HEWLETT
PACKARD

HEWLETT-PACKARD COMPANY
LOGIC SYSTEMS DIVISION

HP 64000 Logic Development System

SYSTEM RELEASE BULLETIN

Part Number: **5958-6019**
E1287

Printed: **DEC. 1987**

HP STARS II

64000 SOFTWARE RELEASE BULLETIN

Issue 88.02

FEBRUARY, 1988

This document supersedes all previously dated SSBs.

HEWLETT
PACKARD

READER COMMENT SHEET

STARS II SRB (STARS B)

Issue _____._____ DATE _____/_____/_____

We welcome your evaluation of this bulletin. Your comments and suggestions help us to improve our publications. Please use additional pages if necessary.

Is this bulletin technically accurate?	Yes [ ] No [ ]	(If no, explain under Comments, below.)
Are the concepts and wording easy to understand?	Yes [ ] No [ ]	(If no, explain under Comments, below.)
Is the format of this bulletin convenient in size, arrangement and readability?	Yes [ ] No [ ]	(If no, explain or suggest improvements under Comments, below.)

Comments:

****

Date: ___________

FROM:

Name ______________________________

Company ______________________________

Address ______________________________

____

____

STARS B
Printed in U.S.A.

FOLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1303 COLORADO SPRINGS, CO.

POSTAGE WILL BE PAID BY ADDRESSEE

STARS Administration
Hewlett-Packard Company
Logic Systems Division
P.O. BOX 617
Colorado Springs, Colorado 80901-0617

FOLD FOLD

P R E F A C E

This Software Release Bulletin documents all fixes and enhancements that are incorporated in the new release identified on the cover page. The SRB is provided as a benefit of Hewlett-Packard's Account Management Support, Response Center Support, and Software Materials Subscription.

Of the five sections contained in the SRB (not including the PREFACE), only the last section which contains the detailed reports has page numbers. These are referenced by the product, report number and keyword indexes in order to direct the user to a particular area or to an individual detailed report. The five sections are described below.

SOFTWARE RELEASE CONTENTS

This section lists the product names, numbers and update/fix levels of all products contained in this release. Products that have changed, or are new are denoted with an asterisk preceding the product name.

PRODUCT INDEX

Each unique product name/number has an entry listing the page number where the detailed report for that product begins.

REPORT NUMBER INDEX

This index is a sequential list of the individual report numbers with the corresponding page number where the report can be found.

KEYWORD INDEX

This index is sorted by product name, keyword, product number (including the update/fix level) and by report number in that order. In addition to the sort items, each entry has a brief description (one line) and the page number where the detailed report can be found. Note that a given report can be listed more than once in this section if it has more than one keyword assigned to it.

DETAILED REPORTS

Each report contains all the available information relevant to the problem being corrected or the enhancement being implemented.

Software release contents

Product name		Product number	uu.ff
*64000-UX OP-ENV	300	64801S004	01.80
*6800 C		64821	02.10
*6800 C	300	64821S004	02.10
*6800 C	500	64821S001	02.10
*6800 C	VAX	64821S003	02.10
*6800 PASCAL		64811	01.90
*6800 PASCAL	300	64811S004	01.90
*6800 PASCAL	500	64811S001	01.90
*6800 PASCAL	VAX	64811S003	01.90
*68000 ASSEMB		64845	02.10
*68000 ASSEMB	300	64845S004	02.10
*68000 ASSEMB	500	64845S001	02.10
*68000 ASSEMB	VAX	64845S003	02.10
*68000 C		64819	02.10
*68000 C	300	64819S004	02.10
*68000 C	500	64819S001	02.10
*68000 C	VAX	64819S003	02.10
*68000 PASCAL		64815	01.90
*68000 PASCAL	300	64815S004	01.90
*68000 PASCAL	500	64815S001	01.90
*68000 PASCAL	VAX	64815S003	01.90
*68000C AXLS COMP	300	64902S004	01.00
*6809 C		64822	01.80
*6809 C	300	64822S004	01.80
*6809 C	500	64822S001	01.80
*6809 C	VAX	64822S003	01.80
*6809 PASCAL		64813	01.60
*6809 PASCAL	300	64813S004	01.60
*6809 PASCAL	500	64813S001	01.60
*6809 PASCAL	VAX	64813S003	01.60
*78310/12 ASSEMB		64866	01.02
*80286B ASSEMB		64859	01.30
*80286B ASSEMB	300	64859S004	01.30
*80286B ASSEMB	500	64859S001	01.30
*80286B ASSEMB	VAX	64859S003	01.30
*8085 B PASCAL		64825	01.90
*8085 B PASCAL	300	64825S004	01.90
*8085 B PASCAL	500	64825S001	01.90
*8085 B PASCAL	VAX	64825S003	01.90
*8085 C		64826	02.10
*8085 C	300	64826S004	02.10
*8085 C	500	64826S001	02.10
*8085 C	VAX	64826S003	02.10
*8085 EMULATION		64203	01.07
*8086/8 ASSEMB		64853	02.70
*8086/8 ASSEMB	300	64853S004	02.70
*8086/8 ASSEMB	500	64853S001	02.70
*8086/8 ASSEMB	VAX	64853S003	02.70
*8086/8 C		64818	03.70
*8086/8 C	300	64818S004	03.70
*8086/8 C	500	64818S001	03.70
*8086/8 C	VAX	64818S003	03.70
*8086/8 PASCAL		64814	03.50
*8086/8 PASCAL	300	64814S004	03.50
*8086/8 PASCAL	500	64814S001	03.50

Software release contents

Product name		Product number	uu.ff
*8086/8 PASCAL	VAX	64814S003	03.50
*9900/0 ASSEMB		64847	01.80
*9900/0 ASSEMB	300	64847S004	01.80
*9900/0 ASSEMB	500	64847S001	01.80
*9900/0 ASSEMB	VAX	64847S003	01.80
*F9450 EMULATION		64286	01.05
*HOST SOFTWARE /	VAX	64882	02.30
*HOST SOFTWARE /	300	64883	01.10
*HOST SOFTWARE /	500	64880	01.80
*MS1750A ASSEMB		64857	01.90
*MS1750A ASSEMB	300	64857S004	01.90
*MS1750A ASSEMB	500	64857S001	01.90
*MS1750A ASSEMB	VAX	64857S003	01.90
*OPERATING SYSTEM		64100	02.10
*RS-232 TRANSFER	300	64885	01.30
*RS-232 TRANSFER	500	64884	01.30
*RS-232 TRANSFER	VAX	64886	01.50
*TMS 32010 MODULES		64285	01.02
*USER DEF ASSEMB	300	64851S004	02.10
*USER DEF ASSEMB	300	64861S004	02.10
*USER DEF ASSEMB	500	64851S001	02.10
*USER DEF ASSEMB	500	64861S001	02.10
*USER DEF ASSEMB	VAX	64851S003	02.10
*USER DEF ASSEMB	VAX	64861S003	02.10
*USER DEF EMULATION		64274	01.06
*USER INTERFACE	300	64808S004	02.10
*USER INTERFACE	500	64808S001	02.10
*Z80/NSC800 C		64824	02.10
*Z80/NSC800 C	300	64824S004	02.10
*Z80/NSC800 C	500	64824S001	02.10
*Z80/NSC800 C	VAX	64824S003	02.10
*Z80/NSC800PASCAL		64823	01.90
*Z80/NSC800PASCAL	300	64823S004	01.90
*Z80/NSC800PASCAL	500	64823S001	01.90
*Z80/NSC800PASCAL	VAX	64823S003	01.90
*Z8000 C		64820	02.10
*Z8000 C	300	64820S004	02.10
*Z8000 C	500	64820S001	02.10
*Z8000 C	VAX	64820S003	02.10
*Z8000 PASCAL		64816	01.90
*Z8000 PASCAL	300	64816S004	01.90
*Z8000 PASCAL	500	64816S001	01.90
*Z8000 PASCAL	VAX	64816S003	01.90
*Z8002 EMULATION		64233	02.01

