
11          BEGIN    MYLABEL
+           .IF MYLABEL .NE. "" FIN
+           MOV     A,#0FH
12          etc.
```

Temporary Solution:

```
Replace     .IF &P1 .NE. "" FIN
with        .IF "&P1" .NE. """" FIN
```

Signed off 06/23/86 in release 301.50

Number: D200019877 Product: USER DEF ASSEMB VAX 64851S003 01.10

One-line description:
Code generated differs from code generated on HP 64000.

Signed off 06/23/86 in release 301.50

Number: D200047027 Product: USER DEF ASSEMB VAX 64851S003 01.20

One-line description:
Assembler should denote an error on non-absolute .SET expressions.

Signed off 08/25/86 in release 301.50

 Number: D200048413 Product: USER DEF ASSEMB VAX 64851S003 01.40

Keywords: MACRO

 One-line description:
 Conditional instr. .IF with rational oper. in Macro creates bad code

 Problem:
 The use of the conditional instruction, .IF, with rational operator (.EQ, .NE, .LT, .GT, .LE, .GE.) in a macro functions incorrectly. The following program demonstrates this problem:

```

BUG          MACRO          &VAR
              .IF &VAR .LE. 0 SUB&&&&
              NOP
              NOP
SUB&&&&       NOP
              NOP
              MEND

              BUG 3
              BUG -1
              BUG 0
              END
  
```

Passing a 3 appears to create correct code, but 0 causes a ML error. Passing -1 to the MACRO creates code which doesn't call the subroutine. This is incorrect since -1 is less than 0. This same problem occurred with all the rational operators on all processors. The problem was consistent on the 64000, VAX, and 9000.

Signed off 06/23/86 in release 301.50

 Number: D200053504 Product: USER DEF ASSEMB VAX 64851S003 01.40

 One-line description:
 Macro def. including .IF, within a IF causes assembler to stop code gen.

 Problem:
 If you have a ".IF" in a macro definition and that macro definition is within a conditional assembly "IF" then no code is generated. The program provided demonstrates the problem (see submitter text).

 Temporary solution:
 Pull the macro definition outside of the conditional if. No code will be generated for the definition.

"processor name"

```

ESSAI      EQU      0

MAC        MACRO
          .IF      ESSAI.EQ.0   FIN
LABEL     LD       A,0
FIN       MEND
  
```

- USER DEF ASSEMB -V

```

IF      ESSAI
MAC
ENDIF
  
```

START LD A,3

Signed off 06/23/86 in release 301.50

 Number: D200055533 Product: USER DEF ASSEMB VAX 64851S003 01.40

 One-line description:
 Comments not delimited by semi-colons appear in the assembler xref.

 Problem:
 If you do not delimit a comment with a semi-colon it will appear in the assembler xref.

"processor"

```

MOVE      D0,D1      COMMENT
  
```

COMMENT appears in the asm xref as an undefined symbol.

 Temporary solution:
 Delimit all comments with a semi-colon.

Signed off 08/25/86 in release 301.50

 Number: D200059303 Product: USER DEF ASSEMB VAX 64851S003 01.40

 One-line description:
 Host compilers do not put absolute pats specifications in relocatables

 Problem:
 Host compilers do not specify the full path name in the relocatable file.

 Temporary solution:
 No known temporary solution.

Signed off 08/25/86 in release 301.50

 Number: D200059410 Product: USER DEF ASSEMB VAX 64851S003 01.40

 One-line description:
 PROBLEMS WHEN USING "FDB" OR "FCB" WITH A STRING

```

Problem:
FDB  "STRING"
FCB  "STRING"
  
```

THESE COMMANDS GENERATE INCORRECT CODE

Signed off 08/25/86 in release 301.50

- USER DEF ASSEMB -V

 Number: D200059956 Product: USER DEF ASSEMB VAX 64851S003 01.40

One-line description:
 QUOTING CHARACTERS WITHIN STRINGS ARE ALL TRANSLATED TO "."

Problem:
 When using quoting characters within strings ('',",^) they are all translated to "." This was done to facilitate string comparisons but causes a problem when the string is to be part of the generated code.

Signed off 08/25/86 in release 301.50

 Number: D200049395 Product: USER DEF ASSEMB VAX 64851S003 00.00

One-line description:
 Linker output file should use alternate file extension.

Signed off 06/23/86 in release 301.50

 Number: 5000132720 Product: Z80 ASSEMB 64842 01.11

One-line description:
 Z80 assembler allowing illegal instructions.

Problem:
 The following instructions are illegal, but no assembler errors are generated:

"Z80"

```
ADD IX,HL
ADD HL,IX
```

Temporary solution:
 Do not use these instructions.

Signed off 08/25/86 in release 201.12

 Number: D200033407 Product: Z80 ASSEMB 64842 00.01

One-line description:
 Legal range error is flagged when .NT. logical operator is used.

Problem:
 If you use the .NT. logical operator on an immediate of FFH a Legal range error is flagged. Any value below 0FFH will not flag the error. Also, in all cases the correct op code is generated.
 "Z80"

```
AND .NT.0FFH ;LEGAL RANGE ERROR FLAGGED
AND .NT.0FEH ;NO ERROR FLAGGED
```

Signed off 08/25/86 in release 201.12

 Number: D200036509 Product: Z80 ASSEMB 64842 00.01

One-line description:
 No error flagged when illegal 16 bit addition is preformed.

Problem:
 No error message is generated for 16 bit add instructions which use unavailable registers. Object code is generated for an allowed register pair.
 "Z80"

```
DD29 ADD IX,IY ;This is illegal, yet object code is
;generated.
FD29 ADD IY,HL ;Another example
```

Signed off 08/25/86 in release 201.12

 Number: D200046821 Product: Z80 ASSEMB 64842 00.01

One-line description:
 Assembler should denote an error on non-absolute .SET expressions.

Signed off 08/25/86 in release 201.12

Number: D200048249 Product: Z80 ASSEMB 300 64842S004 01.00

Keywords: MACRO

One-line description:
Conditional instr. .IF with rational oper. in Macro creates bad code

Problem:
The use of the conditional instruction, .IF, with rational operator (.EQ.,.NE.,.LT.,.GT.,.LE.,.GE.) in a macro functions incorrectly. The following program demonstrates this problem:

```

BUG          MACRO          &VAR
              .IF &VAR .LE. 0 SUB&&&
              NOP
              NOP
SUB&&&&      NOP
              NOP
              MEND

              BUG 3
              BUG -1
              BUG 0
              END

```

Passing a 3 appears to create correct code, but 0 causes a ML error. Passing -1 to the MACRO creates code which doesn't call the subroutine. This is incorrect since -1 is less than 0. This same problem occurred with all the rational operators on all processors. The problem was consistent on the 64000, VAX, and 9000.

Signed off 08/25/86 in release 401.10

Number: D200053215 Product: Z80 ASSEMB 300 64842S004 01.00

One-line description:
Z80 assembler allowing illegal instructions.

Problem:
The following instructions are illegal, but no assembler errors are generated:

```

"Z80"
  ADD  IX,HL
  ADD  HL,IX

```

Temporary solution:
Do not use these illegal instructions.

Signed off 08/25/86 in release 401.10

Number: D200053330 Product: Z80 ASSEMB 300 64842S004 01.00

One-line description:
Macro def. including .IF, within a IF causes assembler to stop code gen.

Problem:

If you have a ".IF" in a macro definition and that macro definition is within a conditional assembly "IF" then no code is generated. The program provided demonstrates the problem (see submitter text).

Temporary solution:

Pull the macro definition outside of the conditional if. No code will be generated for the definition.

"Z80"

```

ESSAI      EQU      0

MAC        MACRO
           .IF      ESSAI.EQ.0   FIN
LABEL     LD        A,0
FIN       MEND

           IF      ESSAI
           MAC
           ENDIF

START     LD        A,3

```

Signed off 08/25/86 in release 401.10

Number: D200049221 Product: Z80 ASSEMB 300 64842S004 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 401.10

- Z80 ASSEMB -

Number: D200046839 Product: Z80 ASSEMB 500 64842S001 01.20

One-line description:

Assembler should denote an error on non-absolute .SET expressions.

Signed off 08/25/86 in release 101.40

Number: D200048223 Product: Z80 ASSEMB 500 64842S001 01.30

Keywords: MACRO

One-line description:

Conditional instr. .IF with rational oper. in Macro creates bad code

Problem:

The use of the conditional instruction, .IF, with rational operator (.EQ.,.NE.,.LT.,.GT.,.LE.,.GE.) in a macro functions incorrectly. The following program demonstrates this problem:

```

          BUG        MACRO          &VAR
          .IF &VAR .LE. 0 SUB&&&
          NOP
          NOP
SUB&&&    NOP
          NOP
          MEND

          BUG 3
          BUG -1
          BUG 0
          END

```

Passing a 3 appears to create correct code, but 0 causes a ML error. Passing -1 to the MACRO creates code which doesn't call the subroutine. This is incorrect since -1 is less than 0. This same problem occurred with all the rational operators on all processors. The problem was consistent on the 64000, VAX, and 9000.

Signed off 08/25/86 in release 101.40

Number: D200053199 Product: Z80 ASSEMB 500 64842S001 01.30

One-line description:

Z80 assembler allowing illegal instructions.

Problem:

The following instructions are illegal, but no assembler errors are generated:

```

"Z80"
  ADD IX,HL
  ADD HL,IX

```

Temporary solution:

Do not use these illegal instructions.

Signed off 08/25/86 in release 101.40

- Z80 ASSEMB -

Number: D200053322 Product: Z80 ASSEMB 500 64842S001 01.30

One-line description:

Macro def. including .IF, within a IF causes assembler to stop code gen.

Problem:

If you have a ".IF" in a macro definition and that macro definition is within a conditional assembly "IF" then no code is generated. The program provided demonstrates the problem (see submitter text).

Temporary solution:

Pull the macro definition outside of the conditional if. No code will be generated for the definition.

"Z80"

```

ESSAI      EQU      0

MAC        MACRO
           .IF      ESSAI.EQ.0  FIN
LABEL     LD        A,0
FIN       MEND

           IF      ESSAI
           MAC
           ENDIF

START     LD        A,3

```

Signed off 08/25/86 in release 101.40

Number: D200049205 Product: Z80 ASSEMB 500 64842S001 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 101.40

- Z80 ASSEMB -

Number: 5000121178 Product: Z80 ASSEMB VAX 64842S003 01.30

One-line description:

Macro def. including .IF, within a IF causes assembler to stop code gen.

Problem:

If you have a ".IF" in a macro definition and that macro definition is within a conditional assembly "IF" then no code is generated. The program provided demonstrates the problem (see submitter text).

Temporary solution:

Pull the macro definition outside of the conditional if. No code will be generated for the definition.

"Z80"

```

ESSAI      EQU      0

MAC        MACRO
           .IF      ESSAI.EQ.0  FIN
LABEL     LD        A,0
FIN       MEND

           IF      ESSAI
           MAC
           ENDIF

START     LD        A,3

```

Signed off 08/25/86 in release 301.60

Number: D200046847 Product: Z80 ASSEMB VAX 64842S003 01.20

One-line description:

Assembler should denote an error on non-absolute .SET expressions.

Signed off 08/25/86 in release 301.60

Number: D200048231 Product: Z80 ASSEMB VAX 64842S003 01.40

Keywords: MACRO

One-line description:

Conditional instr. .IF with rational oper. in Macro creates bad code

Problem:

The use of the conditional instruction, .IF, with rational operator (.EQ.,.NE.,.LT.,.GT.,.LE.,.GE.) in a macro functions incorrectly. The following program demonstrates this problem:

```

BUG        MACRO      &VAR
           .IF &VAR .LE. 0 SUB&&&
           NOP
           NOP
           NOP
           NOP
SUB&&&&

```

- Z80 ASSEMB -

MEND

```

BUG 3
BUG -1
BUG 0
END

```

Passing a 3 appears to create correct code, but 0 causes a ML error. Passing -1 to the MACRO creates code which doesn't call the subroutine. This is incorrect since -1 is less than 0. This same problem occurred with all the rational operators on all processors. The problem was consistent on the 64000, VAX, and 9000.

Signed off 08/25/86 in release 301.60

Number: D200053207 Product: Z80 ASSEMB VAX 64842S003 01.40

One-line description:
Z80 assembler allowing illegal instructions.

Problem:
The following instructions are illegal, but no assembler errors are generated:

```

"Z80"
  ADD  IX,HL
  ADD  HL,IX

```

Temporary solution:
Do not use these illegal instructions.

Signed off 08/25/86 in release 301.60

Number: D200049213 Product: Z80 ASSEMB VAX 64842S003 00.00

One-line description:
Linker output file should use alternate file extension.

Signed off 08/25/86 in release 301.60

Number: D200013987 Product: Z80/NSC800 C 64824 01.01

Keywords: PASS 1

One-line description:
No warning or error: taking the sizeof a struct var. not declared.

Problem:
The compiler should generate an error in the following code.

```

"C"
"Z80"
main () {
    int y;
    y = sizeof(struct x);
}

```

If x is not declared or is declared as anything other than a structure, the program compiles with no error messages or warnings. It stores as the size zero bytes.

Signed off 08/25/86 in release 401.03

Number: D200025668 Product: Z80/NSC800 C 64824 01.01

Keywords: CODE GENERATOR

One-line description:
Dereferenced and incremented 2nd field of structure fails when parameter

Problem:
When the second pointer field of a structure is dereferenced and incremented and passed as a parameter, the code generated puts the result in the data area instead of back on the stack for the calling routine. This does not occur with any other field in the structure, only the second one.

```

Example:
"C"
"8085"
struct strct { char *ptr1; char *ptr2; };
func(strct_ptr)
struct strct *strct_ptr;
{
    ++strct_ptr -> ptr1;
    ++strct_ptr -> ptr2; /* This expression causes the problem */
}

```

Temporary solution:
Assign the dereferenced field to a temporary variable of the appropriate type, then increment the temporary variable. Finally, assign the temporary variable to the dereferenced structure field:

```

struct strct { char *ptr1; char *ptr2; };
func(strct_ptr)
struct strct *strct_ptr;

```

```

{
  int temp1;
  ++struct_ptr ->ptr1;
  temp1 = struct_ptr ->ptr2;
  ++temp1;
  struct_ptr ->ptr2 = temp1;
}

```

Signed off 08/25/86 in release 401.03

Number: D200026989 Product: Z80/NSC800 C 64824 01.01

One-line description:
Incorrect code gen by assignment to deref'd 8 bit field of structure.

Problem:
When an 8 bit field of a structure is dereferenced and used as the left hand side of an assignment statement using the += operator, incorrect code is generated. This does not occur with the first field in the structure. The incorrect code is an LHLD Dmain instruction which loads H and L with garbage since Dmain is uninitialized. The following code is an example of this:

```

"C"
"processor name"
$RECURSIVE OFF$
main() {
  extern char KEY,X1();
  struct ROW {
    char A;
    char B;
  } *PTR;
  PTR->B+=X1(KEY); /*This instruction generates an incorrect
  }                LHLD Dmain instruction*/
  If the = operator is used instead of the += operator in the assignment
  statement, the problem does not occur.

```

Temporary solution:
Use a temporary variable:
temp = PTR->B;
temp+=X1(KEY);
PTR->B = temp;

Signed off 08/25/86 in release 401.03

Number: D200027458 Product: Z80/NSC800 C 64824 01.01

One-line description:
Incorrect code for switch on dereferenced non-integer structure element.

Problem:
Incorrect code is generated for a switch statement when the switch is on a dereferenced element of a structure which is not the first element and is not an integer. The following code exemplifies the problem:

```

"C"
"processor name"
typedef struct {

```

- Z80/NSC800 C -

```

  char data1;
  long data2;
  char data3;
  int data4;
  long data5;
} group;
extern group *grp_ptr;
main() {
  switch(grp_ptr->data4) { /*This works fine*/
  case 0: break;
  }
  switch(grp_ptr->data5) { /*This generates incorrect code*/
  case 0: break;
  }
}

```

Temporary solution:
Use a temporary variable of the appropriate type in the switch statement:
long temp;

```

temp = grp_ptr->data5;
switch(temp){}

```

If the field you are dereferencing is an enumeration type this temporary solution will not work. You will have to place the enumeration type as the first field in the structure.

Signed off 08/25/86 in release 401.03

Number: D200027771 Product: Z80/NSC800 C 64824 01.01

One-line description:
No form feed between the expanded listing and the cross reference table.

Problem:
During compilation, with XREF option on, the compiler does not provide a form feed (FF) in the listing file. The XREF starts on the same page as the end of the listing. Also, the page number says 535 when it should be page 2.

Temporary solution:
After compiling with the xref option, edit the expanded listing file and insert a "control L" before the beginning of the cross reference listing.

Signed off 08/25/86 in release 401.03

Number: D200027888 Product: Z80/NSC800 C 64824 01.01

One-line description:
Addition of dereferenced pointers to structures may fail.

Problem:
Adding two operands that are dereferenced pointers to structures may fail because the compiler forgets to store the H and L registers and overwrites them. The following code is an example of this:

- Z80/NSC800 C -

```

"C"
"processor name"
struct tree {
    int distance;
    int x_start;
    int x_range;
};
trees(treex)
struct tree *treex;
{
    treex->distance=treex->x_start+treex->x_range; /*This line
generates an ADD HL,DE instruction to index
into the structure tree, but overwrites H and L
in the next instruction instead of storing it*/
}

```

Temporary solution:
Use local temporary variables of the appropriate types to store the values of the dereferenced structure pointers before using them in a complex expression. Depending on the complexity of the expression, more than one temporary variable may have to be used.

```

trees(treex)
struct tree *treex;
{
    int x;
    x = treex->x_start;
    treex->distance= x + treex->x_range;
}

```

Signed off 08/25/86 in release 401.03

Number: D200028746 Product: Z80/NSC800 C 64824 01.01

One-line description:
Incorrect code when indexing into an array passed as a parameter.

Problem:
The code generator produces incorrect code when indexing into an array which was passed to a function. The HL register pair is overwritten in the following example before it is saved:

```

"C"
"Z80"
char *func(var1,out)
char var1,out[];
{
    out[6] = 1 + var1; /*HL register pair is overwritten before saved*/
    return(out);
}

```

Temporary solution:
Use a local temporary variable:

```

"C"
"Z80"
char *func(var1,out)
char var1,out[];

```

```

{
    char temp;
    temp = out[6];
    temp = 1 + var1;
    out[6] = temp;
    return(out);
}

```

Signed off 08/25/86 in release 401.03

Number: D200028779 Product: Z80/NSC800 C 64824 01.01

One-line description:
Dereferencing pointers to structures in assignment statements may fail.

Problem:
Dereferencing a pointer to a structure in an assignment statement may produce incorrect code which overwrites the HL register pair before saving it. The following code is an example:

```

"C"
"Z80"
typedef struct {
    int *data1;
    long *data2;
    long *data3;
    long *data4;
} alldata;
func(var1)
alldata *var1;
{
    var1->data4 = var1->data2;
}

```

Temporary solution:
Use a temporary variable:

```

func(var1)
alldata *var1;
{
    long *temp;
    temp = var1->data2;
    var1->data4 = temp;
}

```

Signed off 08/25/86 in release 401.03

Number: D200031427 Product: Z80/NSC800 C 64824 01.01

One-line description:
++ and -- operators evaluated with improper precedence.

Problem:
According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

```

Example 1: array[index++] = 1;
Example 2: array[index] = 1;
           index++;

```

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:
Separate the expression as shown in example 2.

Signed off 08/25/86 in release 401.03

Number: D200033225 Product: Z80/NSC800 C 64824 01.01

One-line description:
Comparing character to zero in while loop generates incorrect code.

Problem:
If you compare a character variable to zero in a while loop incorrect code is generated. The below code demonstrates the problem.

```
"C"
"6809"

proc()
{
    char timeout = 10;

    while(timeout--); /* Code generated here causes infinite loop.
}

```

The code generated for the while loop clears the A register and then compares the D register to -1. Therefore the condition is never met.

Temporary solution:
Declare the variable used in the test condition as an integer.

```
"C"
"6809"

proc()
{
    int timeout = 10;

    while (timeout--);
}

```

Signed off 08/25/86 in release 401.03

Number: D200034264 Product: Z80/NSC800 C 64824 01.01

Keywords: CODE GENERATOR

One-line description:
A shift assignment operation (<<=) generates incorrect code.

Problem:
If a shift assignment is used instead of a shift within an assignment, the compiler uses the high byte of the variable to be used as the shift

- Z80/NSC800 C -

counter instead of the low byte. The following is an example:

```
"C"
"processor name"
char data=1;
int shift=4;
main () {
    data=data<<shift; /* works correctly */
    data<<=shift; /* uses higher order byte of "shift" */
}

```

Temporary solution:
Use

```
data=data<<shift;
instead of
data<<=shift;

```

Signed off 08/25/86 in release 401.03

Number: D200035899 Product: Z80/NSC800 C 64824 01.01

Keywords: CODE GENERATOR

One-line description:
16 bit comparison on a 8 bit unsigned short field.

Problem:
IMPROPER CODE GENERATED FOR STATEMENT INVOLVING unsigned short VARIABLE UNLESS EXPLICITLY RE-CAST AS unsigned short.

```
main()
{
    static unsigned short digit_index;
    static unsigned short digit[12];
    int a,b;
    if (digit[digit_index--]){
        a=4;
        b=4;}
    else{
        a=5;
        b=5;}
}

IMPROPER CODE IS GENERATED FOR THE COMPARISON (ie THE COMPARISON IS DONE ON 16 BITS (8 OF WHICH HAVE BEEN CLEARED) AGAINST #0FFFFH.
12/10/85: The problem also arises if you compare a constant against an unsigned short. For example if you declared:
#define constant ~0
unsigned short var;
and later compared these two the compiler will zero out the upper byte of the variable var and then compare it to FFFFH. Thus, the condition is never met.

```

12/16/85: Another example of incorrect code being generated when a char variable is used in a test condition is as follows:

```
char a;
main()
{

```

- Z80/NSC800 C -


```

a = -1;
if(a == -1)
  a = 'A';
}

```

Temporary solution:
IF THE LINE IN QUESTION IS CHANGED TO:

```
if ((unsigned short)digit[digit_index]--){
```

CORRECT CODE IS GENERATED ALTHOUGH digit[] HAS ALREADY BEEN DECLARED unsigned short.
12/10/85: Declare the constant as a short. In other words:
#define constant OFFH.
12/16/85: If only 128 valid characters are required the variable can be declared as a short int.

Signed off 08/25/86 in release 401.03

Number: D200040782 Product: Z80/NSC800 C 64824 01.01

Keywords: PASS 3

One-line description:
Pass 3 fails to detect relative jump address out-of-range.

Problem:
Pass 3 of the compilation process may fail to detect a relative jump which is out of range. In the test program submitted the relative jump is generated for an IF..THEN statement while the compiler option OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:
As a temporary work around disable the compiler option OPTIMIZE around those sections of code which are suspect.

Signed off 08/25/86 in release 401.03

Number: D200041186 Product: Z80/NSC800 C 64824 01.01

One-line description:
Problem with integer pointer in conditional statement.

Problem:
In the following example, two loads are performed, but no other code is generated to check for zero value.

```

"C"
"processor name"
#define NULL 0
fct(parm)
int *parm;
{
  if (parm - NULL)
    parm = 10;
}

```

Signed off 08/25/86 in release 401.03

Number: D200043596 Product: Z80/NSC800 C 64824 01.01

One-line description:
STACK POINTER OFFSETS ARE INCORRECT WHEN ENTERING REAL_TRUNC.

Problem:
Stack pointer offsets to local variables are incorrect on entry into library routine REAL_TRUNC. Below program will demonstrate the problem.
"C"
"Z80"

```

main()
{
  float f;
  int i;

  f = -1.0;
  i = f;
}

```

Temporary solution:
Declare the variables as globals.
"C"
"Z80"

```

float f;
int i;
main()
{
  f = -1;
  i = f;
}

```

Signed off 08/25/86 in release 401.03

Number: D200043968 Product: Z80/NSC800 C 64824 01.01

One-line description:
Illegal forward reference error generated when initializing structures.

Signed off 08/25/86 in release 401.03

Number: D200044685 Product: Z80/NSC800 C 64824 01.01

One-line description:
Stack offset to parameter is incorrect.

Signed off 08/25/86 in release 401.03

Number: D200045518 Product: Z80/NSC800 C 64824 01.01

One-line description:
Conditional containing 'pointer to func' is not calling correct func.

Temporary solution:

You must break up the conditional statement as follows:

"C"
"Z80"

```
extern struct a{
    char var1;
    char var2;
    int (*sc_decide)();
    char var3;};
```

```
extern struct a *trans_tbl;
```

```
main()
{
    int (*temp)();          /* Add these temp. var's. */
    int trans_on;

    temp = trans_tbl->sc_decide;
    trans_on = (*temp)();

    if (trans_on);
}
```

Signed off 08/25/86 in release 401.03

Number: D200045526 Product: Z80/NSC800 C 64824 01.01

One-line description:
Character being sign converted to a word causing conditional to be false

Temporary solution:

Typecast both KEY_IN and the constant to characters.

"C"
"Z80"

```
main()
{
    char KEY_IN;

    while (((char)KEY_IN) == ((char) 0xFF));
}
```

Signed off 08/25/86 in release 401.03

Number: D200045872 Product: Z80/NSC800 C 64824 01.01

One-line description:
Updating & assigning ptr a new value causes compiler to genera

Problem:

Updating and assigning a pointer a new value causes the result to be stored in the wrong memory location.

"C"
"Z80"

```
int func(pl,time)
int pl;
short *time;

{
    int t_val;

    if (*time) {
        *(time + 1) += (char)t_val; /* Result of this expression is
                                     stored in wrong memory loc. */
    }
}
```

Temporary solution:

Use a local variable to hold the updated pointer value.

"C"
"Z80"

```
int func1(pl,time)
int pl;
short *time;

{
    int t_val;
    short *ptr;

    ptr = time + 1;
    if(*time) {
        *ptr += (char)t_val;
    }
}
```

Signed off 08/25/86 in release 401.03

Number: D200046177 Product: Z80/NSC800 C 64824 01.01

One-line description:
Post increment of pointer results in incorrect code.

Problem:

Post increment of a pointer value will cause incorrect code to be generated. First, the pointer is pre-incremented rather than post incremented. Secondly, the result is stored in the wrong location.

"C"
"8085"
\$SHORT ARITH +\$
\$RECURSIVE OFF\$

\$SEPARATE ON\$

```
main()
{
  long ai[2],*aiptr,a1,a2;
  ai[0]=0L;
  ai[1]=1L;
  aiptr=ai;
  ai=*aiptr++; /* Problem Statement. *aiptr is pre-incremented
                and the result is stored in wrong location. */
}
```

Temporary solution:

Increment the pointer after the assignment is made.

```
Use: a1=*aiptr;
     *aiptr++;
```

Rather than:

```
a1=*aiptr++;
```

Signed off 08/25/86 in release 401.03

Number: D200047662 Product: Z80/NSC800 C 64824 01.01

One-line description:

TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 401.03

Number: D200050740 Product: Z80/NSC800 C 300 64824S004 01.00

One-line description:

Defining TRUE and FALSE as global may result in duplicate symbol names.

Problem:

Defining the variables (constants) TRUE and FALSE to be global may result in a duplicate symbol error during a link. These variables are incorrectly defined as global in the Zwordcmp routine located in 'Zlibrary'.

NOTE: Redefining the values of TRUE and/or FALSE is not a legal Pascal operation. Redefinition of these constants is therefore not supported when using the HP 64000 compiler.

Temporary solution:

Obtain the source to Zwordcmp from your local HP Systems Engineer.

Signed off 08/25/86 in release 401.10

Number: D200051300 Product: Z80/NSC800 C 300 64824S004 01.00

One-line description:

++ and -- operators evaluated with improper precedence.

Problem:

According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

```
Example 1: array[index++] = 1;
```

```
Example 2: array[index] = 1;
           index++;
```

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:

Separate the expression as shown in example 2.

Signed off 08/25/86 in release 401.10

Number: D200052308 Product: Z80/NSC800 C 300 64824S004 00.00

Keywords: CODE GENERATOR

One-line description:

Incorrect opcode "MOV A,ACC" allowed by our assembler

Problem:

The instruction "MOV A,ACC" was assemble and emulated by our products; however, the Intel 8051 goes into the weeds at this instruction. At first glance the machine code in the assembler listing appears valid (MOV A,ACC ->0000 E5E0), but the bottom of page 8-35 in Intel's microcontroller handbook states: *MOV A,ACC is not a valid instruction.

Neither our manuals nor AMD's user manual mention this instruction.

Signed off 08/25/86 in release 401.10

Number: D200059089 Product: Z80/NSC800 C 300 64824S004 01.00

One-line description:

Host compilers do not put absolute pats specifications in relocatables

Problem:

Host compilers do not specify the full path name in the relocatable file.

Signed off 08/25/86 in release 401.10

Number: D200049072 Product: Z80/NSC800 C 300 64824S004 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 401.10

Number: D200025676 Product: Z80/NSC800 C 500 64824S001 01.10

Keywords: CODE GENERATOR

One-line description:

Dereferenced and incremented 2nd field of structure fails when parameter

Problem:

When the second pointer field of a structure is dereferenced and incremented and passed as a parameter, the code generated puts the result in the data area instead of back on the stack for the calling routine. This does not occur with any other field in the structure, only the second one.

Example:

```
"C"
"8085"
struct strct { char *ptr1; char *ptr2; };
func(strct_ptr)
struct strct *strct_ptr;
{
  ++strct_ptr -> ptr1;
  ++strct_ptr -> ptr2; /* This expression causes the problem */
}
```

Temporary solution:

Assign the dereferenced field to a temporary variable of the appropriate type, then increment the temporary variable. Finally, assign the temporary variable to the dereferenced structure field:

```
struct strct { char *ptr1; char *ptr2; };
func(strct_ptr)
struct strct *strct_ptr;
{
  int temp1;
  ++strct_ptr -> ptr1;
  temp1 = strct_ptr -> ptr2;
  ++temp1;
  strct_ptr -> ptr2 = temp1;
}
```

Signed off 08/25/86 in release 101.50

Number: D200026997 Product: Z80/NSC800 C 500 64824S001 01.10

One-line description:

Incorrect code gen by assignment to deref'd 8 bit field of structure.

Problem:

When an 8 bit field of a structure is dereferenced and used as the left hand side of an assignment statement using the += operator, incorrect code is generated. This does not occur with the first field in the structure. The incorrect code is an LHLD Dmain instruction which loads H and L with garbage since Dmain is uninitialized. The following code is an example of this:

"C"

```

"processor name"
$RECURSIVE OFF$
main() {
extern char KEY,X1();
struct ROW {
    char A;
    char B;
} *PTR;
PTR->B+=X1(KEY);    /*This instruction generates an incorrect
                    LHL Dmain instruction*/
}
If the = operator is used instead of the += operator in the assignment
statement, the problem does not occur.

```

Temporary solution:
 Use a temporary variable:
 temp = PTR->B;
 temp+=X1(KEY);
 PTR->B = temp;

Signed off 08/25/86 in release 101.50

Number: D200027896 Product: Z80/NSC800 C 500 64824S001 01.10

One-line description:
 Addition of dereferenced pointers to structures may fail.

Problem:
 Adding two operands that are dereferenced pointers to structures may fail because the compiler forgets to store the H and L registers and overwrites them. The following code is an example of this:

```

"C"
"processor name"
struct tree {
    int distance;
    int x_start;
    int x_range;
};
trees(treex)
struct tree *treex;
{
    treex->distance=treex->x_start+treex->x_range; /*This line
                                                generates an ADD HL,DE instruction to index
                                                into the structure tree, but overwrites H and L
                                                in the next instruction instead of storing it*/
}

```

Temporary solution:
 Use local temporary variables of the appropriate types to store the values of the dereferenced structure pointers before using them in a complex expression. Depending on the complexity of the expression, more than one temporary variable may have to be used.