Product index

Report number index

Report #	page	Report #	page	Report #	page	Report #	page
1650004598	50	5000170654	124	D200014340	23	D200071423	140
1650004705	84	5000171124	20	D200015073	9	D200071431	128
1650006908	112	5000171876	89	D200015347	9	D200072371	76
1650008409	8	5000171884	89	D200019265	111	D200073031	141
1650009456	26	5000171900	90	D200019273	92	D200074237	77
1650011585	133	5000172239	64	D200022137	92	D200075044	128
1650013235	50	5000172593	51	D200023283	92	D200075861	15
1650018689	84	5000172742	43	D200025858	71	D200075911	34
1650018721	110	5000172825	124	D200027953	114	D200075952	100
1650019224	26	5000173278	125	D200031476	72	D200076026	149
1650019257	108	5000181552	113	D200035220	105	D200076067	142
1650019331	27	5000182014	135	D200038778	127	D200076109	40
1650026583	20	5000185066	1	D200042606	72	D200076588	2
1650026708	56	5000186718	65	D200043828	120	D200076620	2
1650033209	112	5000186742	135	D200046623	120	D200076679	3
1650036525	121	5000188839	21	D200049916	73	D200076760	146
1650038281	1	5000189985	113	D200053710	93	D200076802	11
1650038430	82	5000190629	136	D200057802	74	D200076844	31
1650038521	121	5000193466	65	D200060509	52	D200076919	45
2700005520	56	5000195628	67	D200061556	19	D200077008	3
5000103432	85	5000197624	90	D200062604	114	D200077065	24
5000111666	112	5000201012	51	D200063123	14	D200077107	146
5000116848	14	5000201749	68	D200063719	34	D200077149	11
5000118844	85	5000202770	113	D200063750	94	D200077180	31
5000121830	105	5000202846	43	D200063792	28	D200077222	129
5000124313	86	5000203596	69	D200063834	149	D200077263	46
5000134593	57	5000204586	22	D200063875	137	D200077396	79
5000134601	57	5000206649	22	D200063909	39	D200077495	121
5000134817	87	5000207845	90	D200064212	39	D200077727	79
5000136085	50	5000216036	70	D200065060	94	D200077768	52
5000136267	58	5000217927	136	D200066357	107	D200077875	101
5000137869	153	5000219204	110	D200066761	137	D200077941	3
5000138388	87	5000220186	43	D200067629	10	D200077958	4
5000149229	59	5000221788	83	D200068403	44	D200078642	101
5000149757	60	5000221994	91	D200068684	74	D200078923	7
5000149765	61	5000223099	23	D200068759	96	D200079038	5
5000151431	153	5000223792	116	D200068767	98	D200079079	129
5000152918	49	5000223800	70	D200068825	38	D200079087	46
5000161000	133	5000224022	106	D200068957	107	D200079095	5
5000161034	134	5000224204	145	D200069104	108	D200079111	80
5000162487	62	5000226605	132	D200069666	24	D200079129	25
5000163287	134	5000226613	54	D200070532	74	D200079137	146
5000163410	63	5000229476	71	D200070615	75	D200079145	11
5000163824	88	5000231605	125	D200071332	138	D200079152	31
5000164004	106	5000232959	105	D200071340	139	D200079160	47
5000165134	63	5000233866	126	D200071365	139	D200079178	16
5000169474	1	5000234237	23	D200071373	127	D200079186	35
5000170415	51	5000241562	119	D200071415	154	D200079194	103

Report number index

Report #	page	Report #	page	Report #	page	Report #	page
D200079202	28	D200079285	42	D200080135	122	D200081141	122
D200079210	150	D200079293	5	D200080333	81	D200081471	130
D200079228	41	D200079301	143	D200080341	147	D200081620	117
D200079236	17	D200079335	55	D200080358	12	D200081638	118
D200079244	36	D200079509	6	D200080366	32	D200081984	6
D200079251	104	D200080051	81	D200080374	130	D200082636	109
D200079269	29	D200080093	6	D200080382	47	D200082669	108
D200079277	151	D200080101	7	D200080721	122	D200085050	115

Keyword index

- -4

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64801S004	01.00	No error message given when opt run on memory board.	5000169474	1
	64801S004	01.00	Edbuild does not work whether invoked by the user or the emulator.	5000185066	1
	64801S004	01.00	Msinit may autoconfigure incorrectly.	D200077941	3
	64801S004	01.00	HP enhancements are too slow to run softkeys.	D200077958	4
	64801S004	01.50	If second card cage used in pv, it may not be released	D200076588	2
	64801S004	01.50	Pressing return during a cycle command causes pv to hang	D200076620	2
	64801S004	01.50	Msinit crashes opt test user.	D200076679	3
	64801S004	01.50	If /usr/hp64000/lock does not exist, msconfig gives odd messages.	D200077008	3
	64801S004	01.60	pwd truncates the /net/system portion of the path when RFA'ed to system.	1650038281	1
	64801S004	01.60	Search command files via "PATH" variable	D200078923	7
	64801S004	01.60	option_test multitest can't handle more than 24 cards	D200079038	5
	64801S004	01.60	EDBUILD IS NOT WORKING PROPERLY	D200079095	5
	64801S004	01.60	Incompatible /etc/update programs.	D200079509	6
	64801S004	01.60	64000-UX processes do not die when modem carrier is lost.	D200080093	6
	64801S004	01.60	Processes sometimes left running after parent has stopped.	D200081984	6
	64801S004	01.70	Multiple <cr> after selecting card will leave softkeyw in an odd state.	D200079293	5
ENHANCEMENT	64801S004	01.60	64000-UX uses TERMINFO database only partially.	D200080101	7

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64821	01.06	Assigning 0 to bitfield causes entire structure to be reset.	D200067629	10
	64821	01.07	Array is being placed in the PROG section rather than data.	D200076802	11
	64821	01.07	Warning message text is incorrect.	D200080358	12
CODE GENERATOR	64821	00.01	Nested IF stmnt. with bit field structure members genrate incorr. code	1650008409	8
	64821	01.04	Using a pointer dereference as an array index generates incorrect code.	D200015073	9
	64821	01.04	Compiler doesn't reload register after value changes in a nested if expr	D200015347	9
	64821	01.07	Floating point division of 2 constants generates incorrect result	D200077149	11
PASS 1	64821	01.07	DIV, MOD and COMParisons may do unsigned estend of signed values	D200079145	11

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64811	01.10	functional type change of a constant into multi-byte structure gen's err	D200063123	14
	64811	01.20	Unsigned_8 treated as signed value in FOR loop test.	D200075861	15
	64811	01.20	Pascal does not report error for assignment of constant to structure	D200079178	16
PASS 1	64811	01.20	Functional type changes not always evaluated correctly	D200079236	17
WITH	64811	01.08	Generates bad code for parameter as ADDR of component.	5000116848	14

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64845	01.10	EQU is not working with DC correctly.	D200061556	19

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64819	01.02	68008 libraries cause target processor disagree errors.	5000234237	23
	64819	01.09	Result is invalid if terenary expression evaluates to false.	5000171124	20
	64819	01.09	Two external declarations are made for one function.	5000206649	22