```

trees(treex)
struct tree *treex;
{
    int x;
    x = treex->x_start;
}

```

- Z80/NSC800 C -

```

    treex->distance= x + treex->x_range;
}

```

Signed off 08/25/86 in release 101.50

Number: D200028753 Product: Z80/NSC800 C 500 64824S001 01.10

One-line description:
 Incorrect code when indexing into an array passed as a parameter.

Problem:
 The code generator produces incorrect code when indexing into an array which was passed to a function. The HL register pair is overwritten in the following example before it is saved:

```

"C"
"Z80"
char *func(var1,out)
char var1,out[];
{
    out[6] = 1 + var1; /*HL register pair is overwritten before saved*/
    return(out);
}

```

Temporary solution:
 Use a local temporary variable:

```

"C"
"Z80"
char *func(var1,out)
char var1,out[];
{
    char temp;
    temp = out[6];
    temp = 1 + var1;
    out[6] = temp;
    return(out);
}

```

Signed off 08/25/86 in release 101.50

Number: D200029223 Product: Z80/NSC800 C 500 64824S001 01.10

One-line description:
 Dereferencing pointers to structures in assignment statements may fail.

Problem:
 Dereferencing a pointer to a structure in an assignment statement may produce incorrect code which overwrites the HL register pair before saving it. The following code is an example:

```

"C"
"Z80"
typedef struct {
    int *data1;
    long *data2;
    long *data3;
}

```

- Z80/NSC800 C -

```

        long *data4;
    } alldata;
func(var1)
alldata *var1;
{
    var1->data4 = var1->data2;
}

```

Temporary solution:
Use a temporary variable:

```

func(var1)
alldata *var1;
{
    long *temp;
    temp = var1->data2;
    var1->data4 = temp;
}

```

Signed off 08/25/86 in release 101.50

Number: D200031435 Product: Z80/NSC800 C 500 64824S001 01.10

One-line description:

++ and -- operators evaluated with improper precedence.

Problem:

According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

Example 1: array[index++] = 1;

Example 2: array[index] = 1;
 index++;

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:

Separate the expression as shown in example 2.

Signed off 08/25/86 in release 101.50

Number: D200033233 Product: Z80/NSC800 C 500 64824S001 01.10

One-line description:

Comparing character to zero in while loop generates incorrect code.

Problem:

If you compare a character variable to zero in a while loop incorrect code is generated. The below code demonstrates the problem.

```

"C"
"6809"

proc()
{
    char timeout = 10;

```

- Z80/NSC800 C -

```

        while(timeout--); /* Code generated here causes infinite loop.
    }

```

The code generated for the while loop clears the A register and then compares the D register to -1. Therefore the condition is never met.

Temporary solution:

Declare the variable used in the test condition as an integer.

```

"C"
"6809"

```

```

proc()
{
    int  timeout = 10;

    while (timeout--);
}

```

Signed off 08/25/86 in release 101.50

Number: D200034272 Product: Z80/NSC800 C 500 64824S001 01.10

Keywords: CODE GENERATOR

One-line description:

A shift assignment operation (<<=) generates incorrect code.

Problem:

If a shift assignment is used instead of a shift within an assignment, the compiler uses the high byte of the variable to be used as the shift counter instead of the low byte. The following is an example:

```

"C"
"procesor name"
char data=1;
int shift=4;
main () {
    data=data<<shift; /* works correctly */
    data<<=shift; /* uses higher order byte of "shift" */
}

```

Temporary solution:

```

Use
    data=data<<shift;
instead of
    data<<=shift;

```

Signed off 08/25/86 in release 101.50

Number: D200035907 Product: Z80/NSC800 C 500 64824S001 01.10

Keywords: CODE GENERATOR

One-line description:

16 bit comparison on a 8 bit unsigned short field.

Problem:

- Z80/NSC800 C -

IMPROPER CODE GENERATED FOR STATEMENT INVOLVING unsigned short VARIABLE UNLESS EXPLICITLY RE-CAST AS unsigned short.

```
main()
{
static unsigned short digit_index;
static unsigned short digit[12];
int a,b;
if (digit[digit_index--]){
a=4;
b=4;}
else{
a=5;
b=5;}
}
IMPROPER CODE IS GENERATED FOR THE COMPARISON (ie THE COMPARISON IS DONE
ON 16 BITS (8 OF WHICH HAVE BEEN CLEARED) AGAINST #0FFFFH.
12/10/85: The problem also arises if you compare a constant against
an unsigned short. For example if you declared:
#define constant ~0
unsigned short var;
and later compared these two the compiler will zero out the upper byte
of the variable var and then compare it to FFFFH. Thus, the condition
is never met.
```

12/16/85: Another example of incorrect code being generated when a char variable is used in a test condition is as follows:

```
char a;
main()
{
a = -1;
if(a == -1)
a = 'A';
}
```

Temporary solution:
IF THE LINE IN QUESTION IS CHANGED TO:

```
if ((unsigned short)digit[digit_index--]){
CORRECT CODE IS GENERATED ALTHOUGH digit[] HAS ALREADY BEEN
DECLARED unsigned short.
12/10/85: Declare the constant as a short. In other words:
#define constant OFFH.
12/16/85: If only 128 valid characters are required the variable can
be declared as a short int.
```

Signed off 08/25/86 in release 101.50

Number: D200037176 Product: Z80/NSC800 C 500 64824S001 01.20

Keywords: PASS 3

One-line description:
Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:

Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect data being output to the list file. In selected cases, machine code will be incorrectly listed. For example, consider the following Pascal program.

```
$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;

VAR
a, b : BOOLEAN;

PROCEDURE one;

BEGIN
a := b;
END;
```

In the example listed above, the output file will denote machine code of the form FFFFC00001 for one of the generated assembly statements. The correct value should be C8000001. This problem is caused by an incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 101.50

Number: D200040790 Product: Z80/NSC800 C 500 64824S001 01.20

Keywords: PASS 3

One-line description:
Pass 3 fails to detect relative jump address out-of-range.

Problem:
Pass 3 of the compilation process may fail to detect a relative jump which is out of range. In the test program submitted the relative jump is generated for an IF..THEN statement while the compiler option OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:
As a temporary work around disable the compiler option OPTIMIZE around those sections of code which are suspect.

Signed off 08/25/86 in release 101.50

Number: D200041350 Product: Z80/NSC800 C 500 64824S001 01.20

One-line description:
Problem with integer pointer in conditional statement.

Problem:
In the following example, two loads are performed, but no other code is generated to check for zero value.

```
"C"
"processor name"
#define NULL 0
fct(parm)
int *parm;
{
  if (parm - NULL)
    parm = 10;
}
```

Signed off 08/25/86 in release 101.50

Number: D200045997 Product: Z80/NSC800 C 500 64824S001 01.20

One-line description:
Title description is incorrect.

Signed off 08/25/86 in release 101.50

Number: D200046078 Product: Z80/NSC800 C 500 64824S001 01.20

One-line description:
Updating & assigning ptr a new value causes compiler to genera

Problem:
Updating and assigning a pointer a new value causes the result to
be stored in the wrong memory location.

```
"C"
"Z80"
```

```
int func(p1,time)
int p1;
short *time;
```

```
{
  int t_val;
  if (*time) {
    *(time + 1) += (char)t_val; /* Result of this expression is
                                stored in wrong memory loc. */
  }
}
```

Temporary solution:
Use a local variable to hold the updated pointer value.

```
"C"
"Z80"
```

```
int func1(p1,time)
int p1;
short *time;
```

```
{
  int t_val;
  short *ptr;

  ptr = time +1;
```

- Z80/NSC800 C -

```
if(*time) {
  *ptr += (char)t_val;
}
```

Signed off 08/25/86 in release 101.50

Number: D200046185 Product: Z80/NSC800 C 500 64824S001 01.20

One-line description:
Post increment of pointer results in incorrect code.

Problem:
Post increment of a pointer value will cause incorrect code to be
generated. First, the pointer is pre-incremented rather than
post incremented. Secondly, the result is stored in the wrong location.

```
"C"
"8085"
$SHORT ARITH +$
$RECURSIVE OFF$
$SEPARATE ON$
```

```
main()
{
  long ai[2],*aiptr,a1,a2;
  ai[0]=0L;
  ai[1]=1L;
  aiptr=ai;
  ai=*aiptr++; /* Problem Statement. *aiptr is pre-incremented
                and the result is stored in wrong location. */
}
```

Temporary solution:
Increment the pointer after the assignment is made.
Use: a1=*aiptr;
 *aiptr++;

Rather than:
a1=*aiptr++;

Signed off 08/25/86 in release 101.50

Number: D200047670 Product: Z80/NSC800 C 500 64824S001 01.20

One-line description:
TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 101.50

Number: D200049775 Product: Z80/NSC800 C 500 64824S001 00.00

One-line description:
NO CROSS REFERENCE TABLE IS GENERATED

Problem:
"C" COMPILERS DO NOT GENERATE A CROSS REFERENCE TABLE ON THE
VAX.

- Z80/NSC800 C -

Temporary solution:
NONE KNOWN AT PRESENT

Signed off 04/18/86 in release 101.50

Number: D200059063 Product: Z80/NSC800 C 500 64824S001 01.40

One-line description:
Host compilers do not put absolute pats specifications in relocatables

Problem:
Host compilers do not specify the full path name in the relocatable file.

Temporary solution:
No known temporary solution.

Signed off 08/25/86 in release 101.50

Number: D200049056 Product: Z80/NSC800 C 500 64824S001 00.00

One-line description:
Linker output file should use alternate file extension.

Signed off 08/25/86 in release 101.50

Number: D200025684 Product: Z80/NSC800 C VAX 64824S003 01.10

Keywords: CODE GENERATOR

One-line description:
Dereferenced and incremented 2nd field of structure fails when parameter

Problem:
When the second pointer field of a structure is dereferenced and incremented and passed as a parameter, the code generated puts the result in the data area instead of back on the stack for the calling routine. This does not occur with any other field in the structure, only the second one.

Example:
"C"
"8085"
struct strct { char *ptr1; char *ptr2; };
func(strct_ptr)
struct strct *strct_ptr;
{
++strct_ptr -> ptr1;
++strct_ptr -> ptr2; /* This expression causes the problem */
}

Temporary solution:
Assign the dereferenced field to a temporary variable of the appropriate type, then increment the temporary variable. Finally, assign the temporary variable to the dereferenced structure field:

```
struct strct { char *ptr1; char *ptr2; };
func(strct_ptr)
struct strct *strct_ptr;
{
    int temp1;
    ++strct_ptr -> ptr1;
    temp1 = strct_ptr -> ptr2;
    ++temp1;
    strct_ptr -> ptr2 = temp1;
}
```

Signed off 08/25/86 in release 301.80

Number: D200027003 Product: Z80/NSC800 C VAX 64824S003 01.20

One-line description:
Incorrect code gen by assignment to deref'd 8 bit field of structure.

Problem:
When an 8 bit field of a structure is dereferenced and used as the left hand side of an assignment statement using the += operator, incorrect code is generated. This does not occur with the first field in the structure. The incorrect code is an LHLD Dmain instruction which loads H and L with garbage since Dmain is uninitialized. The following code is an example of this:
"C"

```
"processor name"
$RECURSIVE OFF$
main() {
extern char KEY,X1();
struct ROW {
char A;
char B;
} *PTR;
PTR->B+=X1(KEY); /*This instruction generates an incorrect
} LHL Dmain instruction*/
If the = operator is used instead of the += operator in the assignment
statement, the problem does not occur.
```

Temporary solution:
Use a temporary variable:
temp = PTR->B;
temp+=X1(KEY);
PTR->B = temp;

Signed off 08/25/86 in release 301.80

Number: D200027904 Product: Z80/NSC800 C VAX 64824S003 01.20

One-line description:
Addition of dereferenced pointers to structures may fail.

Problem:
Adding two operands that are dereferenced pointers to structures may fail because the compiler forgets to store the H and L registers and overwrites them. The following code is an example of this:

```
"c"
"processor name"
struct tree {
int distance;
int x_start;
int x_range;
};
trees(treex)
struct tree *treex;
{
treex->distance=treex->x_start+treex->x_range; /*This line
} generates an ADD HL,DE instruction to index
into the structure tree, but overwrites H and L
in the next instruction instead of storing it*/
```

Temporary solution:
Use local temporary variables of the appropriate types to store the values of the dereferenced structure pointers before using them in a complex expression. Depending on the complexity of the expression, more than one temporary variable may have to be used.

```
trees(treex)
struct tree *treex;
{
int x;
x = treex->x_start;
```

- Z80/NSC800 C -

```
treex->distance= x + treex->x_range;
}
```

Signed off 08/25/86 in release 301.80

Number: D200028761 Product: Z80/NSC800 C VAX 64824S003 01.20

One-line description:
Incorrect code when indexing into an array passed as a parameter.

Problem:
The code generator produces incorrect code when indexing into an array which was passed to a function. The HL register pair is overwritten in the following example before it is saved:

```
"c"
"Z80"
char *func(var1,out)
char var1,out[];
{
out[6] = 1 + var1; /*HL register pair is overwritten before saved*/
return(out);
}
```

Temporary solution:
Use a local temporary variable:

```
"c"
"Z80"
char *func(var1,out)
char var1,out[];
{
char temp;
temp = out[6];
temp = 1 + var1;
out[6] = temp;
return(out);
}
```

Signed off 08/25/86 in release 301.80

Number: D200029215 Product: Z80/NSC800 C VAX 64824S003 01.20

One-line description:
Dereferencing pointers to structures in assignment statements may fail.

Problem:
Dereferencing a pointer to a structure in an assignment statement may produce incorrect code which overwrites the HL register pair before saving it. The following code is an example:

```
"c"
"Z80"
typedef struct {
int *data1;
long *data2;
long *data3;
```

- Z80/NSC800 C -

```

        long *data4;
    } alldata;
func(var1)
alldata *var1;
{
    var1->data4 = var1->data2;
}

```

Temporary solution:
Use a temporary variable:

```

func(var1)
alldata *var1;
{
    long *temp;
    temp = var1->data2;
    var1->data4 = temp;
}

```

Signed off 08/25/86 in release 301.80

Number: D200031443 Product: Z80/NSC800 C VAX 64824S003 01.20

One-line description:

++ and -- operators evaluated with improper precedence.

Problem:

According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

Example 1: array[index++] = 1;

Example 2: array[index] = 1;
index++;

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:

Separate the expression as shown in example 2.

Signed off 08/25/86 in release 301.80

Number: D200033241 Product: Z80/NSC800 C VAX 64824S003 01.20

One-line description:

Comparing character to zero in while loop generates incorrect code.

Problem:

If you compare a character variable to zero in a while loop incorrect code is generated. The below code demonstrates the problem.

"C"

"6809"

```

proc()
{
    char timeout = 10;

```

- Z80/NSC800 C -

```

        while(timeout--); /* Code generated here causes infinite loop.
    }

```

The code generated for the while loop clears the A register and then compares the D register to -1. Therefore the condition is never met.

Temporary solution:

Declare the variable used in the test condition as an integer.

"C"

"6809"

```

proc()
{
    int timeout = 10;

    while (timeout--);
}

```

Signed off 08/25/86 in release 301.80

Number: D200034280 Product: Z80/NSC800 C VAX 64824S003 01.20

Keywords: CODE GENERATOR

One-line description:

A shift assignment operation (<<=) generates incorrect code.

Signed off 08/25/86 in release 301.80

Number: D200035915 Product: Z80/NSC800 C VAX 64824S003 01.20

Keywords: CODE GENERATOR

One-line description:

16 bit comparison on a 8 bit unsigned short field.

Problem:

IMPROPER CODE GENERATED FOR STATEMENT INVOLVING unsigned short VARIABLE UNLESS EXPLICITLY RE-CAST AS unsigned short.

```

main()
{
    static unsigned short digit_index;
    static unsigned short digit[12];
    int a,b;
    if (digit[digit_index--]){
        a=4;
        b=4;}
    else{
        a=5;
        b=5;}
}

```

IMPROPER CODE IS GENERATED FOR THE COMPARISON (ie THE COMPARISON IS DONE ON 16 BITS (8 OF WHICH HAVE BEEN CLEARED) AGAINST #0FFFFH.

12/10/85: The problem also arises if you compare a constant against an unsigned short. For example if you declared:

- Z80/NSC800 C -

```
#define constant ~0
unsigned short var;
and later compared these two the compiler will zero out the upper byte
of the variable var and then compare it to FFFFH. Thus, the condition
is never met.
```

12/16/85: Another example of incorrect code being generated when a char variable is used in a test condition is as follows:

```
char a;
main()
{
  a = -1;
  if(a == -1)
    a = 'A';
}
```

Temporary solution:
IF THE LINE IN QUESTION IS CHANGED TO:

```
if ((unsigned short)digit[digit_index]--){
```

CORRECT CODE IS GENERATED ALTHOUGH digit[] HAS ALREADY BEEN
DECLARED unsigned short.

12/10/85: Declare the constant as a short. In other words:

```
#define constant 0FFH.
```

12/16/85: If only 128 valid characters are required the variable can
be declared as a short int.

Signed off 08/25/86 in release 301.80

Number: D200037184 Product: Z80/NSC800 C VAX 64824S003 01.20

Keywords: PASS 3

One-line description:
Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:

Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect
data being output to the list file. In selected cases, machine code
will be incorrectly listed. For example, consider the following
Pascal program.

```
$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;

VAR
  a, b : BOOLEAN;

PROCEDURE one;

BEGIN
  a := b;
END;
```

- Z80/NSC800 C -

In the example listed above, the output file will denote machine code
of the form FFFFC00001 for one of the generated assembly statements.
The correct value should be C8000001. This problem is caused by an
incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 301.80

Number: D200040808 Product: Z80/NSC800 C VAX 64824S003 01.20

Keywords: PASS 3

One-line description:
Pass 3 fails to detect relative jump address out-of-range.

Problem:

Pass 3 of the compilation process may fail to detect a relative jump
which is out of range. In the test program submitted the relative
jump is generated for an IF..THEN statement while the compiler option
OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:

As a temporary work around disable the compiler option OPTIMIZE
around those sections of code which are suspect.

Signed off 08/25/86 in release 301.80

Number: D200041368 Product: Z80/NSC800 C VAX 64824S003 01.20

One-line description:
Problem with integer pointer in conditional statement.

Problem:

In the following example, two loads are performed, but no other code is
generated to check for zero value.

```
"C"
"processor name"
#define NULL 0
fct(parm)
int *parm;
{
  if (parm - NULL)
    parm = 10;
}
```

Signed off 08/25/86 in release 301.80

Number: D200046003 Product: Z80/NSC800 C VAX 64824S003 01.20

One-line description:
Title description is incorrect.

Signed off 08/25/86 in release 301.80

- Z80/NSC800 C -

Number: D200046086 Product: Z80/NSC800 C VAX 64824S003 01.20

One-line description:
Updating & assigning ptr a new value causes compiler to genera

Problem:
Updating and assigning a pointer a new value causes the result to be stored in the wrong memory location.

"C"
"Z80"

```
int func(p1,time)
int pi;
short *time;

{
  int t_val;
  if (*time) {
    *(time + 1) += (char)t_val; /* Result of this expression is
                               stored in wrong memory loc. */
  }
}
```

Temporary solution:
Use a local variable to hold the updated pointer value.

"C"
"Z80"

```
int func1(p1,time)
int pi;
short *time;

{
  int t_val;
  short *ptr;
  ptr = time + 1;
  if(*time) {
    *ptr += (char)t_val;
  }
}
```

Signed off 08/25/86 in release 301.80

Number: D200046193 Product: Z80/NSC800 C VAX 64824S003 01.20

One-line description:
Post increment of pointer results in incorrect code.

Problem:
Post increment of a pointer value will cause incorrect code to be generated. First, the pointer is pre-incremented rather than post incremented. Secondly, the result is stored in the wrong location.

"C"
"8085"
\$SHORT_ARITH +\$

- Z80/NSC800 C -

\$RECURSIVE OFF\$
\$SEPARATE ON\$

```
main()
{
  long ai[2],*aiptr,a1,a2;
  ai[0]=0L;
  ai[1]=1L;
  aiptr=ai;
  ai=*aiptr++; /* Problem Statement. *aiptr is pre-incremented
               and the result is stored in wrong location. */
}
```

Temporary solution:
Increment the pointer after the assignment is made.

Use: ai=*aiptr;
*aiptr++;

Rather than:
ai=*aiptr++;

Signed off 08/25/86 in release 301.80

Number: D200047688 Product: Z80/NSC800 C VAX 64824S003 01.20

One-line description:
TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 301.80

Number: D200055178 Product: Z80/NSC800 C VAX 64824S003 01.50

One-line description:
Compilation on the VAX using batch mode generates incorrect listing file

Problem:
The test files can be found on the VAX750 under user\$disk:[robin.hughes.rgalo.test]. The following test files were used:

1. MTINHST_C. - File which contains one error- a missing '}' on line 70
2. TMTINHST_C. - Error-free version of MTINHST_C.
3. MTOPNDF_C. - File which contains one error - missing declaration for integer 'j'
4. MTOPNDF_T_C. - Error-free version of MTOPNDF_C.

One logical name must be defined as follows to access the include files referenced by the test programs:

```
$define BSLN user$disk:[robin.hughes.wbsln.baseline]
```

When the four files were compiled interactively, the two error-free versions generated correct listings. The first file (MTINHST_C.) generated an incomplete and incorrect listing file. The listing

- Z80/NSC800 C -

showed the include files inserted first, followed by "C", "8086" and a few other lines of the program. The output displayed on the screen looked like:

```
In pass1.
  70 else
    ^25
    136
    ^408
```

In C Nocode.

comp: C Nocode cannot recover from errors.

When the third file (MTOPNDF_C.) was compiled, the listing appeared fine except for the insertion a some strange control charaters.

These last two files were compiled in batch mode (file: user\$disk: {robin.hughes.rgalo.test}hughes.com). The first file (MTINHST_C.) generated a complete but incorrect listing. Although two errors were found (25 & 408) the line at the bottom stated that errors = 0. The include file expansion preceeded the "C" and "8086" in the listing, and lines like, #include filename, were still in the file. The error message was at line 72 of the listing instead of line 2472 were the '}' was actual missing. Finally the last 100 lines had useless numbers in the left margin.

When the third file (MTOPNDF_C.) was compiled, an incomplete listing was generated with the include file expansions listed first.

All of these tests were done on the VAX750 with the /e/v/o options.

This problem also occurs on the 68000.

Temporary solution:
No temporary solution available

Signed off 08/25/86 in release 301.80

Number: D200059071 Product: Z80/NSC800 C VAX 64824S003 01.50

One-line description:
Host compilers do not put absolute pats specifications in relocatables

Problem:
Host compilers do not specify the full path name in the relocatable file.

Temporary solution:
No known temporary solution.

Signed off 08/25/86 in release 301.80

Number: D200049064 Product: Z80/NSC800 C VAX 64824S003 00.00

One-line description:
Linker output file should use alternate file extension.

Signed off 08/25/86 in release 301.80

Number: 1650004630 Product: Z80/NSC800PASCAL 64823 01.01

One-line description:
Accessing parameter two nesting levels up is not working.

Problem:
The following program will generate code which uses the HL register pair before initializing them.

```
"BZ80"
$EXTENSIONS ON$
PROGRAM HLP AIR;
```

```
TYPE
  LENGTH = 0..5;

PROCEDURE ONE(LEN: LENGTH);
  PROCEDURE TWO;
    PROCEDURE THREE;

  VAR I: INTEGER;

  BEGIN
    FOR I:=0 TO LEN DO; /* CODE GENERATED USES HL W/O INIT.*/
      END;

  BEGIN {TWO}
    THREE;
  END; {TWO}
BEGIN {ONE}
  TWO;
END; {ONE}
.
```

This will only happen when the procedure is nested two levels. In other words, if the FOR statement was in PROCEDURE TWO the correct code is generated.

Temporary solution:
When nesting more than one level pass the upper level parameters to the lower level routines as parameters.

```
"BZ80"
$EXTENSIONS ON$
PROGRAM HLP FIX;
```

```
TYPE
  LENGTH = 0..5;

PROCEDURE ONE(LEN: LENGTH);
  PROCEDURE TWO(LEN: LENGTH);
  PROCEDURE THREE(LEN: LENGTH);
  VAR I: INTEGER;
  BEGIN
    FOR I:=0 TO LEN DO;
      END;
```

```
BEGIN { TWO }
  THREE(LEN);
END; { TWO }
```

```
BEGIN { ONE }
  TWO(LEN);
END; { ONE }
```

Signed off 08/25/86 in release 301.03

```
Number: 2700005371 Product: Z80/NSC800PASCAL 64823 00.00
```

Keywords: STRING ARRAYS

One-line description:
Multidimensional arrays of packed string arrays cannot be assigned to.

Problem:
"BZ80" or "B8085"
PROGRAM TEST;
TYPE STRING_40 = PACKED ARRAY [0..15] OF CHAR;
VAR ARRAY1 : ARRAY[1..2,1..2] OF STRING_40;

```
BEGIN
ARRAY1[1,1] := 'HELLO'
****Pass 2 error ?? 1006 => Contact HP
END.
```

Temporary solution:

No known work-around at this time.

Signed off 08/25/86 in release 301.03

```
Number: 5000103267 Product: Z80/NSC800PASCAL 64823 01.01
```

Keywords: SETS

One-line description:
SUPERSET or SUBSET checking doesn't work.

Problem:
TYPE SET_TYPE = SET OF (B0,B1,B2,B3,B4,B5,B6,B7);
VAR X : SET_TYPE;
BEGIN
IF X <= [B3,B4] THEN; {GENERATES INCORRECT CODE}
IF X >= [B3,B4] THEN; {GENERATES INCORRECT CODE}

Temporary solution:
None at this time.

Signed off 08/25/86 in release 301.03

```
Number: 5000109934 Product: Z80/NSC800PASCAL 64823 01.01
```

Keywords: RECURSIVE

One-line description:
FOR loops don't work with \$RECURSIVE +\$ and WITH.

Problem:
TYPE RECORDTYPE = RECORD
 FIELD1, FIELD2, FIELD3 : BYTE; END;
VAR VARTYPE = ARRAY [1..5] OF RECORDTYPE;
 J : BYTE;

```
PROCEDURE TEST;
BEGIN
WITH VARTYPE[J] DO
  FOR J := FIELD2 TO FIELD3 DO K := K + 1;
  {This doesn't work. For the pre-loop test, the L and A registers
  should be loaded before the call to Zsbytelt. The L register is
  not loaded.}
```

Temporary solution:

```
Instead of "WITH VARTYPE[J]" etc do
FOR J := VARTYPE[J].FIELD2 TO VARTYPE[J].FIELD3 etc
OR $RECURSIVE OFF$
```

Signed off 08/25/86 in release 301.03

```
Number: 5000115402 Product: Z80/NSC800PASCAL 64823 01.01
```

Keywords: FOR LOOP

One-line description:
FOR Signed8 := 0 TO 2 DO REAL1 := REAL1/REAL2 overwrites the A-register.

Temporary solution:
Use the compiler option \$AMNESIA +\$

Signed off 08/25/86 in release 301.03

```
Number: D200016329 Product: Z80/NSC800PASCAL 64823 01.01
```

Keywords: PASS 3

One-line description:
Pass 3 fails to detect relative jump address out-of-range.

Problem:
Pass 3 of the compilation process may fail to detect a relative jump which is out of range. In the test program submitted the relative jump is generated for an IF..THEN statement while the compiler option OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:
As a temporary work around disable the compiler option OPTIMIZE around those sections of code which are suspect.

Signed off 08/25/86 in release 301.03

Number: D200022467 Product: Z80/NSC800PASCAL 64823 01.01

Keywords: CODE GENERATOR

One-line description:

Incorrect code generated for IF statement.

Problem:

Compiling the following program demonstrates a code generation problem for the IF statement.

```
PROGRAM test;
$EXTENSIONS$
```

```
VAR
SCAN_TYPE : BYTE;
```

```
BEGIN
IF (SCAN_TYPE > 6) OR (SCAN_TYPE = 2) THEN
END.
```

After determining the result of (SCAN_TYPE > 6) the compiler overwrites the result (stored in the accumulator) with other data. Thus, the only comparison made is (SCAN_TYPE = 2).

Temporary solution:

Divide the IF statement into two separate statements.

Signed off 08/25/86 in release 301.03

Number: D200022525 Product: Z80/NSC800PASCAL 64823 01.01

Keywords: CODE GENERATOR

One-line description:

Incorrect code generated for SET inclusion statement.

Problem:

The following program demonstrates a code generation problem for the SET inclusion statement.

```
PROGRAM test;
$EXTENSIONS$
```

```
TYPE
BYTE_SET = SET OF (B0, B1, B2, B3, B4, B5, B6, B7);
```

```
VAR
status_byte : BYTE_SET;
```

```
BEGIN
IF [B0] <= status_byte THEN
END.
```

In the example listed, the compiler generates code which OR's and

- Z80/NSC800PASCAL -

CP's (compare) rather than an AND operation.

Temporary solution:

Use the set inclusion statement: IF B0 IN status_byte THEN ...

Signed off 08/25/86 in release 301.03

Number: D200026419 Product: Z80/NSC800PASCAL 64823 01.01

One-line description:

Defining TRUE and FALSE as global may result in duplicate symbol names.

Problem:

Defining the variables (constants) TRUE and FALSE to be global may result in a duplicate symbol error during a link. These variables are incorrectly defined as global in the Zwordcmp routine located in 'Zlibrary'.

NOTE: Redefining the values of TRUE and/or FALSE is not a legal Pascal operation. Redefinition of these constants is therefore not supported when using the HP 64000 compiler.

Temporary solution:

Obtain the source to Zwordcmp from your local HP Systems Engineer.

Signed off 08/25/86 in release 301.03

Number: D200028878 Product: Z80/NSC800PASCAL 64823 01.01

One-line description:

Incorrect code generated for WHILE construct.

Temporary solution:

There are two possible work-arounds for this problem:

- (1) alter the order of comparisons, or
- (2) change the TYPE of a to something other than SIGNED_16.

Signed off 08/25/86 in release 301.03

Number: D200034108 Product: Z80/NSC800PASCAL 64823 01.01

Keywords: STRING

One-line description:

Pointers to STRINGS cannot be assigned a string of length one.

Problem:

```
TYPE STR_ARR : PACKED ARRAY [0..7] OF CHAR; {I.E., A STRING}
ARR_PTR : ^STR_ARR;
```

```
VAR PTR : ARR_PTR;
```

```
BEGIN
```

```
.
```

- Z80/NSC800PASCAL -


```

PTR^ := "1234567"; {WORKS FINE}
PTR^ := "1";       {GENERATES THE FOLLOWING INCORRECT CODE}
LD A,001H        {THIS WILL BE THE STRING LENGTH}
LD HL,[PTR]
LD [HL], A       {SO FAR SO GOOD, WE'VE LOADED THE BYTE COUNT IN
                  STR ARR[0]}
LD HL,[PTR+001H]{THIS IS THE MISTAKE. WE SHOULD HAVE DONE A
                  LD HL,[PTR] INC HL}
LD [HL], 031H

```

Temporary solution:
None at this time.

Signed off 08/25/86 in release 301.03

Number: D200036806 Product: Z80/NSC800PASCAL 64823 01.01

Keywords: INCLUDE

One-line description:
Nested INCLUDE files 3 or more deep cause 64000 to "hang" in pass 3.

Problem:
Nested INCLUDE files 3 or more deep cause 64000 to hang in pass 3.

Temporary solution:
None at this time.

Signed off 08/25/86 in release 301.03

Number: D200047639 Product: Z80/NSC800PASCAL 64823 01.01

One-line description:
TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 301.03

Number: D200047944 Product: Z80/NSC800PASCAL 64823 01.01

One-line description:
Zcaseerror jumped to rather than called.

Problem:
If the following code is compiled, it is possible for Zcaseerror to be jumped to rather than called. By being jumped to, Zcaseerror doesn't have a return address.