Keyword index

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64819	01.09	Compiler aborts when it encounters a legal structure declaration.	D200069666	24
	64819	01.10	Invalid error 60 flagged.	1650026583	20
	64819	01.10	$LIST ON/OFF$ doesn't work correctly.	5000188839	21
	64819	01.10	Wrong floating point value assigned in a terenary expression.	5000204586	22
CODE GENERATOR	64819	01.07	Bad code using $RANGE$ or $DEBUG$ with $CALL_PC_LONG$ or LIB_PC_LONG	D200014340	23
	64819	01.10	Floating point division of 2 constants generates incorrect result	D200077065	24
ENHANCEMENT	64819	01.90	Generate XREF from compiler which is readable by EDT.	5000223099	23
PASS 1	64819	01.10	DIV, MOD and COMParisons may do unsigned estend of signed values	D200079129	25

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64815	00.01	System reboot for syntax error.	1650009456	26
	64815	01.10	Incorrect code generated for array which has boolean indices.	1650019224	26
	64815	01.10	Set manipulation results in incorrect code being generated.	1650019331	27
	64815	01.11	functional type change of a constant into multi-byte structure gen's err	D200063792	28
	64815	01.12	Pascal does not report error for assignment of constant to structure	D200079202	28
PASS 1	64815	01.12	Functional type changes not always evaluated correctly	D200079269	29

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64822	01.08	Array is being placed in the PROG section rather than data.	D200076844	31
	64822	01.08	Warning message text is incorrect.	D200080366	32
CODE GENERATOR	64822	01.08	Floating point division of 2 constants generates incorrect result	D200077180	31
PASS 1	64822	01.08	DIV, MOD and COMParisons may do unsigned estend of signed values	D200079152	31

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64813	01.10	functional type change of a constant into multi-byte structure gen's err	D200063719	34
	64813	01.11	Unsigned_8 treated as signed value in FOR loop test.	D200075911	34
	64813	01.11	Pascal does not report error for assignment of constant to structure	D200079186	35
PASS 1	64813	01.11	Functional type changes not always evaluated correctly	D200079244	36

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64859	01.02	Assembling on 64100 & linking on VAX generates erroneous absolute file	D200068825	38

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64825	01.03	functional type change of a constant into multi-byte structure gen's err	D200063909	39
	64825	01.04	Unsigned_8 treated as signed value in FOR loop test.	D200076109	40
	64825	01.04	Pascal does not report error for assignment of constant to structure	D200079228	41
PASS 1	64825	01.04	Functional type changes not always evaluated correctly	D200079285	42
PASS 2	64825	01.03	Incorrect code generated when set elements are passed as parameters.	D200064212	39

Keyword index

- -0

Keyword	Product number	uu.ff Description	Report #	page
********none********	64826	01.03 Expression used as array index generates incorrect code.	D200068403	44
	64826	01.04 Array is being placed in the PROG section rather than data.	D200076919	45
	64826	01.04 Warning message text is incorrect.	D200080382	47
CODE GENERATOR	64826	01.03 bad code generated with *pointer++ operation	5000172742	43
	64826	01.04 > = does not work with float type	5000202846	43
	64826	01.04 When subtract an integer from a pointer, get unnecessary warning message	5000220186	43
	64826	01.04 Floating point division of 2 constants generates incorrect result	D200077263	46
	64826	01.04 +=, -=, *=, & /= may fail to auto vars with $RECURSIVE ON$	D200079087	46
PASS 1	64826	01.04 DIV, MOD and COMParisons may do unsigned estend of signed values	D200079160	47

- -0

Keyword	Product number	uu.ff Description	Report #	page
********none********	64203	01.06 Can't use 9.5" paper to print mem map, due to centering of printout.	5000152918	49

- -0

Keyword	Product number	uu.ff Description	Report #	page
********none********	64853	02.00 Assembler/linker does not correctly handle EQU <EXT_LABEL> statement.	5000136085	50
	64853	02.01 Wrong values during EQU from externals.	1650004598	50
	64853	02.01 reusing accumulator	D200077768	52
	64853	02.03 Tabs in source file are expanded to 6 spaces instead of 8 spaces	1650013235	50
	64853	02.03 The noload files aren't showing up in the listing; absolute correct	5000170415	51
CODE GENERATOR	64853	02.02 Incorrect Object code generated	5000201012	51
LINKER	64853	02.01 Linker generates error if COMN segment is not 0000H	D200060509	52
	64853	02.03 Label preceeded with WORD PTR,NEAR PTR, etc. will not appear in the xref	5000172593	51

- -0

Keyword	Product number	uu.ff Description	Report #	page
LINKER	64853S004	02.20 Linker can generate invalid DATE on listing file.	5000226613	54

- -0

Keyword	Product number	uu.ff Description	Report #	page
CODE GENERATOR	64853S003	02.50 VMS Hosted linker does not recognize logical names	D200079335	55

- -0

Keyword	Product number	uu.ff Description	Report #	page
********none********	64818	02.00 Using a postfix decrement operator in a conditional statement fails.	D200031476	72
	64818	02.00 Compiler uses wrong segment register.	D200042606	72
	64818	03.00 DX register is used although it is overwritten by IMUL instruction	D200049916	73
	64818	03.01 ES reg overwritten when assign char array to complex data structure	5000195628	67
	64818	03.02 Code generated for illegal C statement - POP BH generated	5000149757	60
	64818	03.02 Address of array element incorrectly calculated	5000149765	61
	64818	03.02 external used 2x in same pgm, w/ ASM_FILE ON, get 2 EXT stmts in ASMfile	5000172239	64
	64818	03.02 PROGRAMS WITH DUPLICATE GOTO LABELS MAY FAIL IN PASS 3	D200074237	77
	64818	03.02 Warning message text is incorrect.	D200080333	81

Keyword index

- -0

Keyword	Product number	uu.ff	Description	Report #	page
CODE GENERATOR	64818	01.06	Argument to switch statement may be doubled.	D200025858	71
	64818	02.01	1102 error generated - register needed but not availiable	5000134593	57
	64818	02.01	Incorrect segment of array transfered to pointer	5000134601	57
	64818	02.01	ES register corrupt when used to get address of array to place in ptr.	5000136267	58
	64818	03.00	Return statement not putting value on BX register	5000149229	59
	64818	03.00	Nonsense code generated by dynamic struc declaration in a funct.	D200057802	74
	64818	03.01	Right shift using var. for # of places to shift generates bad code	1650026708	56
	64818	03.01	Floating point division of 2 constants generates incorrect result	5000186718	65
	64818	03.01	When using calculated value for array index, uses BX register twice	5000193466	65
	64818	03.01	Compile incorrect when a ptr to an int is casted as a short and incremen	D200068684	74
	64818	03.01	Incorrect segment passed to external function	D200070615	75
	64818	03.01	Assignment to ptr var. (w/ Separate_const off) causes corrupt stack	D200072371	76
	64818	03.02	Vax not creating same code as the 64000	5000162487	62
	64818	03.02	BX register overwritten with a switch statement	5000165134	63
	64818	03.02	Bad code generated for address of external character if for loop	5000223800	70
	64818	03.02	Incorrect segment passed to external function	D200070532	74
	64818	03.02	VAX and 64100 generate different constants. VAX is incorrect.	D200077396	79
	64818	03.02	CL register being used twice	D200077727	79
	64818	03.02	Cannot prevent adding Esymbol and Rsymbol info to global symbol table	D200080051	81
	64818	03.20	compiler reusing CX register	5000201749	68
	64818	03.30	Problem w/ unreleased Rev. Dx Register destroyed	5000203596	69
LINKER	64818	03.10	The noload files aren't showing up in the listing, absolute correct	5000216036	70
PASS 1	64818	03.02	compiler using DS segment rather than ES segment for 32 bit pointers	5000163410	63
	64818	03.02	DIV, MOD and COMParisons may do unsigned estend of signed values	D200079111	80
RUN-TIME LIBRARY	64818	00.00	REAL NUMBER COMPARISONS MAY NOT EVALUATE CORRECTLY.	2700005520	56
SF1001	64818	03.02	When casting unsigned ints to floats using +=, generates a error #1001	5000229476	71