```

"BZ80"
$DEBUG ON$
$RANGE ON$
PROGRAM TEST;

VAR Ch : CHAR;

BEGIN
  Ch := 'D'; /* LOAD ILLEGAL VALUE. */

```

- Z80/NSC800PASCAL -

```

CASE Ch OF
  'A' : Ch := PRED(Ch);
  'B' : Ch := PRED(Ch);
  'C' : Ch := PRED(Ch);
  'E' : Ch := PRED(Ch);
END.

```

The expanded code shows that Zcaseerror is jumped to rather than being called.

Signed off 08/25/86 in release 301.03

Number: D200048074 Product: Z80/NSC800PASCAL 64823 01.01

One-line description:
Level 3 recursive procedure or function causes Error 1008 - Stack Error.

Problem:
A pass 2 error 1008 occurs if a level 3 subroutine or function makes an assignment to a 16 bit variable defined by the level 2 parent procedure if the level 2 parent procedure is recursive. The following code causes 3 stack errors, error #1008:

```

"BZ80"
$EXTENSIONS ON$
PROGRAM X;
$RECURSIVE ON$
PROCEDURE Y;
VAR
  A : SIGNED_16;
  B : UNSIGNED_16;
  C : 0..257;

PROCEDURE Z;
BEGIN
  A := 3;
  B := UNSIGNED_16(5);
  C := 257;
END;
BEGIN
END;

```

Temporary solution:
Putting the main program in the same file as the recursive routine that causes the error 1008 may solve the problem.

Another possible solution is to insert a dummy main program

```

BEGIN
END.

```

In this case, the user must be aware of where the real main program is in order to run from the correct place.

Signed off 08/25/86 in release 301.03

- Z80/NSC800PASCAL -

Number: D200048116 Product: Z80/NSC800PASCAL 64823 01.01

One-line description:

Missing semicolon causes compiler to hang in Pass 1.

Problem:

The following code causes the 64000 to hang in pass 1. An error is generated on the hosts stating that parsing has stopped at a particular line number.

```
"BZ80"
PROGRAM MAIN;
TYPE
STRUCTURED= RECORD
    INT1:INTEGER;
    INT2:INTEGER;
END;
```

```
PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
I:=P1      <--This missing semicolon causes the problem
I:=-P1.2;
I:=-P2;
END;
```

```
BEGIN
END.
```

Temporary solution:

If the compiler hangs, look for a statement without a semicolon. On the 64000, the status line will show which line of code it stopped on. On the hosts, the error message generated indicates which line of code parsing stopped on.

Signed off 08/25/86 in release 301.03

Number: D200049890 Product: Z80/NSC800PASCAL 64823 01.02

One-line description:

Level 3 access of level 1 variables generates incorrect code.

Problem:

PROBLEM DESCRIPTION:

A Pascal Program in which variables declared at level 1 (procedures and functions) are referenced at level 3 (2nd level nested procedures and functions) will generate bad code. The following example illustrates.

```
"BZ80"
PROGRAM SCOPE;

    PROCEDURE LEVEL_1;
    VAR
        VAR1 : INTEGER ;

    PROCEDURE LEVEL_2 ;
```

- Z80/NSC800PASCAL -

```
    PROCEDURE LEVEL_3 ;
    BEGIN { LEVEL_3 }
VAR1 := 6 ;      (* bad code generated here *)
END; { LEVEL_3 }
```

```
    BEGIN { LEVEL_2 }
        LEVEL_3 ;
    END ; { LEVEL_2 }
```

```
    BEGIN { LEVEL_1 }
        LEVEL_2 ;
    END ; { LEVEL_1 }
```

```
BEGIN { MAIN PROG - LEVEL_0 }
    LEVEL_1 ;
END. { MAIN PROG - LEVEL_0 }
```

Temporary solution:

No known temporary solution.

Signed off 08/25/86 in release 301.03

Number: D200052241 Product: Z80/NSC800PASCAL 64823 01.02

One-line description:

Incorrect code generated when a CHAR is converted to an UNSIGNED_16.

Problem:

Incorrect code is generated when a CHAR variable is converted to an UNSIGNED_16. The following code is an example:

```
"processor name"
$EXTENSIONS ON$
$RECURSIVE OFF$
PROGRAM PASCALTEST;
TYPE
    BUG_TYPE = UNSIGNED_16; (*There is no problem if this is
                            SIGNED_16*)
```

```
PROCEDURE BUGGY(COUNT:BUG_TYPE);EXTERNAL;
```

```
FUNCTION OPEN:SIGNED_16;
```

```
VAR
```

```
    COUNT : BUG_TYPE;
```

```
    LEN: CHAR;
```

```
BEGIN
```

```
    (*THE NEXT TWO STATEMENTS GENERATE INCORRECT CODE*)
```

```
    COUNT := BUG_TYPE(LEN);
```

```
        (* LD A,001H *)
```

```
        (* LD [Dopen+00002H],A *)
```

```
        (* LD A,[Dopen+00004H] *)
```

```
        (* LD [Dopen+00003H],A *)
```

```
    BUGGY(BUG_TYPE(LEN));
```

```
        (* LD A,001H *)
```

```
        (* LD [Dopen+00005H],BC*)
```

```
        (* LD A,[Dopen+00004H] *)
```

```
        (* LD HL,[Dopen+00005H]*)
```

- Z80/NSC800PASCAL -

```
(* PUSH HL      *)
(* CALL BUGGY   *)
(* INC SP      *)
(* INC SP      *)
```

END;

Something very strange occurs when the same code is compiled with \$RECURSIVE ON\$. The statement BUGGY(BUG_TYPE(LEN)); generates the following code:

```
LD A,001H
LD [IX-11],A
LD [IX-10],WHAT???
```

```
LD A,[IX-5]
LD L,A
LD H,[IX-10]
PUSH HL
CALL BUGGY
INC SP
INC SP
```

Temporary solution:

No known temporary solution.

Signed off 08/25/86 in release 301.03

Number: D200052373 Product: Z80/NSC800PASCAL 300 64823S004 01.00

One-line description:

Incorrect code generated when a CHAR is converted to an UNSIGNED_16.

Problem:

Incorrect code is generated when a CHAR variable is converted to an UNSIGNED_16. The following code is an example:

```
"processor name"
$EXTENSIONS ON$
$RECURSIVE OFF$
PROGRAM PASCALTEST;
TYPE
    BUG_TYPE = UNSIGNED_16; (*There is no problem if this is
                           SIGNED_16*)
```

PROCEDURE BUGGY(COUNT:BUG_TYPE);EXTERNAL;

FUNCTION OPEN:SIGNED_16;

VAR

COUNT : BUG_TYPE;

LEN: CHAR;

BEGIN

(*THE NEXT TWO STATEMENTS GENERATE INCORRECT CODE*)

COUNT := BUG_TYPE(LEN);

```
(* LD A,001H      *)
(* LD [Dopen+00002H],A *)
(* LD A,[Dopen+00004H] *)
(* LD [Dopen+00003H],A *)
```

BUGGY(BUG_TYPE(LEN));

```
(* LD A,001H      *)
(* LD [Dopen+00005H],BC *)
(* LD A,[Dopen+00004H] *)
(* LD HL,[Dopen+00005H] *)
(* PUSH HL        *)
(* CALL BUGGY     *)
(* INC SP         *)
(* INC SP         *)
```

END;

Something very strange occurs when the same code is compiled with \$RECURSIVE ON\$. The statement BUGGY(BUG_TYPE(LEN)); generates the following code:

```
LD A,001H
LD [IX-11],A
LD [IX-10],WHAT???
```

```
LD A,[IX-5]
LD L,A
LD H,[IX-10]
PUSH HL
CALL BUGGY
INC SP
INC SP
```

Temporary solution:
No known temporary solution.

Signed off 08/25/86 in release 401.10

Number: D200052662 Product: Z80/NSC800PASCAL 300 64823S004 01.00

One-line description:
Missing semicolon causes compiler to hang in Pass 1.

Problem:
The following code causes the 64000 to hang in pass 1. An error is generated on the hosts stating that parsing has stopped at a particular line number.

```
"BZ80"
PROGRAM MAIN;
TYPE
STRUCTURED= RECORD
  INT1:INTEGER;
  INT2:INTEGER;
END;
```

```
PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
I:=P1      <--This missing semicolon causes the problem
I:=P1.2;
I:=P2;
END;
```

```
BEGIN
END.
```

Temporary solution:
If the compiler hangs, look for a statement without a semicolon. On the 64000, the status line will show which line of code it stopped on. On the hosts, the error message generated indicates which line of code parsing stopped on.

Signed off 08/25/86 in release 401.10

Number: D200053769 Product: Z80/NSC800PASCAL 300 64823S004 01.00

One-line description:
Accessing parameter two nesting levels up is not working.

Problem:
The following program will generate code which uses the HL register pair before initializing them.

```
"BZ80"
$EXTENSIONS ON$
PROGRAM HLP AIR;
TYPE
  LENGTH = 0..5;
```

- Z80/NSC800PASCAL 300 -

```
PROCEDURE ONE(LEN: LENGTH);
  PROCEDURE TWO;
  PROCEDURE THREE;

  VAR I: INTEGER;

  BEGIN
    FOR I:=0 TO LEN DO; /* CODE GENERATED USES HL W/O INIT.*/
    END;

  BEGIN {TWO}
    THREE;
  END; {TWO}
  BEGIN {ONE}
    TWO;
  END; {ONE}
.
```

This will only happen when the procedure is nested two levels. In other words, if the FOR statement was in PROCEDURE TWO the correct code is generated.

Temporary solution:
When nesting more than one level pass the upper level parameters to the lower level routines as parameters.

```
"BZ80"
$EXTENSIONS ON$
PROGRAM HLP AIR;
TYPE
  LENGTH = 0..5;

PROCEDURE ONE(LEN: LENGTH);
  PROCEDURE TWO(LEN: LENGTH);
  PROCEDURE THREE(LEN: LENGTH);
  VAR I: INTEGER;
  BEGIN
    FOR I:=0 TO LEN DO;
    END;

  BEGIN { TWO }
    THREE(LEN);
  END; { TWO }

  BEGIN { ONE }
    TWO(LEN);
  END; { ONE }
.
```

Signed off 08/25/86 in release 401.10

- Z80/NSC800PASCAL 300 -

SRB detail reports as of 08/25/86

Page: 289

Number: D200058859 Product: Z80/NSC800PASCAL 300 64823S004 01.00

Keywords: PREPROCESSOR

One-line description:

Preprocessor reports errors when symbols hp64000, vms or hpux w #if

Signed off 08/25/86 in release 401.10

Number: D200059253 Product: Z80/NSC800PASCAL 300 64823S004 01.00

One-line description:

Host compilers do not put absolute pats specifications in relocatables

Problem:

Host compilers do not specify the full path name in the relocatable file.

Temporary solution:

No known temporary solution.

Signed off 08/25/86 in release 401.10

Number: D200049049 Product: Z80/NSC800PASCAL 300 64823S004 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 401.10

- Z80/NSC800PASCAL 300 -

SRB detail reports as of 08/25/86

Page: 290

Number: D200016337 Product: Z80/NSC800PASCAL 500 64823S001 01.10

Keywords: PASS 3

One-line description:

Pass 3 fails to detect relative jump address out-of-range.

Problem:

Pass 3 of the compilation process may fail to detect a relative jump which is out of range. In the test program submitted the relative jump is generated for an IF..THEN statement while the compiler option OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:

As a temporary work around disable the compiler option OPTIMIZE around those sections of code which are suspect.

Signed off 08/25/86 in release 101.40

Number: D200020115 Product: Z80/NSC800PASCAL 500 64823S001 01.10

Keywords: STRING ARRAYS

One-line description:

Multidimensional arrays of packed string arrays cannot be assigned to.

Problem:

"BZ80" or "B8085"

PROGRAM TEST;

TYPE STRING_40 = PACKED ARRAY [0..15] OF CHAR;

VAR ARRAY1 : ARRAY[1..2,1..2] OF STRING_40;

BEGIN

ARRAY1[1,1] := 'HELLO'

****Pass 2 error ?? 1006 => Contact HP

END.

Temporary solution:

Put the assignment statement within a procedure and call the procedure when necessary. The array may be accessed by either global or local 004variables.

Signed off 08/25/86 in release 101.40

Number: D200022475 Product: Z80/NSC800PASCAL 500 64823S001 01.10

Keywords: CODE GENERATOR

One-line description:

Incorrect code generated for IF statement.

Problem:

Compiling the following program demonstrates a code generation problem for the IF statement.

```
PROGRAM test;
```

- Z80/NSC800PASCAL 500 -

\$EXTENSIONS\$

```

VAR
  SCAN_TYPE : BYTE;

BEGIN
  IF (SCAN_TYPE > 6) OR (SCAN_TYPE = 2) THEN
  END.

```

After determining the result of (SCAN_TYPE > 6) the compiler overwrites the result (stored in the accumulator) with other data. Thus, the only comparison made is (SCAN_TYPE = 2).

Temporary solution:

Divide the IF statement into two separate statements.

Signed off 08/25/86 in release 101.40

Number: D200022533 Product: Z80/NSC800PASCAL 500 64823S001 01.10

Keywords: CODE GENERATOR

One-line description:

Incorrect code generated for SET inclusion statement.

Problem:

The following program demonstrates a code generation problem for the SET inclusion statement.