- -0

Keyword	Product number	uu.ff	Description	Report #	page
CODE GENERATOR	64818S004	03.02	printer arithmetic gives warning "integer not pointer size"	1650038430	82

- -0

Keyword	Product number	uu.ff	Description	Report #	page
CODE GENERATOR	64818S001	03.30	bade code gen if local ptr to extrnl strcture is assgn vlu frm extrn ary	5000221788	83

- -0

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64814	02.01	Incorrect code generated in FOR loop.	5000103432	85
	64814	02.01	The library routine, DISPOSE, overwrites the ES register	5000124313	86
	64814	03.01	functional type change of a constant into multi-byte structure gen's err	D200063750	94
	64814	03.02	WITH construct causes wrong offset	D200068759	96
	64814	03.02	With construct causes wrong offset	D200068767	98
	64814	03.02	Unsigned 8 treated as signed value in FOR loop test.	D200075952	100
	64814	03.02	$RECURSIVE $ option defaults to incorrect mode (OFF)	D200077875	101
	64814	03.02	Pascal does not report error for assignment of constant to structure	D200079194	103
CODE GENERATOR	64814	00.01	Stack POP'S exceed Stack PUSH'S when assignment made to ext var.	1650018689	84
	64814	01.10	Illegal PUSH instruction generated.	D200023283	92
	64814	02.00	Wrong code generated for expression in 'FOR' loop	5000118844	85

Keyword index

- -0

Keyword	Product number	uu.ff Description	Report #	page
CODE GENERATOR	64814	02.01 Using ES register without initalization - REP MOVSB.	1650004705	84
	64814	02.01 Incorrect address calculated for beginning of ary in WITH stamnt	5000134817	87
	64814	03.00 Incorrect code gener. when shift function operand is mult. dimen. array	5000138388	87
	64814	03.00 bad code for accessing parameters in nested procedures	5000207845	90
	64814	03.00 Var. addresses incorrect inside nested WITH statements	D200053710	93
	64814	03.00 pointers passed as procedure parameters not fully dereferenced.	D200078642	101
	64814	03.01 Multiplication result stored in CX and overwritten when counter reg need	5000163824	88
	64814	03.01 Code produces an #1102 error - reg. needed but not available	5000171876	89
	64814	03.01 BX register gets overwritten when accessing arrays of records	5000171884	89
	64814	03.01 Contents of register A gets overwritten when accessing mult. arrys of rd	5000171900	90
	64814	03.01 FOR loop counter gets destroyed when loop includes multiple WITH's	D200065060	94
	64814	03.03 Using WITH statement and complex record structure causes bad code	5000221994	91
PASCAL	64814	03.01 for loop w/ counter = unsignd_8 type uses BX twice	5000197624	90
PASS 1	64814	03.02 Functional type changes not always evaluated correctly	D200079251	104
RUN-TIME LIBRARY	64814	01.10 Problem with Pascal I/O library (PIOLIB).	D200019273	92
	64814	01.10 Real number comparisons may not 'evaluate' correctly.	D200022137	92

- -9

Keyword	Product number	uu.ff Description	Report #	page
********none********	64847	00.46 In macros, "" and '' are not equivalent.	D200035220	105
CODE GENERATOR	64847	00.00 Problem with negative displacementwith SBO SBZ instructions.	5000232959	105
	64847	00.46 Autoincrement with indirect addressing does not assemble correctly.	5000121830	105

- -9

Keyword	Product number	uu.ff Description	Report #	page
********none********	64286	01.04 Answer to "Emulate SCR?" is automatically changed to "yes"	5000224022	106
ENHANCEMENT	64286	01.03 Cannot single step through a Software Breakpoint	5000164004	106
	64286	01.03 MASK, STATUS, and IC are not always cleared when running from reset.	D200066357	107
	64286	01.04 Enhance the register display to show the Pending Interrupt Register	D200068957	107

- -0

Keyword	Product number	uu.ff Description	Report #	page
********none********	64882	01.20 VAX help on MAPBUS command causes system error	1650019257	108
	64882	01.70 64100 cluster disk free list is corrupted so there is not enough space	D200069104	108
TRANSFER	64882	02.00 CSIB process does not come up on system bootup	D200082669	108

- -0

Keyword	Product number	uu.ff Description	Report #	page
********none********	64883	01.00 Transfer does not handle imbedded linefeeds in binary files.	D200082636	109

- -0

Keyword	Product number	uu.ff Description	Report #	page
********none********	64880	01.10 Invalid file names are not detected by the transfer utility.	D200019265	111
	64880	01.60 Debug file transfers may not function with 14 character file names.	1650018721	110

Keyword index

- -O

Keyword	Product number	uu.ff Description	Report #	page
TRANSFER	64880	01.06 Bad file format does not cause error in transfer.	5000219204	110

- -P

Keyword	Product number	uu.ff Description	Report #	page
********none********	64100	02.01 Formatting a floppy from a command file sometimes is unsuccessful.	5000111666	112
	64100	02.04 Comment is taken as a parameter when a null parameter is passed.	D200062604	114
	64100	02.06 Logical operators generate MO error.	5000202770	113
	64100	02.07 Unique label is flagged as undefined in macro expansion.	1650033209	112
	64100	02.07 Instructions assembling differently than previous assembler.	5000189985	113
	64100	02.07 Phase error incorrectly reported on 64000 and hosted assemblers.	D200085050	115
COPY	64100	02.01 "copy f:link_com to display" doesn't display all attributes of "f".	D200027953	114
DC600	64100	00.01 64000 backup from 7946 to 9144 with 150' tape produces wrong message.	1650006908	112
	64100	02.04 No multi-tape backup strategy for discs > 64MB on the 64000.	5000181552	113

- -M

Keyword	Product number	uu.ff Description	Report #	page
********none********	64285	01.00 Inverse assembly for 1E91H is incorrectly shown as SUB *-, E, 0	5000223792	116

- -S

Keyword	Product number	uu.ff Description	Report #	page
********none********	64851S001	01.60 expressions of the form 123456.78 cause errors	D200081620	117

- -S

Keyword	Product number	uu.ff Description	Report #	page
********none********	64851S003	01.60 expressions of form 123456.78 cause errors	D200081638	118

- -S

Keyword	Product number	uu.ff Description	Report #	page
********none********	64274	01.04 Bad display of trace data with 8-bit UDE	D200043828	120
	64274	01.04 modify memory starting at odd addresses does not always work	D200046623	120
	64274	01.05 run until <addr> fails from reset when reset points to user code.	5000241562	119

- -S

Keyword	Product number	uu.ff Description	Report #	page
********none********	64808S004	01.20 Pmon flags a syntax error when attempting to append files	1650036525	121
	64808S004	01.20 User softkey display should be erased after shell escape.	D200077495	121
	64808S004	01.20 pwd truncates the /net/system portion of the path when RFA'ed to system.	D200080135	122
	64808S004	01.20 Pmon command completion intermittantly fails after completion error.	D200080721	122
	64808S004	01.20 Command search algorithm should match the softkey package.	D200081141	122
MENUS	64808S004	02.00 Pmon command completion via shell variables not working correctly	1650038521	121