```

PROGRAM test;
$EXTENSIONS$

```

```

TYPE
  BYTE_SET = SET OF (B0, B1, B2, B3, B4, B5, B6, B7);

```

```

VAR
  status_byte : BYTE_SET;

```

```

BEGIN
  IF [B0] <= status_byte THEN
  END.

```

In the example listed, the compiler generates code which OR's and CP's (compare) rather than an AND operation.

Temporary solution:

Use the set inclusion statement: IF B0 IN status_byte THEN ...

Signed off 08/25/86 in release 101.40

Number: D200026484 Product: Z80/NSC800PASCAL 500 64823S001 01.10

One-line description:

Defining TRUE and FALSE as global may result in duplicate symbol names.

Problem:

Defining the variables (constants) TRUE and FALSE to be global may result in a duplicate symbol error during a link. These variables

- Z80/NSC800PASCAL 500 -

are incorrectly defined as global in the Zwordcmp routine located in 'Zlibrary'.

NOTE: Redefining the values of TRUE and/or FALSE is not a legal Pascal operation. Redefinition of these constants is therefore not supported when using the HP 64000 compiler.

Temporary solution:

Obtain the source to Zwordcmp from your local HP Systems Engineer.

Signed off 08/25/86 in release 101.40

Number: D200027755 Product: Z80/NSC800PASCAL 500 64823S001 01.10

One-line description:

No form feed between the expanded listing and the cross reference table.

Problem:

During compilation, with XREF option on, the compiler does not provide a form feed (FF) in the listing file. The XREF starts on the same page as the end of the listing. Also, the page number says 535 when it should be page 2.

Temporary solution:

After compiling with the xref option, edit the expanded listing file and insert a "control L" before the beginning of the cross reference listing.

Signed off 08/25/86 in release 101.40

Number: D200028886 Product: Z80/NSC800PASCAL 500 64823S001 01.10

One-line description:

Incorrect code generated for WHILE construct.

Temporary solution:

There are two possible work-arounds for this problem:

- (1) alter the order of comparisons, or
- (2) change the TYPE of a to something other than SIGNED_16.

Signed off 08/25/86 in release 101.40

Number: D200034132 Product: Z80/NSC800PASCAL 500 64823S001 01.10

Keywords: STRING

One-line description:

Pointers to STRINGS cannot be assigned a string of length one.

Problem:

```

TYPE STR_ARR : PACKED ARRAY [0..7] OF CHAR; (I.E., A STRING)
  ARR_PTR : ^STR_ARR;

```

```

VAR PTR : ARR_PTR;

```

- Z80/NSC800PASCAL 500 -

BEGIN

```

PTR^ := "1234567"; {WORKS FINE}
PTR^ := "1"; {GENERATES THE FOLLOWING INCORRECT CODE}
LD A,001H {THIS WILL BE THE STRING LENGTH}
LD HL,[PTR]
LD [HL], A {SO FAR SO GOOD, WE'VE LOADED THE BYTE COUNT IN
STR_ARR[0]}
LD HL,[PTR+001H]{THIS IS THE MISTAKE. WE SHOULD HAVE DONE A
LD HL,[PTR] INC HL}
LD [HL], 031H

```

Temporary solution:
None at this time.

Signed off 08/25/86 in release 101.40

Number: D200037150 Product: Z80/NSC800PASCAL 500 64823S001 01.20

Keywords: PASS 3

One-line description:
Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:
Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect data being output to the list file. In selected cases, machine code will be incorrectly listed. For example, consider the following Pascal program.

```

$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;

VAR
  a, b : BOOLEAN;

PROCEDURE one;

BEGIN
  a := b;
END;

```

In the example listed above, the output file will denote machine code of the form FFFFC00001 for one of the generated assembly statements. The correct value should be C8000001. This problem is caused by an incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 101.40

- Z80/NSC800PASCAL 500 -

Number: D200040246 Product: Z80/NSC800PASCAL 500 64823S001 01.20

Keywords: SETS

One-line description:
SUPERSET or SUBSET checking doesn't work.

Problem:
TYPE SET_TYPE = SET OF (B0,B1,B2,B3,B4,B5,B6,B7);
VAR X : SET_TYPE;
BEGIN
IF X <= [B3,B4] THEN; {GENERATES INCORRECT CODE}
IF X >= [B3,B4] THEN; {GENERATES INCORRECT CODE}

Temporary solution:
None at this time.

Signed off 08/25/86 in release 101.40

Number: D200043851 Product: Z80/NSC800PASCAL 500 64823S001 01.20

Keywords: RECURSIVE

One-line description:
FOR loops don't work with \$RECURSIVE +\$ and WITH.

Problem:
TYPE RECORDTYPE = RECORD
FIELD1, FIELD2, FIELD3 : BYTE; END;
VAR VARTYPE = ARRAY [1..5] OF RECORDTYPE;
J : BYTE;

```

PROCEDURE TEST;
BEGIN
WITH VARTYPE[J] DO
  FOR J := FIELD2 TO FIELD3 DO K := K + 1;
  {This doesn't work. For the pre-loop test, the L and A registers
  should be loaded before the call to Zsbytelt. The L register is
  not loaded.}

```

Temporary solution:

Instead of "WITH VARTYPE[J]" etc do
FOR J := VARTYPE[J].FIELD2 TO VARTYPE[J].FIELD3 etc
OR \$RECURSIVE OFF\$

Signed off 08/25/86 in release 101.40

Number: D200044719 Product: Z80/NSC800PASCAL 500 64823S001 01.20

Keywords: FOR LOOP

One-line description:
FOR Signed8 := 0 TO 2 DO REAL1 := REAL1/REAL2 overwrites the A-register.

Temporary solution:
Use the compiler option \$AMNESIA +\$

- Z80/NSC800PASCAL 500 -

Signed off 08/25/86 in release 101.40

Number: D200047647 Product: Z80/NSC800PASCAL 500 64823S001 01.20

One-line description:

TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 101.40

Number: D200048090 Product: Z80/NSC800PASCAL 500 64823S001 01.20

One-line description:

Level 3 recursive procedure or function causes Error 1008 - Stack Error.

Problem:

A pass 2 error 1008 occurs if a level 3 subroutine or function makes an assignment to a 16 bit variable defined by the level 2 parent procedure if the level 2 parent procedure is recursive. The following code causes 3 stack errors, error #1008:

```
"BZ80"
$EXTENSIONS ON$
PROGRAM X;
$RECURSIVE ON$
PROCEDURE Y;
VAR
  A : SIGNED_16;
  B : UNSIGNED_16;
  C : 0..257;

PROCEDURE Z;
BEGIN
  A := 3;
  B := UNSIGNED_16(5);
  C := 257;
END;
BEGIN
END;
```

Temporary solution:

Putting the main program in the same file as the recursive routine that causes the error 1008 may solve the problem.

Another possible solution is to insert a dummy main program

```
BEGIN
END.
```

In this case, the user must be aware of where the real main program is in order to run from the correct place.

Signed off 08/25/86 in release 101.40

Number: D200052357 Product: Z80/NSC800PASCAL 500 64823S001 01.30

One-line description:

Incorrect code generated when a CHAR is converted to an UNSIGNED_16.

Problem:

Incorrect code is generated when a CHAR variable is converted to an UNSIGNED_16. The following code is an example:

```
"processor name"
$EXTENSIONS ON$
$RECURSIVE OFF$
PROGRAM PASCALTEST;
TYPE
  BUG_TYPE = UNSIGNED_16; (*There is no problem if this is
                           SIGNED_16*)
```

PROCEDURE BUGGY(COUNT:BUG_TYPE);EXTERNAL;

FUNCTION OPEN:SIGNED_16;

VAR

COUNT : BUG_TYPE;

LEN: CHAR;

BEGIN

(*THE NEXT TWO STATEMENTS GENERATE INCORRECT CODE*)

COUNT := BUG_TYPE(LEN);

```
(* LD A,001H *)
(* LD [Dopen+00002H],A *)
(* LD A,[Dopen+00004H] *)
(* LD [Dopen+00003H],A *)
```

BUGGY(BUG_TYPE(LEN));

```
(* LD A,001H *)
(* LD [Dopen+00005H],BC*)
(* LD A,[Dopen+00004H] *)
(* LD HL,[Dopen+00005H] *)
(* PUSH HL *)
(* CALL BUGGY *)
(* INC SP *)
(* INC SP *)
```

END;

Something very strange occurs when the same code is compiled with \$RECURSIVE ON\$. The statement BUGGY(BUG_TYPE(LEN)); generates the following code:

```
LD A,001H
LD [IX-11],A
LD [IX-10],WHAT???
```

```
LD A,[IX-5]
LD L,A
LD H,[IX-10]
PUSH HL
CALL BUGGY
INC SP
INC SP
```


Temporary solution:
No known temporary solution.

Signed off 08/25/86 in release 101.40

Number: D200052647 Product: Z80/NSC800PASCAL 500 64823S001 01.30

One-line description:
Missing semicolon causes compiler to hang in Pass 1.

Problem:
The following code causes the 64000 to hang in pass 1. An error is generated on the hosts stating that parsing has stopped at a particular line number.

```
"BZ80"
PROGRAM MAIN;
TYPE
STRUCTURED= RECORD
    INT1:INTEGER;
    INT2:INTEGER;
END;
```

```
PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
I:=P1      <--This missing semicolon causes the problem
I:=P1.2;
I:=P2;
END;

BEGIN
END.
```

Temporary solution:
If the compiler hangs, look for a statement without a semicolon. On the 64000, the status line will show which line of code it stopped on. On the hosts, the error message generated indicates which line of code parsing stopped on.

Signed off 08/25/86 in release 101.40

Number: D200053744 Product: Z80/NSC800PASCAL 500 64823S001 01.30

One-line description:
Accessing parameter two nesting levels up is not working.

Problem:
The following program will generate code which uses the HL register pair before initializing them.

```
"BZ80"
$EXTENSIONS ON$
PROGRAM HLPAIR;

TYPE
    LENGTH = 0..5;
```

```
PROCEDURE ONE(LEN: LENGTH);
    PROCEDURE TWO;
        PROCEDURE THREE;

    VAR I: INTEGER;

    BEGIN
        FOR I:=0 TO LEN DO;          /* CODE GENERATED USES HL W/O INIT.*/
        END;

    BEGIN {TWO}
        THREE;
    END; {TWO}
    BEGIN {ONE}
        TWO;
    END; {ONE}
    .
```

This will only happen when the procedure is nested two levels. In other words, if the FOR statement was in PROCEDURE TWO the correct code is generated.

Temporary solution:
When nesting more than one level pass the upper level parameters to the lower level routines as parameters.

```
"BZ80"
$EXTENSIONS ON$
PROGRAM HLPFIX;

TYPE
    LENGTH = 0..5;

PROCEDURE ONE(LEN: LENGTH);
    PROCEDURE TWO(LEN: LENGTH);
        PROCEDURE THREE(LEN: LENGTH);
            VAR I : INTEGER;
            BEGIN
                FOR I:=0 TO LEN DO;
            END;

    BEGIN { TWO }
        THREE(LEN);
    END; { TWO }

    BEGIN { ONE }
        TWO(LEN);
    END; { ONE }
    .
```

Signed off 08/25/86 in release 101.40

Number: D200058834 Product: Z80/NSC800PASCAL 500 64823S001 01.30

Keywords: PREPROCESSOR

One-line description:

Preprocessor reports errors when symbols hp64000, vms or hpux w #if

Signed off 08/25/86 in release 101.40

Number: D200059238 Product: Z80/NSC800PASCAL 500 64823S001 01.30

One-line description:

Host compilers do not put absolute pats specifications in relocatables

Problem:

Host compilers do not specify the full path name in the relocatable file.

Signed off 08/25/86 in release 101.40

Number: D200049023 Product: Z80/NSC800PASCAL 500 64823S001 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 101.40

Number: D200016345 Product: Z80/NSC800PASCAL VAX 64823S003 01.10

Keywords: PASS 3

One-line description:

Pass 3 fails to detect relative jump address out-of-range.

Problem:

Pass 3 of the compilation process may fail to detect a relative jump which is out of range. In the test program submitted the relative jump is generated for an IF..THEN statement while the compiler option OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:

As a temporary work around disable the compiler option OPTIMIZE around those sections of code which are suspect.

Signed off 08/25/86 in release 301.60

Number: D200020123 Product: Z80/NSC800PASCAL VAX 64823S003 01.10

Keywords: STRING ARRAYS

One-line description:

Multidimensional arrays of packed string arrays cannot be assigned to.

Problem:

```
"BZ80" or "B8085"
PROGRAM TEST;
TYPE STRING_40 = PACKED ARRAY [0..15] OF CHAR;
VAR ARRAY1 : ARRAY[1..2,1..2] OF STRING_40;
```

BEGIN

```
ARRAY1[1,1] := 'HELLO'
```

```
****Pass 2 error ?? 1006 => Contact HP
```

END.

Temporary solution:

Put the assignment statement within a procedure and call the procedure when necessary. The array may be accessed by either global or local 004variables.

Signed off 08/25/86 in release 301.60

Number: D200022483 Product: Z80/NSC800PASCAL VAX 64823S003 01.10

Keywords: CODE GENERATOR

One-line description:

Incorrect code generated for IF statement.

Problem:

Compiling the following program demonstrates a code generation problem for the IF statement.

```
PROGRAM test;
```

\$EXTENSIONS\$

```

VAR
  SCAN_TYPE : BYTE;

BEGIN
  IF (SCAN_TYPE > 6) OR (SCAN_TYPE = 2) THEN
  END.

```

After determining the result of (SCAN_TYPE > 6) the compiler overwrites the result (stored in the accumulator) with other data. Thus, the only comparison made is (SCAN_TYPE = 2).

Temporary solution:

Divide the IF statement into two separate statements.

Signed off 08/25/86 in release 301.60

Number: D200022541 Product: Z80/NSC800PASCAL VAX 64823S003 01.10

Keywords: CODE GENERATOR

One-line description:

Incorrect code generated for SET inclusion statement.

Problem:

The following program demonstrates a code generation problem for the SET inclusion statement.

```

PROGRAM test;
$EXTENSIONS$

TYPE
  BYTE_SET = SET OF (B0, B1, B2, B3, B4, B5, B6, B7);

VAR
  status_byte : BYTE_SET;

BEGIN
  IF [B0] <= status_byte THEN
  END.

```

In the example listed, the compiler generates code which OR's and CP's (compare) rather than an AND operation.

Temporary solution:

Use the set inclusion statement: IF B0 IN status_byte THEN ...

Signed off 08/25/86 in release 301.60

Number: D200026492 Product: Z80/NSC800PASCAL VAX 64823S003 01.10

One-line description:

Defining TRUE and FALSE as global may result in duplicate symbol names.

Problem:

Defining the variables (constants) TRUE and FALSE to be global may result in a duplicate symbol error during a link. These variables

- Z80/NSC800PASCAL VAX -

are incorrectly defined as global in the Zwordcmp routine located in 'Zlibrary'.

NOTE: Redefining the values of TRUE and/or FALSE is not a legal Pascal operation. Redefinition of these constants is therefore not supported when using the HP 64000 compiler.

Temporary solution:

Obtain the source to Zwordcmp from your local HP Systems Engineer.

Signed off 08/25/86 in release 301.60

Number: D200027763 Product: Z80/NSC800PASCAL VAX 64823S003 01.20

One-line description:

No form feed between the expanded listing and the cross reference table.

Problem:

During compilation, with XREF option on, the compiler does not provide a form feed (FF) in the listing file. The XREF starts on the same page as the end of the listing. Also, the page number says 535 when it should be page 2.

Temporary solution:

After compiling with the xref option, edit the expanded listing file and insert a "control L" before the beginning of the cross reference listing.

Signed off 08/25/86 in release 301.60

Number: D200028894 Product: Z80/NSC800PASCAL VAX 64823S003 01.20

One-line description:

Incorrect code generated for WHILE construct.

Temporary solution:

There are two possible work-arounds for this problem:

- (1) alter the order of comparisons, or
- (2) change the TYPE of a to something other than SIGNED_16.

Signed off 08/25/86 in release 301.60

Number: D200034140 Product: Z80/NSC800PASCAL VAX 64823S003 01.20

Keywords: STRING

One-line description:

Pointers to STRINGS cannot be assigned a string of length one.

Problem:

```

TYPE STR_ARR : PACKED ARRAY [0..7] OF CHAR; {I.E., A STRING}
  ARR_PTR : ^STR_ARR;

```

```

VAR PTR : ARR_PTR;

```

- Z80/NSC800PASCAL VAX -

BEGIN

```

PTR^ := "1234567"; {WORKS FINE}
PTR^ := "1"; {GENERATES THE FOLLOWING INCORRECT CODE}
LD A,001H {THIS WILL BE THE STRING LENGTH}
LD HL,[PTR]
LD [HL], A {SO FAR SO GOOD, WE'VE LOADED THE BYTE COUNT IN
STR_ARR[0]}
LD HL,[PTR+001H]{THIS IS THE MISTAKE. WE SHOULD HAVE DONE A
LD HL,[PTR] INC HL}
LD [HL], 031H

```

Temporary solution:
None at this time.

Signed off 08/25/86 in release 301.60

Number: D200037168 Product: Z80/NSC800PASCAL VAX 64823S003 01.20

Keywords: PASS 3

One-line description:
Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:
Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect data being output to the list file. In selected cases, machine code will be incorrectly listed. For example, consider the following Pascal program.