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64824	01.01	Incorrect transfer address when linking 9 or more files.	D200038778	127
	64824	01.03	Error using switch (*x++).	5000170654	124
	64824	01.03	RETI is not generated when exiting an interrupt procedure.	5000172825	124
	64824	01.03	Array is being placed in the PROG section rather than data.	5000173278	125
	64824	01.03	INT Multiplication of short by negative constant with SHORT_ARITH.	D200071373	127
	64824	01.04	+= operator does not work for pointers to structures.	5000231605	125
	64824	01.04	ZDconvert library module has two errors in the ZDdwordtoword routine.	5000233866	126
	64824	01.04	.	D200075044	128
	64824	01.04	+=, -=, *=, & /= may fail to auto vars with $RECURSIVE ON$	D200079079	129
	64824	01.04	Warning message text is incorrect.	D200080374	130
	64824	01.04	MOD operation returning the wrong value.	D200081471	130
CODE GENERATOR	64824	01.04	Floating point division of 2 constants generates incorrect result	D200077222	129
PASS 1	64824	01.03	DIV, MOD and COMParisons may do unsigned estend of signed values	D200071431	128

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64824S004	01.20	Double word divide library returning incorrect result.	5000226605	132

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64823	01.03	Unbelieveable amount of library code linked for no-line program.	5000161000	133
	64823	01.03	Libraries reference procedures not actually needed.	5000161034	134
	64823	01.03	FOR statement with SIGNED_BYTE produces incorrect code.	5000182014	135
	64823	01.03	functional type change of a constant into multi-byte structure gen's err	D200063875	137
	64823	01.03	Code generated by compiler increased 12% with latest version.	D200066761	137
	64823	01.03	Links not correctly established during calls of nested procedures.	D200071332	138
	64823	01.03	Certain variable accesses by nested procedures may not work	D200071340	139
	64823	01.03	Function Calls via pointer may fail	D200071423	140
	64823	01.03	Pascal does not report error for assignment of constant to structure	D200073031	141
	64823	01.04	Code generated by compiler increased 12% with latest version.	5000163287	134
	64823	01.04	Bad code generated when CASE expression involves addition of two bytes.	5000190629	136
	64823	01.04	Signed_32 divide returns wrong result.	5000217927	136
	64823	01.04	Unsigned_8 treated as signed value in FOR loop test.	D200076067	142
	64823	01.04	Compiler may confuse similar parameters in different subroutines	D200079301	143
ADDR	64823	01.03	ADDR(x) generates incorrect code is x is of type BYTE.	5000186742	135
PASS 1	64823	01.03	Functional type changes not always evaluated correctly	D200071365	139
PASS 2	64823	01.01	Incorrect code generated when set elements are passed as parameters.	1650011585	133

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64823S003	01.70	Nested external procedure call causes bad code to be generated.	5000224204	145

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64820	01.06	Array is being placed in the PROG section rather than data.	D200076760	146
	64820	01.06	Warning message text is incorrect.	D200080341	147

- -8

Keyword	Product number	uu.ff	Description	Report #	page
CODE GENERATOR	64820	01.06	Floating point division of 2 constants generates incorrect result	D200077107	146
PASS 1	64820	01.03	DIV, MOD and COMParisons may do unsigned estend of signed values	D200079137	146

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64816	01.11	functional type change of a constant into multi-byte structure gen's err	D200063834	149
	64816	01.12	Unsigned_8 treated as signed value in FOR loop test.	D200076026	149
	64816	01.12	Pascal does not report error for assignment of constant to structure	D200079210	150
PASS 1	64816	01.12	Functional type changes not always evaluated correctly	D200079277	151

- -8

Keyword	Product number	uu.ff	Description	Report #	page
********none********	64233	01.07	Monitor displays wrong value for R15 and SSTK	5000137869	153
	64233	02.00	User interrupts are not serviced for 17ms after analysis generated break	5000151431	153
	64233	02.00	Can't find symbols loaded with more address bits than specified.	D200071415	154

Number: 1650038281 Product: 64000-UX OP-ENV 300 64801S004 01.60

One-line description:
pwd truncates the /net/system portion of the path when RFA'ed to system.

Problem:
When using the HP 64000-UX products and netunaming across the LAN to another system, such as a compile server, the HP-UX command "pwd" which is used by the HP64000-UX product to tell what the local directory is, truncates the "/net/system" part of the path.

This is a HP-UX operating system defect. It is not a defect in the HP 64000-UX application software. As soon as this defect is fixed in HP-UX, it will work correctly when using the HP 64000-UX applications.

Signed off 01/14/88 in release Z01.80

Number: 5000169474 Product: 64000-UX OP-ENV 300 64801S004 01.00

One-line description:
No error message given when opt run on memory board.

Problem:
64000 UX Options Test.
No error or warning is given if a user tries to do options test on a memory board in the 64120 cardcage. The test for memory must be done through the memory controler board. The 64000 (classic) gives an error

ERROR: No test availabe for selected card.

The 64000-UX gives file not found error and the file
    /usr/hp64000/inst/pv/pv01f8 (for the 128k memory board). It looks for the board id number and puts "pv" in front of it for the file name it looks for. The user has no clue as to what is wrong. He only knows that file pv01f8 does not exist.

Temporary solution:
There are no performance verification tests available for memory boards. All memory boards are tested when the memory controller is tested. Do not select memory boards for testing within opt.

Signed off 01/14/88 in release Z01.80

Number: 5000185066 Product: 64000-UX OP-ENV 300 64801S004 01.00

One-line description:
Edbuild does not work whether invoked by the user or the emulator.

Problem:
About the EDBUILD Command fails.

The customer system can not get response for the edbuild command, then a "load" command (in 8086 emulator mode) do not, either.

-- THE CUSTOMER SYSTEM CONFIGURATION --

```
    CPU - HP9000/320
          ( The swap area = 10 Mbytes
            HP-UX rev.5.172 )

  --------------------------------------

    An absolute file name is sys.X ( sys.L, sys.K ).
    The program (sys.X) is large of 5_Mbytes on source.
COMMANDS:
    edbuild sys, edbuild -h2 sys, edbuild -f Alist sys, edbuild -C sys
    edbuild -f Alist -h2 sys  # All commands not response forever.
```

Temporary solution:
No workaround at this time.

Signed off 01/14/88 in release Z01.80

Number: D200076588 Product: 64000-UX OP-ENV 300 64801S004 01.50

One-line description:
If second card cage used in pv, it may not be released

Problem:
If second card cage used in pv, it may not be released.

Problem:

If a second card cage is needed to test a card (e.g., use a card in a second cage to test IMB), if the second card is the last one allocated it may not be freed when option_test (opt) is exited.

Workaround:

After opt is finished, execute msunlock and msinit.

Temporary solution:
After opt is finished, execute msunlock and msinit.

Signed off 01/14/88 in release Z01.80

Number: D200076620 Product: 64000-UX OP-ENV 300 64801S004 01.50

One-line description:
Pressing return during a cycle command causes pv to hang

Problem:
When starting Performance verification, if the cycle softkey is pressed followed by the Return key, PV then says "Awaiting command" and the "stop" softkey is available, but the test is hung.

Temporary solution:
Do not press the return key after the "cycle" softkey.

Signed off 01/14/88 in release Z01.80

Number: D200076679 Product: 64000-UX OP-ENV 300 64801S004 01.50

One-line description:
Msinit crashes opt test user.

Problem:
msinit crashes another user's opt test. One user is in the opt test display, and does something (anything that talks to the cage). The other guy runs "msinit", and gets the message 'inconsistent module data, cleaning up'. When the opt test user then tries to do something, opt test is hosed.

Temporary solution:
Do not run msinit while any other user is running opt.

Signed off 01/14/88 in release Z01.80

Number: D200077008 Product: 64000-UX OP-ENV 300 64801S004 01.50

One-line description:
If /usr/hp64000/lock does not exist, msconfig gives odd messages.

Problem:
Text:
  if /usr/hp64000/lock does not exist, msconfig gives odd messages

If, for some reason, the /usr/hp64000/lock directory is removed (which never happens unless the customer does so!), and if there were some measurement_systems defined prior to the lock directory being removed, the command

remove_system <anysys>
gives a status message of

"system <anysys> in use by ???"

Temporary solution:
As superuser, recreate the directory /usr/hp64000/lock with the command "mkdir /usr/hp64000/lock". Then run msconfig again and remove the measurement system.

Signed off 01/14/88 in release Z01.80

Number: D200077941 Product: 64000-UX OP-ENV 300 64801S004 01.00

One-line description:
Msinit may autoconfigure incorrectly.

Problem:
Assume the following cards in the card cage:

```
     internal analyzer (wide or narrow)
     emulator #1
     state system (with EBPP)
```

```
     emulator #2
```

When configuring with msinit, the user will be asked for an analyzer for emulator #1. If the internal analyzer is selected, msinit automatically configures the EBPP as the analyzer for emulator #2 without asking any more questions This can be annoying if the state system is in fact connected to emulator #1.

Temporary solution:
Put the emulator without the analyzer (#2) in lower numbered slots. When msinit asks for the analyzer to be used, hit return to specify no analyzer.

Signed off 01/14/88 in release Z01.80

Number: D200077958 Product: 64000-UX OP-ENV 300 64801S004 01.00

One-line description:
HP enhancements are too slow to run softkeys.

Problem:
Softkeys track too slow on HP Terminals. Softkey lines are written out each time they change. Each time they are written, 400 characters are sent to the display; 80 for the contents of the line and another 320 for the underlining enhancement. The extra characters result is very slow tracking because of all the I/O.

Temporary solution:
All underlining can be turned off by creating a customized terminfo entry that does not have underlining capability.

As Superuser,

```
1) untic TERM >file   #TERM is the value of the TERM variable for
                      #that terminal
```
2)Edit the file and a) replace the top line with an entry that uniquely identifies the new terminal type. The top line contains valid names that represent the characteristics defined in this file.

Ex. 2392.Jdb,

   b) Remove the entry called "smul".

3)Save the file.

4)tic file.

5)Set the TERM variable to the new terminal type.
```
   TERM=2392.Jdb
   export TERM
```

Signed off 01/14/88 in release Z01.80

Number: D200079038 Product: 64000-UX OP-ENV 300 64801S004 01.60

One-line description:
option_test multitest can't handle more than 24 cards

Problem:
When one attempts to do multitest, including all_cages, all_slots, the option_test software quits after configuring 24 cards (6 68020 emulators). If each emulator is included separately, option_test will configure all 8 emulators, but it goes into the weeds during cycling of the test and reports multiple failures even though no low-level display show any failures. In addition, the low-level displays may show only 3 test completed while the main-level display reports thousands.

Temporary solution:
Do not test more than 24 boards at a time with multitest.

Signed off 01/14/88 in release Z01.80

Number: D200079095 Product: 64000-UX OP-ENV 300 64801S004 01.60

One-line description:
EDBUILD IS NOT WORKING PROPERLY

Problem:
Edbuild runs much slower when a code segment is at a higher address than a Data segment.

This problem is not really related to the segment type, but rather to the type of symbols in the segments.

If the segments comprising a program are ordered such that one or ones containing no global symbols are loaded at the highest addresses in the program and there are no global symbols at higher addresses, edbuild will "forget" about the segment and all the symbol information within that segment. This leads to edbuild running faster because it is not processing all the symbol information.

Note that when it runs faster in this case, edbuild is working incorrectly.

Temporary solution:
To make edbuild work correctly, add one or more global symbols to the segment at the highest address.

Signed off 01/14/88 in release Z01.80

Number: D200079293 Product: 64000-UX OP-ENV 300 64801S004 01.70

One-line description:
Multiple <cr> after selecting card will leave softkeyw in an odd state.

Problem:
If you
1) select a card
2) press carriage-return several times before the next screen

displayed

then
the softkeys show the softkeys for the 'selection' mode, but the display shows the display for the test selected.

WORKAROUND:
don't hit return until the next display appears.

Temporary solution:
don't hit return until the next display appears.

Signed off 01/14/88 in release Z01.80

Number: D200079509 Product: 64000-UX OP-ENV 300 64801S004 01.60

One-line description:
Incompatible /etc/update programs.

Problem:
The version of /etc/update which is shipped with HP 64000-UX products is NOT compatible with the HP-UX version. This results in some rather unique problems when updates for the two types of products are to be loaded (i.e., "Cannot find table of contents").

Temporary solution:
Unload the update tools each time a new update is performed. This will insure that a compatible /etc/update is used for the appropriate software.

Signed off 01/14/88 in release Z01.80

Number: D200080093 Product: 64000-UX OP-ENV 300 64801S004 01.60

One-line description:
64000-UX processes do not die when modem carrier is lost.

Problem:
When in measurement system via a modem line connection, if the carrier is lost then the processes do not get killed. Yet the highest level process (your shell) goes away. This results in another getty being spawned on this line even though meas. sys. is still reading and writing to this port.

Temporary solution:
None exists at this time.

Signed off 01/14/88 in release Z01.80

Number: D200081984 Product: 64000-UX OP-ENV 300 64801S004 01.60

One-line description:
Processes sometimes left running after parent has stopped.

Problem:
Sometimes, when the parent process to a measurement system is killed some of the measurement systems processes are left running. Please

change the behaviour of the products so that these processes die nicely.

Temporary solution:
If the tty associated with the process is a pty, then you can release the processes by

```
       cat < ptyxx
```

This causes the pending output to be flushed, and the processes will die naturally.

Signed off 01/14/88 in release Z01.80

Number: D200078923 Product: 64000-UX OP-ENV 300 64801S004 01.60

One-line description:
Search command files via "PATH" variable

Problem:
Allow search for command files using the PATH variable.

Temporary solution:
Command files must be specified with an absolute path or reside in the current working directory.

Signed off 01/14/88 in release Z01.80

Number: D200080101 Product: 64000-UX OP-ENV 300 64801S004 01.60

Keywords: ENHANCEMENT

One-line description:
64000-UX uses TERMINFO database only partially.

Problem:
64-UX software uses TERMINFO database only partially. There is no description of the fact that it uses the TERMINFO database or the fact that it uses it in a way inconsistent with the definitions in section 4 of the HPUX reference manual.

Signed off 01/14/88 in release Z01.80

Number: 1650008409 Product: 6800 C 64821 00.01

Keywords: CODE GENERATOR

One-line description:
Nested IF stmnt. with bit field structure members genrate incorr. code

Problem:
The following program generates incorrect code:

```
    "6809"
    $SEPARATE  ON$
    $RECURSIVE OFF$
    unsigned short e;
    unsigned short a;
    struct {unsigned  d:2;
            unsigned short c:2;} b;

    QUA()
      { if (b.c == 1)
        {if ((a==6)||(e==0))
           {b.c=0;} }
      else
        {if ((a!=6)&&(e!=0))
            b.c=0;
         }
       }
  main (){}
```

If the members of the structure are not bit fields then this problem does not occur.