```

$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;

VAR
  a, b : BOOLEAN;

PROCEDURE one;

BEGIN
  a := b;
END;

```

In the example listed above, the output file will denote machine code of the form FFFFC00001 for one of the generated assembly statements. The correct value should be C8000001. This problem is caused by an incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 301.60

- Z80/NSC800PASCAL VAX -

Number: D200040253 Product: Z80/NSC800PASCAL VAX 64823S003 01.20

Keywords: SETS

One-line description:
SUPERSET or SUBSET checking doesn't work.

Problem:
TYPE SET TYPE = SET OF (B0,B1,B2,B3,B4,B5,B6,B7);
VAR X : SET_TYPE;
BEGIN
IF X <= [B3,B4] THEN; {GENERATES INCORRECT CODE}
IF X >= [B3,B4] THEN; {GENERATES INCORRECT CODE}

Temporary solution:
None at this time.

Signed off 08/25/86 in release 301.60

Number: D200043869 Product: Z80/NSC800PASCAL VAX 64823S003 01.20

Keywords: RECURSIVE

One-line description:
FOR loops don't work with \$RECURSIVE +\$ and WITH.

Problem:
TYPE RECORDTYPE = RECORD
FIELD1, FIELD2, FIELD3 : BYTE; END;
VAR VARTYPE = ARRAY [1..5] OF RECORDTYPE;
J : BYTE;

```

PROCEDURE TEST;
BEGIN
WITH VARTYPE[J] DO
  FOR J := FIELD2 TO FIELD3 DO K := K + 1;
  {This doesn't work. For the pre-loop test, the L and A registers
  should be loaded before the call to Zsbytelt. The L register is
  not loaded.}

```

Temporary solution:

Instead of "WITH VARTYPE[J]" etc do
FOR J := VARTYPE[J].FIELD2 TO VARTYPE[J].FIELD3 etc
OR \$RECURSIVE OFF\$

Signed off 08/25/86 in release 301.60

Number: D200044727 Product: Z80/NSC800PASCAL VAX 64823S003 01.20

Keywords: FOR LOOP

One-line description:
FOR Signed8 := 0 TO 2 DO REAL1 := REAL1/REAL2 overwrites the A-register.

Temporary solution:
Use the compiler option \$AMNESIA +\$

- Z80/NSC800PASCAL VAX -

Signed off 08/25/86 in release 301.60

Number: D200047654 Product: Z80/NSC800PASCAL VAX 64823S003 01.20

One-line description:
TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 301.60

Number: D200048108 Product: Z80/NSC800PASCAL VAX 64823S003 01.20

One-line description:
Level 3 recursive procedure or function causes Error 1008 - Stack Error.

Problem:
A pass 2 error 1008 occurs if a level 3 subroutine or function makes an assignment to a 16 bit variable defined by the level 2 parent procedure if the level 2 parent procedure is recursive. The following code causes 3 stack errors, error #1008:

```
"BZ80"
$EXTENSIONS ON$
PROGRAM X;
$RECURSIVE ON$
PROCEDURE Y;
VAR
  A : SIGNED_16;
  B : UNSIGNED_16;
  C : 0..257;
PROCEDURE Z;
BEGIN
  A := 3;
  B := UNSIGNED_16(5);
  C := 257;
END;
BEGIN
END;
.
```

Temporary solution:
Putting the main program in the same file as the recursive routine that causes the error 1008 may solve the problem.

Another possible solution is to insert a dummy main program

```
BEGIN
END.
```

In this case, the user must be aware of where the real main program is in order to run from the correct place.

Signed off 08/25/86 in release 301.60

Number: D200052365 Product: Z80/NSC800PASCAL VAX 64823S003 01.40

One-line description:
Incorrect code generated when a CHAR is converted to an UNSIGNED_16.

Problem:
Incorrect code is generated when a CHAR variable is converted to an UNSIGNED_16. The following code is an example:

```
"processor name"
$EXTENSIONS ON$
$RECURSIVE OFF$
PROGRAM PASCALTEST;
TYPE
  BUG_TYPE = UNSIGNED_16; (*There is no problem if this is
                           SIGNED_16*)
PROCEDURE BUGGY(COUNT:BUG_TYPE);EXTERNAL;
FUNCTION OPEN:SIGNED_16;
VAR
  COUNT : BUG_TYPE;
  LEN: CHAR;
BEGIN
  (*THE NEXT TWO STATEMENTS GENERATE INCORRECT CODE*)
  COUNT := BUG_TYPE(LEN);
  (* LD A,001H *)
  (* LD [Dopen+00002H],A *)
  (* LD A,[Dopen+00004H] *)
  (* LD [Dopen+00003H],A *)
  BUGGY(BUG_TYPE(LEN));
  (* LD A,001H *)
  (* LD [Dopen+00005H],BC*)
  (* LD A,[Dopen+00004H] *)
  (* LD HL,[Dopen+00005H]*)
  (* PUSH HL *)
  (* CALL BUGGY *)
  (* INC SP *)
  (* INC SP *)
END;
```

Something very strange occurs when the same code is compiled with \$RECURSIVE ON\$. The statement BUGGY(BUG_TYPE(LEN)); generates the following code:

```
LD A,001H
LD [IX-11],A
LD [IX-10],WHAT??
LD A,[IX-5]
LD L,A
LD H,[IX-10]
PUSH HL
CALL BUGGY
INC SP
INC SP
```

Temporary solution:
No known temporary solution.

Signed off 08/25/86 in release 301.60

Number: D200052654 Product: Z80/NSC800PASCAL VAX 64823S003 01.40

One-line description:
Missing semicolon causes compiler to hang in Pass 1.

Problem:
The following code causes the 64000 to hang in pass 1. An error is generated on the hosts stating that parsing has stopped at a particular line number.

```
"BZ80"
PROGRAM MAIN;
TYPE
STRUCTURED= RECORD
    INT1:INTEGER;
    INT2:INTEGER;
END;
```

```
PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
I:=P1      <--This missing semicolon causes the problem
I:=P1.2;
I:=P2;
END;
```

```
BEGIN
END.
```

Temporary solution:
If the compiler hangs, look for a statement without a semicolon. On the 64000, the status line will show which line of code it stopped on. On the hosts, the error message generated indicates which line of code parsing stopped on.

Signed off 08/25/86 in release 301.60

Number: D200053751 Product: Z80/NSC800PASCAL VAX 64823S003 01.40

One-line description:
Accessing parameter two nesting levels up is not working.

Problem:
The following program will generate code which uses the HL register pair before initializing them.

```
"BZ80"
$EXTENSIONS ON$
PROGRAM HLP AIR;
```

```
TYPE
LENGTH = 0..5;
```

- Z80/NSC800PASCAL VAX -

```
PROCEDURE ONE(LEN: LENGTH);
    PROCEDURE TWO;
        PROCEDURE THREE;

    VAR I: INTEGER;

    BEGIN
        FOR I:=0 TO LEN DO;          /* CODE GENERATED USES HL W/O INIT.*/
        END;

    BEGIN {TWO}
        THREE;
    END; {TWO}
    BEGIN {ONE}
        TWO;
    END; {ONE}
.
```

This will only happen when the procedure is nested two levels. In other words, if the FOR statement was in PROCEDURE TWO the correct code is generated.

Temporary solution:
When nesting more than one level pass the upper level parameters to the lower level routines as parameters.

```
"BZ80"
$EXTENSIONS ON$
PROGRAM HLP AIR;

TYPE
LENGTH = 0..5;

PROCEDURE ONE(LEN: LENGTH);
    PROCEDURE TWO(LEN: LENGTH);
        PROCEDURE THREE(LEN: LENGTH);
            VAR I : INTEGER;
            BEGIN
                FOR I:=0 TO LEN DO;
            END;

    BEGIN { TWO }
        THREE(LEN);
    END; { TWO }

    BEGIN { ONE }
        TWO(LEN);
    END; { ONE }
.
```

Signed off 08/25/86 in release 301.60

- Z80/NSC800PASCAL VAX -

SRB detail reports as of 08/25/86

Page: 309

Number: D200058842 Product: Z80/NSC800PASCAL VAX 64823S003 01.40

Keywords: PREPROCESSOR

One-line description:

Preprocessor reports errors when symbols hp64000, vms or hpux w #if

Signed off 08/25/86 in release 301.60

Number: D200059246 Product: Z80/NSC800PASCAL VAX 64823S003 01.40

One-line description:

Host compilers do not put absolute paths specifications in relocatables

Problem:

Host compilers do not specify the full path name in the relocatable file.

Temporary solution:

No known temporary solution.

Signed off 08/25/86 in release 301.60

Number: D200049031 Product: Z80/NSC800PASCAL VAX 64823S003 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 301.60

- Z80/NSC800PASCAL VAX -

SRB detail reports as of 08/25/86

Page: 310

Number: D200013979 Product: Z8000 C 64820 01.03

Keywords: PASS 1

One-line description:

No warning or error: taking the sizeof a struct var. not declared

Problem:

The compiler should generate an error in the following code.

```
"C"
"Z8001"
main () {
    int y;
    y = sizeof(struct x);
}
```

If x is not declared or is declared as anything other than a structure, the program compiles with no error messages or warnings. It stores as the size zero bytes.

Signed off 08/25/86 in release 001.05

Number: D200027722 Product: Z8000 C 64820 01.03

One-line description:

No form feed between the expanded listing and the cross reference table.

Problem:

During compilation, with XREF option on, the compiler does not provide a form feed (FF) in the listing file. The XREF starts on the same page as the end of the listing. Also, the page number says 535 when it should be page 2.

Temporary solution:

After compiling with the xref option, edit the expanded listing file and insert a "control L" before the beginning of the cross reference listing.

Signed off 08/25/86 in release 001.05

Number: D200031351 Product: Z8000 C 64820 01.03

One-line description:

++ and -- operators evaluated with improper precedence.

Problem:

According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

Example 1: array[index++] = 1;

Example 2: array[index] = 1;
index++;

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed

- Z8000 C -

to be used.

Temporary solution:

Separate the expression as shown in example 2.

Signed off 08/25/86 in release 001.05

Number: D200033167 Product: Z8000 C 64820 01.03

One-line description:

Comparing character to zero in while loop generates incorrect code.

Problem:

If you compare a character variable to zero in a while loop incorrect code is generated. The below code demonstrates the problem.

```
"C"
"6809"

proc()
{
    char timeout = 10;

    while(timeout--); /* Code generated here causes infinite loop.
}

```

The code generated for the while loop clears the A register and then compares the D register to -1. Therefore the condition is never met.

Temporary solution:

Declare the variable used in the test condition as an integer.

```
"C"
"6809"

proc()
{
    int timeout = 10;

    while (timeout--);
}

```

Signed off 08/25/86 in release 001.05

Number: D200040691 Product: Z8000 C 64820 01.03

Keywords: PASS 3

One-line description:

Pass 3 fails to detect relative jump address out-of-range.

Problem:

Pass 3 of the compilation process may fail to detect a relative jump which is out of range. In the test program submitted the relative jump is generated for an IF..THEN statement while the compiler option OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:

As a temporary work around disable the compiler option OPTIMIZE

- Z8000 C -

around those sections of code which are suspect.

Signed off 08/25/86 in release 001.05

Number: D200041251 Product: Z8000 C 64820 01.03

One-line description:

Problem with integer pointer in conditional statement.

Problem:

In the following example, two loads are performed, but no other code is generated to check for zero value.

```
"C"
"processor name"
#define NULL 0
fct(parm)
int *parm;
{
    if (parm - NULL)
        parm = 10;
}

```

Signed off 08/25/86 in release 001.05

Number: D200047548 Product: Z8000 C 64820 01.03

One-line description:

TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 001.05

- Z8000 C -

SRB detail reports as of 08/25/86

Page: 313

Number: D200051250 Product: Z8000 C 300 64820S004 01.00

One-line description:

++ and -- operators evaluated with improper precedence.

Problem:

According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

Example 1: array[index++] = 1;

Example 2: array[index] = 1;

index++;

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:

Separate the expression as shown in example 2.

Signed off 08/25/86 in release 401.10

Number: D200052274 Product: Z8000 C 300 64820S004 00.00

Keywords: CODE GENERATOR

One-line description:

Incorrect opcode "MOV A,ACC" allowed by our assembler

Problem:

The instruction "MOV A,ACC" was assemble and emulated by our products; however, the Intel 8051 goes into the weeds at this instruction. At first glance the machine code in the assembler listing appears valid (MOV A,ACC ->0000 E5E0), but the bottom of page 8-35 in Intel's microcontroller handbook states: *MOV A,ACC is not a valid instruction.

Neither our manuals nor AMD's user manual mention this instruction.

Signed off 08/25/86 in release 401.10

Number: D200058990 Product: Z8000 C 300 64820S004 01.00

One-line description:

Host compilers do not put absolute pats specifications in relocatables

Problem:

Host compilers do not specify the full path name in the relocatable file.

Temporary solution:

No known temporary solution.

Signed off 08/25/86 in release 401.10

- Z8000 C -

SRB detail reports as of 08/25/86

Page: 314

Number: D200048959 Product: Z8000 C 300 64820S004 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 401.10

- Z8000 C -

Number: D200029728 Product: Z8000 C 500 64820S001 01.10

One-line description:
Program compiles on 64K, not 9000. Pass 3 error generated.

Problem:
The file will compile if any one include file is commented out.

Signed off 08/25/86 in release 101.50

Number: D200031369 Product: Z8000 C 500 64820S001 01.10

One-line description:
++ and -- operators evaluated with improper precedence.

Problem:
According to Kernighan and Ritchie, page 43, the following expressions are equivalent:
Example 1: array[index++] = 1;
Example 2: array[index] = 1;
 index++;
However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:
Separate the expression as shown in example 2.

Signed off 08/25/86 in release 101.50

Number: D200033175 Product: Z8000 C 500 64820S001 01.10

One-line description:
Comparing character to zero in while loop generates incorrect code.

Problem:
If you compare a character variable to zero in a while loop incorrect code is generated. The below code demonstrates the problem.

"C"
"6809"

```
proc()
{
  char timeout = 10;

  while(timeout--); /* Code generated here causes infinite loop.
}

```

The code generated for the while loop clears the A register and then compares the D register to -1. Therefore the condition is never met.

Temporary solution:
Declare the variable used in the test condition as an integer.
"C"

- Z8000 C -

"6809"

```
proc()
{
  int timeout = 10;

  while (timeout--);
}

```

Signed off 08/25/86 in release 101.50

Number: D200037093 Product: Z8000 C 500 64820S001 01.20

Keywords: PASS 3

One-line description:
Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:
Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect data being output to the list file. In selected cases, machine code will be incorrectly listed. For example, consider the following Pascal program.

```
$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;

VAR
  a, b : BOOLEAN;

PROCEDURE one;

BEGIN
  a := b;
END;

```

In the example listed above, the output file will denote machine code of the form FFFFC00001 for one of the generated assembly statements. The correct value should be C8000001. This problem is caused by an incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 101.50

Number: D200040709 Product: Z8000 C 500 64820S001 01.20

Keywords: PASS 3

One-line description:
Pass 3 fails to detect relative jump address out-of-range.

Problem:
Pass 3 of the compilation process may fail to detect a relative jump

- Z8000 C -

which is out of range. In the test program submitted the relative jump is generated for an IF..THEN statement while the compiler option OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:

As a temporary work around disable the compiler option OPTIMIZE around those sections of code which are suspect.

Signed off 08/25/86 in release 101.50

Number: D200041269 Product: Z8000 C 500 64820S001 01.20

One-line description:

Problem with integer pointer in conditional statement.

Problem:

In the following example, two loads are performed, but no other code is generated to check for zero value.