The problem occurs after the else statemnet. The code generated for the line "{if ((a!=6)&&(e!=0))" looks like:

```
         CLRB                should be LDAB  Dstatic+00001H
         CMPB #006H
         BNE  QUA01_11
         JMP  QUA1_6
    QUA01_11
                             should be LDAB  Dstatic
         CMPB #000H
         BNE  QUA01_12
         etc.
```

Since the variables are not loaded into B the comparisons are meaningles s.

Temporary solution:
The following code might be used instead:

```
      { if ((B.C==1)&&((A==6)||(E==0)))
          {B.C=0;}
      else
       {if ((B.C!=1)&&((A!=6)&&(E!=0)))
          {B.C=0;}
        }}
```

Signed off 01/14/88 in release Z02.10

Number: D200015073 Product: 6800 C 64821 01.04

Keywords: CODE GENERATOR

One-line description:
Using a pointer dereference as an array index generates incorrect code.

Problem:
Given the following program, an incorrect "LDX ,X" is generated.

```
"C"
"6800"
struct {
          struct {
                   short a;
                   short b[5];
                 } c;
          short d;
       }  *p;
short t;
main () {
     t = p->c.b[ p->d ];
}
```

After loading accumulator B with p->d by the "LDAB ,X" instruction,
two subsequent loads of X are generated.

```
         LDX   Dmain
         LDX   ,X
```

The "LDX ,X" is the incorrect instruction.

Temporary solution:
Define the structure as follows,

```
      struct {
              struct {
                       short a;
                       short b[5];
                     } c;
              short d;
             } str;
      struct str *p;
```

Then change the source line as follows for less and correct code.

```
      t = p -> c.b[ str.d ];
```

Signed off 01/14/88 in release Z02.10

Number: D200015347 Product: 6800 C 64821 01.04

Keywords: CODE GENERATOR

One-line description:
Compiler doesn't reload register after value changes in a nested if expr

Problem:
In the following nested if expression, the compiler assumes the value of register D has not changed when it makes the comparison of test to 0X01.

```
"C"
"6800"
int test=0x0040;
main () {
  if ( test & 0xff00) {
     if ( test & 0x01 );
     }
}
```

Register D is initially loaded with the value of test, anded with 0xff00 and then set equal to the result of the and. The code for the next if begins by anding the D register with 0x01 before reloading D with the value of test.

Temporary solution:
Turn $AMNESIA ON$ before second if statement.

Signed off 01/14/88 in release Z02.10

Number: D200067629 Product: 6800 C 64821 01.06

One-line description:
Assigning 0 to bitfield causes entire structure to be reset.

Problem:
If you have a static structure containing bit fields and you set one of the members to 0 the entire structure is reset.

```
"C"
"6800"

unsigned int i;
struct {
    unsigned  one  : 1;
    unsigned  two  : 1;
    unsigned  three: 1;
    unsigned  four : 1;
    unsigned  five : 1;
    unsigned  six  : 1;
    unsigned  seven: 1;
    unsigned  eight: 1;
} bit_struct;

main() {

bit_struct.five = 0x01;
bit_struct.eight = 0x00;           /* entire structure is reset. */

}
```

Temporary solution:
The only known work around at this time is to declare a local (automatic) structure of the same type as the external. Upon entering a function equate the two structures and do this again just before the function is exited. This will cause excessive code generation.

Signed off 01/14/88 in release Z02.10

Number: D200076802 Product: 6800 C 64821 01.07

One-line description:
Array is being placed in the PROG section rather than data.

Problem:
Compiler puts array that should be in DATA section in PROG section
Example:

```
"C"
"Z80"
char  array[12];
```

The above code when compiled creates an array of twelve bytes that will reside in the PROG section. This should be placed in the DATA section.

Temporary solution:
Generate an ASM_FILE and edit the ASMProcessor file to place the array under the DATA counter.

Signed off 01/14/88 in release Z02.10

Number: D200077149 Product: 6800 C 64821 01.07

Keywords: CODE GENERATOR

One-line description:
Floating point division of 2 constants generates incorrect result

Problem:
Compiler generates incorrect code for evaluation of double division:

```
"C"
"8088"
main()
{
      double   xx;
      xx = 2.0/3.0;
      xx = 2.0;
}
```

xx is assigned the value 2.0 by both statements.

This problem also occurs with other variable types such as float, long. Any constant divided by a constant will generate this error.

Temporary solution:

```
   xx = 2.0/y;   where y = 3.0;
```

Signed off 01/14/88 in release Z02.10

Number: D200079145 Product: 6800 C 64821 01.07

Keywords: PASS 1

One-line description:
DIV, MOD and COMParisons may do unsigned estend of signed values

Problem:
Conditionals that employ div, mod, or comparison operations may not correctly extend signed short values to int size if the other operand is an unsigned short or char. For example, in the following code s is extended as if it were declared an unsigned short.

```
$SHORT_ARITH OFF$

short s;
unsigned short us;

main()
{
   if ((s/us)^0xffff)     /* both s and us get unsigned extend */
      error();
   if ((us%s)^0x007f)     /* both s and us get unsigned extend */
      error();
   if (us==s)             /* both s and us get unsigned extend */
      error();
   if (s!=us)             /* both s and us get unsigned extend */
      error();
   if (s<us)              /* both s and us get unsigned extend */
      error();
   if (s>us)              /* both s and us get unsigned extend */
      error();
}
```

Signed off 01/14/88 in release Z02.10

Number: D200080358 Product: 6800 C 64821 01.07

One-line description:
Warning message text is incorrect.

Problem:
68000 C compiler, Just updated to 2.07.

Warning 521: Unsigned integer to real conversion treated as signed.
Is incorrect.
The wording should imply that the conversion should be going the other w ay, from real to unsigned integer.

To get the error:

```
"C"
"68000"
unsigned int a;
main()
{
a=0.0;
}
```

NOTE: this error message is not in the manuals.

Temporary solution:
If you do not want to see this message you may specify

$WARN OFF$. This will turn off all warning messages.

Signed off 01/14/88 in release Z02.10

Number: 5000116848 Product: 6800 PASCAL 64811 01.08

Keywords: WITH

One-line description:
Generates bad code for parameter as ADDR of component.

Problem:
"6800"

```
 { PascBug25:  Pascal generates bad code for parameter as ADDR of
               component.

   Pascal is generating bad code if a parameter passed to a pro-
   cedure is the address of the first element of a record, and that
   record is specified in a WITH statement.

   The compiler is erroniously generating an indirect flag preceding
   the parameter specifier in the calling sequence.

   To observe the bad code, try:
     "compile PascBug25 listfile printer options expand"    }

 PROGRAM PascBug25;

 TYPE  PTR = ^INTEGER;

VAR  V: RECORD
          element_1:  INTEGER;
          element_2:  INTEGER;
        END;


PROCEDURE proc (pointer: PTR); EXTERNAL;

BEGIN
  WITH V DO
    BEGIN
      proc (ADDR (element_1)); {bad code - addr passed with indirection}
      proc (ADDR (element_2))  {good code}
    END
END.
```

Temporary solution:
Avoid use of "WITH" statement.

Signed off 01/14/88 in release Z01.90

Number: D200063123 Product: 6800 PASCAL 64811 01.10

One-line description:
functional type change of a constant into multi-byte structure gen's err

Problem:
Functional type casting of a constant into a multi-byte structure
generates bad data.

"processor"

```
PROGRAM BAD_DATA;

TYPE  EVENT = RECORD
         A   : BYTE;
         B   : BYTE;
         C   : INTEGER;
         D   : BYTE;
      END;

VAR   EVENT1   : EVENT;


PROCEDURE   GENERATOR();
   BEGIN
      EVENT1 := EVENT(0);  { THIS ASSIGNMENT RESULTS IN BAD DATA }
   END;

BEGIN
END.
```

Temporary solution:
No temporary solution.