```
"C"
"processor name"
#define NULL 0
fct(parm)
int *parm;
{
    if (parm - NULL)
        parm = 10;
}
```

Signed off 08/25/86 in release 101.50

Number: D200045930 Product: Z8000 C 500 64820S001 01.20

One-line description:

Title description is incorrect.

Signed off 08/25/86 in release 101.50

Number: D200047555 Product: Z8000 C 500 64820S001 01.20

One-line description:

TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 101.50

Number: D200049684 Product: Z8000 C 500 64820S001 00.00

One-line description:

NO CROSS REFERENCE TABLE IS GENERATED

Problem:

"C" COMPILERS DO NOT GENERATE A CROSS REFERENCE TABLE ON THE VAX.
 "C" COMPILERS DO NOT GENERATE A CROSS REFERENCE TABLE ON THE VAX.

Temporary solution:

- Z8000 C -

NONE KNOWN AT PRESENT
 NONE KNOWN AT PRESENT

Signed off 04/18/86 in release 101.50

Number: D200058974 Product: Z8000 C 500 64820S001 01.40

One-line description:

Host compilers do not put absolute pats specifications in relocatables

Problem:

Host compilers do not specify the full path name in the relocatable file.

Temporary solution:

No known temporary solution.

Signed off 08/25/86 in release 101.50

Number: D200048934 Product: Z8000 C 500 64820S001 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 101.50

- Z8000 C -

Number: D200031377 Product: Z8000 C VAX 64820S003 01.20

One-line description:

++ and -- operators evaluated with improper precedence.

Problem:

According to Kernighan and Ritchie, page 43, the following expressions are equivalent:

Example 1: array[index++] = 1;

Example 2: array[index] = 1;

index++;

However, different code is generated for these expressions. The second example is compiled correctly, but the first one increments index before setting array[index] equal to 1. Furthermore, when these statements are executed in a main program, an uninitialized and unknown variable, Dmain, is used to index into array when the variable index is supposed to be used.

Temporary solution:

Separate the expression as shown in example 2.

Signed off 08/25/86 in release 301.80

Number: D200033183 Product: Z8000 C VAX 64820S003 01.20

One-line description:

Comparing character to zero in while loop generates incorrect code.

Problem:

If you compare a character variable to zero in a while loop incorrect code is generated. The below code demonstrates the problem.

"C"
"6809"

```
proc()
{
    char timeout = 10;
    while(timeout--); /* Code generated here causes infinite loop.
}
```

The code generated for the while loop clears the A register and then compares the D register to -1. Therefore the condition is never met.

Temporary solution:

Declare the variable used in the test condition as an integer.

"C"
"6809"

```
proc()
{
    int timeout = 10;
    while (timeout--);
}
```

Signed off 08/25/86 in release 301.80

- Z8000 C -

Number: D200037101 Product: Z8000 C VAX 64820S003 01.20

Keywords: PASS 3

One-line description:

Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:

Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect data being output to the list file. In selected cases, machine code will be incorrectly listed. For example, consider the following Pascal program.

```
$EXTENSIONS ON$
$LIST OBJ ON$
PROGRAM test;
```

```
VAR
```

```
    a, b : BOOLEAN;
```

```
PROCEDURE one;
```

```
BEGIN
```

```
    a := b;
```

```
END;
```

In the example listed above, the output file will denote machine code of the form FFFFC00001 for one of the generated assembly statements. The correct value should be C8000001. This problem is caused by an incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 301.80

Number: D200040717 Product: Z8000 C VAX 64820S003 01.20

Keywords: PASS 3

One-line description:

Pass 3 fails to detect relative jump address out-of-range.

Problem:

Pass 3 of the compilation process may fail to detect a relative jump which is out of range. In the test program submitted the relative jump is generated for an IF..THEN statement while the compiler option OPTIMIZE is enabled. [BLINK_TAS:BUG]

Temporary solution:

As a temporary work around disable the compiler option OPTIMIZE around those sections of code which are suspect.

Signed off 08/25/86 in release 301.80

- Z8000 C -

Number: D200041277 Product: Z8000 C VAX 64820S003 01.20

One-line description:

Problem with integer pointer in conditional statement.

Problem:

In the following example, two loads are performed, but no other code is generated to check for zero value.

```
"C"
"processor name"
#define NULL 0
fct(parm)
int *parm;
{
  if (parm - NULL)
    parm = 10;
}
```

Signed off 08/25/86 in release 301.80

Number: D200045948 Product: Z8000 C VAX 64820S003 01.20

One-line description:

Title description is incorrect.

Signed off 08/25/86 in release 301.80

Number: D200047563 Product: Z8000 C VAX 64820S003 01.20

One-line description:

TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 301.80

Number: D200055145 Product: Z8000 C VAX 64820S003 01.50

One-line description:

Compilation on the VAX using batch mode generates incorrect listing file

Problem:

The test files can be found on the VAX750 under user\$disk:[robin.hughes.rgalo.test]. The following test files were used:

1. MTINHST_C. - File which contains one error- a missing '}' on line 70
2. TMTINHST_C. - Error-free version of MTINHST_C.
3. MTOPNDF_C. - File which contains one error - missing declaration for integer 'j'
4. MTOPNDF_C. - Error-free version of MTOPNDF_C.

One logical name must be defined as follows to access the include files referenced by the test programs:

```
$define BSLN user$disk:[robin.hughes.wsbsln.baseline]
```

When the four files were compiled interactively, the two error-free versions generated correct listings. The first file (MTINHST_C.) generated an incomplete and incorrect listing file. The listing showed the include files inserted first, followed by "C", "8086" and a few other lines of the program. The output displayed on the screen looked like:

```
In pass1.
  70 else ^25
        136 ^408
In C Nocode.
comp: C Nocode cannot recover from errors.
```

When the third file (MTOPNDF_C.) was compiled, the listing appeared fine except for the insertion a some strange control charaters.

These last two files were compiled in batch mode (file: user\$disk:[robin.hughes.rgalo.test]hughes.com). The first file (MTINHST_C.) generated a complete but incorrect listing. Although two errors were found (25 & 408) the line at the bottom stated that errors = 0. The include file expansion preceeded the "C" and "8086" in the listing, and lines like, #include filename, were still in the file. The error message was at line 72 of the listing instead of line 2472 were the '}' was actual missing. Finally the last 100 lines had useless numbers in the left margin.

When the third file (MTOPNDF_C.) was compiled, an incomplete listing was generated with the include file expansions listed first.

All of these tests were done on the VAX750 with the /e/v/o options.

This problem also occurs on the 68000.

Temporary solution:

No temporary solution available

Signed off 08/25/86 in release 301.80

Number: D200058982 Product: Z8000 C VAX 64820S003 01.50

One-line description:

Host compilers do not put absolute pats specifications in relocatables

Problem:

Host compilers do not specify the full path name in the relocatable file.

Signed off 08/25/86 in release 301.80

Number: D200048942 Product: Z8000 C VAX 64820S003 00.00

One-line description:
Linker output file should use alternate file extension.

Signed off 08/25/86 in release 301.80

Number: D200036798 Product: Z8000 PASCAL 64816 01.09

Keywords: INCLUDE

One-line description:
Nested INCLUDE files 3 or more deep cause 64000 to "hang" in pass 3.

Problem:
Nested INCLUDE files 3 or more deep cause 64000 to hang in pass 3.

Temporary solution:
None at this time.

Signed off 08/25/86 in release 601.11

Number: D200047456 Product: Z8000 PASCAL 64816 01.09

One-line description:
TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 601.11

Number: D200052605 Product: Z8000 PASCAL 64816 01.10

One-line description:
Missing semicolon causes compiler to hang in Pass 1.

Problem:
The following code causes the 64000 to hang in pass 1. An error is generated on the hosts stating that parsing has stopped at a particular line number.

```
"processor name"
PROGRAM MAIN;
TYPE
STRUCTURED= RECORD
    INT1:INTEGER;
    INT2:INTEGER;
END;

PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
I:=P1    <--This missing semicolon causes the problem
I:=P1,2;
I:=P2;
END;

BEGIN
END.
```

Temporary solution:
If the compiler hangs, look for a statement without a semicolon. On the 64000, the status line will show which line of code it stopped on. On the hosts, the error message generated indicates which line of code parsing stopped on.

One-line description:
Missing semicolon causes compiler to hang in Pass 1.

Problem:
The following code causes the 64000 to hang in pass 1. An error is generated on the hosts stating that parsing has stopped at a particular line number.

```
"processor name"  
PROGRAM MAIN;  
TYPE  
STRUCTURED= RECORD  
    |   INT1:INTEGER;  
    |   INT2:INTEGER;  
    |   END;  
  
PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);  
VAR I:INTEGER;  
BEGIN  
I:=P1      <--This missing semicolon causes the problem  
I:=P1.2;  
I:=P2;  
END;  
  
BEGIN  
END.
```

Temporary solution:
If the compiler hangs, look for a statement without a semicolon. On the 64000, the status line will show which line of code it stopped on. On the hosts, the error message generated indicates which line of code parsing stopped on.

Signed off 08/25/86 in release 401.10

Number: D200058826 Product: Z8000 PASCAL 300 64816S004 01.00

Keywords: PREPROCESSOR

One-line description:
Preprocessor reports errors when symbols hp64000, vms or hpux w #if

Signed off 08/25/86 in release 401.10

Number: D200048868 Product: Z8000 PASCAL 300 64816S004 00.00

One-line description:
Linker output file should use alternate file extension.

Signed off 08/25/86 in release 401.10

Number: D200027680 Product: Z8000 PASCAL 500 64816S001 01.10

One-line description:

No form feed between the expanded listing and the cross reference table.

Problem:

During compilation, with XREF option on, the compiler does not provide a form feed (FF) in the listing file. The XREF starts on the same page as the end of the listing. Also, the page number says 535 when it should be page 2.

Temporary solution:

After compiling with the xref option, edit the expanded listing file and insert a "control L" before the beginning of the cross reference listing.

Signed off 08/25/86 in release 101.40

Number: D200037036 Product: Z8000 PASCAL 500 64816S001 01.20

Keywords: PASS 3

One-line description:

Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:

Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect data being output to the list file. In selected cases, machine code will be incorrectly listed. For example, consider the following Pascal program.

```
$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;

VAR
  a, b : BOOLEAN;

PROCEDURE one;

  BEGIN
    a := b;
  END;
```

In the example listed above, the output file will denote machine code of the form FFFFC00001 for one of the generated assembly statements. The correct value should be C8000001. This problem is caused by an incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 101.40

- Z8000 PASCAL -

Number: D200047464 Product: Z8000 PASCAL 500 64816S001 01.20

One-line description:

TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 101.40

Number: D200052613 Product: Z8000 PASCAL 500 64816S001 01.30

One-line description:

Missing semicolon causes compiler to hang in Pass 1.

Problem:

The following code causes the 64000 to hang in pass 1. An error is generated on the hosts stating that parsing has stopped at a particular line number.

```
"processor name"
PROGRAM MAIN;
TYPE
  STRUCTURED= RECORD
    INT1:INTEGER;
    INT2:INTEGER;
  END;

PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
  I:=P1      <--This missing semicolon causes the problem
  I:=P1.2;
  I:=P2;
  END;

BEGIN
  END.
```

Temporary solution:

If the compiler hangs, look for a statement without a semicolon. On the 64000, the status line will show which line of code it stopped on. On the hosts, the error message generated indicates which line of code parsing stopped on.

Signed off 08/25/86 in release 101.40

Number: D200058800 Product: Z8000 PASCAL 500 64816S001 01.30

Keywords: PREPROCESSOR

One-line description:

Preprocessor reports errors when symbols hp64000, vms or hpux w #if

Signed off 08/25/86 in release 101.40

- Z8000 PASCAL -

SRB detail reports as of 08/25/86

Page: 329

Number: D200048843 Product: Z8000 PASCAL 500 64816S001 00.00

One-line description:

Linker output file should use alternate file extension.

Signed off 08/25/86 in release 101.40

- Z8000 PASCAL -

SRB detail reports as of 08/25/86

Page: 330

Number: D200027698 Product: Z8000 PASCAL VAX 64816S003 01.20

One-line description:

No form feed between the expanded listing and the cross reference table.

Problem:

During compilation, with XREF option on, the compiler does not provide a form feed (FF) in the listing file. The XREF starts on the same page as the end of the listing. Also, the page number says 535 when it should be page 2.

Temporary solution:

After compiling with the xref option, edit the expanded listing file and insert a "control L" before the beginning of the cross reference listing.

Signed off 08/25/86 in release 301.60

Number: D200037044 Product: Z8000 PASCAL VAX 64816S003 01.20

Keywords: PASS 3

One-line description:

Compiler option \$LIST_OBJ ON\$ generates wrong output information.

Problem:

Use of the compiler option \$LIST_OBJ ON\$ may result in incorrect data being output to the list file. In selected cases, machine code will be incorrectly listed. For example, consider the following Pascal program.

```
$EXTENSIONS ON$
$LIST_OBJ ON$
PROGRAM test;

  VAR
    a, b : BOOLEAN;

  PROCEDURE one;

    BEGIN
      a := b;
    END;
```

In the example listed above, the output file will denote machine code of the form FFFFC00001 for one of the generated assembly statements. The correct value should be C8000001. This problem is caused by an incorrect "printf" mask when generating the output file.

NOTE: THIS DEFECT IS ONLY PRESENT IN THE GENERATED LISTING FILE.
THE GENERATED CODE IS CORRECT.

Signed off 08/25/86 in release 301.60

- Z8000 PASCAL -

Number: D200047472 Product: Z8000 PASCAL VAX 64816S003 01.20

One-line description:
TOO MANY ERRORS IN PASS 3 IF >127 PROCEDURES

Signed off 08/25/86 in release 301.60

Number: D200052621 Product: Z8000 PASCAL VAX 64816S003 01.30

One-line description:
Missing semicolon causes compiler to hang in Pass 1.

Problem:
The following code causes the 64000 to hang in pass 1. An error is generated on the hosts stating that parsing has stopped at a particular line number.

```
"processor name"
PROGRAM MAIN;
TYPE
STRUCTURED= RECORD
    INT1:INTEGER;
    INT2:INTEGER;
    END;
```

```
PROCEDURE OUTER(VAR P1:STRUCTURED; VAR P2:INTEGER);
VAR I:INTEGER;
BEGIN
I:=P1      <--This missing semicolon causes the problem
I:=P1.2;
I:=P2;
END;

BEGIN
END.
```

Temporary solution:
If the compiler hangs, look for a statement without a semicolon. On the 64000, the status line will show which line of code it stopped on. On the hosts, the error message generated indicates which line of code parsing stopped on.

Signed off 08/25/86 in release 301.60

Number: D200058818 Product: Z8000 PASCAL VAX 64816S003 01.30

Keywords: PREPROCESSOR

One-line description:
Preprocessor reports errors when symbols hp64000, vms or hpux w #if

Signed off 08/25/86 in release 301.60

Number: D200048850 Product: Z8000 PASCAL VAX 64816S003 00.00

One-line description:
Linker output file should use alternate file extension.

Signed off 08/25/86 in release 301.60

SRB detail reports as of 08/25/86

Page: 333

Number: D200043398 Product: Z80H EMULATION 64253 01.00

One-line description:
Error in guided softkey syntax.

Problem:
The guided syntax softkeys yeild incorrect sytax in one peculiar case.
The sequence that gives the problem is [trace] [after] [address] [not]
0400H then the softkey options are [and] [status] [occurs] [only]
[counting] [break on]. The 'and' is the problem. It should read
'data'. 'and' yeilds incorrect syntax. If you type 'data' it works.

Signed off 08/25/86 in release 301.02

Number: 5000118414 Product: Z80H EMULATION 64253 01.00

One-line description:
modify memory word to VALUE has bytes reversed from Z80 point of view

Signed off 08/25/86 in release 301.02



**5958-6019, October 1986
E1086**

Printed in U.S.A.