Signed off 01/14/88 in release Z01.90

```
Number: D200075861  Product: 6800 PASCAL          64811               01.20
```

One-line description:
Unsigned_8 treated as signed value in FOR loop test.

Problem:
Assigning a constant to an unsigned_8 variable whose upper bit is set causes problems. Specifically, when the unsigned_8 var is used later it is treated as a signed value. In the example below, an unsigned_8 is assigned 247 decimal at the top of a FOR loop. When the compiler compares it is does a byte compare and therefore interprets the unsigned_8 as a signed quantity.

"processor"

```
$EXTENSIONS ON$

PROGRAM DOLOOP;

VAR   SECTORNUM,STOPSECTOR    : UNSIGNED_8;
      A                       : INTEGER;

BEGIN
    STOPSECTOR := UNSIGNED_8(247);

    FOR  SECTORNUM := UNSIGNED_8(0) TO STOPSECTOR DO BEGIN

         A := 5;

    END;
```

```
END.
```

Temporary solution:
USE AN UNSIGNED_16 FOR THE CONTROLLING VAR.

"PROCESSOR"

```
$EXTENSIONS ON$

PROGRAM DOLOOP;

VAR    SECTORNUM,STOPSECTOR   :  UNSIGNED_16;
       A                      :  INTEGER;


BEGIN

     STOPSECTOR := UNSIGNED_16(247);

     FOR SECTORNUM := UNSIGNED_16(0) TO STOPSECTOR DO BEGIN

     A := 5;
     END;
END.
```

This works for values up to 8000H.

Signed off 01/14/88 in release Z01.90

```
Number: D200079178  Product: 6800 PASCAL          64811               01.20
```

One-line description:
Pascal does not report error for assignment of constant to structure

Problem:
The Pascal/64000 compiler fails to report an error when using the functional type change operator to attempt to assign an immediate constant to a multi-byte structure.

Since the Pascal/64000 compiler does not support structured constants, there is no meaningful way to assign a constant to a structure. Each element must be assigned individually.

The Pascal/64000 compiler does report an error 505 (Warning: type changes physical size), when it should generate a fatal error. It tries to generate code for the illegal statement which will not produce the results expected by the user.
The compiler should produce fatal Error #451: Structured constants not implemented.

Here is a simple example and the workaround by explicit individual assignment statements.

```
"PASCAL" PREPROCESS
"6809"
{ Test program to demonstrate Pascal language defect }
{ Functional type change of constant to multi-byte variable }
PROGRAM PTEST101;
$EXTENSIONS ON$
 TYPE event = RECORD
                type     : BYTE;
                qualifier: BYTE;
                msg      : INTEGER;
                send_task: BYTE;
              END;
 VAR event1: event;
      i: INTEGER;
      R: REAL;

 BEGIN

{The following code is attempting to initialize}
{ the multibyte record event to zeros. }
{It should be interpreted as a Pass 1 error }
{ Error #451: Structured constants not implemented}
{ The code produced will be processor dependent }

     event1 := event(0);  {This code is incorrect Pascal}


 {Correct Pascal using individual assignments}
     event1.type:=0;
     event1.qualifier:=0;
     event1.msg:=0;
     event1.send_task:=0;
 END.
```

Signed off 01/14/88 in release Z01.90

Number: D200079236 Product: 6800 PASCAL   64811   01.20

Keywords: PASS 1

One-line description:
Functional type changes not always evaluated correctly

Problem:
Some functional type changes are not correctly evaluated. For example, the following code illustrates the problem.

```
$EXTENSIONS ON$
PROGRAM PTEST;

VAR
    S8  : SIGNED_8 ;
    U8  : UNSIGNED_8 ;
    S16 : SIGNED_16 ;
    U16 : UNSIGNED_16 ;
```

```
BEGIN
    U16 := UNSIGNED_16(S8);  (* signed extension of S8 - correct *)
    U16 := UNSIGNED_8(S8);   (* signed extension of S8 - incorrect *)

    S16 := SIGNED_16(U8);    (* unsigned extension of U8 - correct *)
    S16 := SIGNED_8(U8);     (* unsigned extention of U8 - incorrect *)
END.
```

Signed off 01/14/88 in release Z01.90

Number: D200061556 Product: 68000 ASSEMB 64845 01.10

One-line description:
EQU is not working with DC correctly.

Problem:
The EQU pseudo when used with the DC psuedo causes incorrect object code to be generated.

```
"68000"

X       EQU     7
        DC.B    X               ;VALUE IS 6
X       SET     X-1
        DC.B    X               ;VALUE IS 5
```

Temporary solution:
No temporary solution known at this time.

Signed off 01/14/88 in release Z02.10

Number: 1650026583 Product: 68000 C 64819 01.10

One-line description:
Invalid error 60 flagged.

Problem:
When multiple assignments which include a function call are made on a single line error 60 is incorrectly flagged.

```
"C"
"processor"

extern int *func();

main() {

int  *xptr;
char *cptr;

cptr = (char *)xptr = func();

}
```

Temporary solution:
Break the assignment across two lines.

```
"C"
"processor"

extern int *func();

main()
{

char *cptr;
int  *xptr;

xptr = func();
cptr = (char *)xptr;

}
```

Signed off 01/14/88 in release Z02.10

Number: 5000171124 Product: 68000 C 64819 01.09

One-line description:
Result is invalid if terenary expression evaluates to false.

Problem:
Incorrect code is generated when the result from a conditional statement is false.

```
"C"
"68000"
struct  s_type {
        int  field1;
        int  field2;  } ;

extern int my_funct();

int test(sp)
struct s_type *sp;

{  int  loc_var = 0;
   sp -> field2 = ( loc_var ? my_funct() : 0 );
}

main() {}
```

The expanded code for the above program demonstrates the problem.
The offset into the the structure is calculated only for the
true condition. If a false condition is the result the program
jumps before it calculates the offset.

Temporary solution:
Replace the terenary expression with an equivalent if-then-else.

Signed off 01/14/88 in release Z02.10

Number: 5000188839 Product: 68000 C 64819 01.10

One-line description:
$LIST ON/OFF$ doesn't work correctly.

Problem:
$LIST ON$ and $LIST OFF$ directives not working properly.
For Example:

```
"C"
"68000"
$LIST OFF$
/* comment 1 off */
$LIST ON$
/* comment 1 on  */
main()
{}
```

When compiled with the output option creates:

```
"C"
"68000"
$LIST OFF$
/* comment 1 off */
$LIST ON$
/* comment 1 on  */
```

Temporary solution:
No temporary solution at this time.

Signed off 01/14/88 in release Z02.10

Number: 5000204586 Product: 68000 C 64819 01.10

One-line description:
Wrong floating point value assigned in a terenary expression.

Problem:

```
64000 - UX C Compiler rev. 1.1
Incorrect value generated when:          result1 should = result2 but doe
"C"                                      sn't. when the var name is alone
"68000"                                  without (  +#) the results are
$FAR$                                    odd.  The same thing happens for
$CAL_ABS_LONG$                           choosing the "TRUE" value.
$LIB_ABS_LONG$
float test1, test10;
float result1, result2, result3, result4;
main() {
 float number = 10;
 float num = 1;
 test1 = 1;
 test10 = 10;
 result1 = (test1>2)?number:num;          -result1 = 3c000000h = 1/128
 result2 = (test1>2)?(number+0):(num+0);  -result2 = 3f800000h = 1.0 OK!
 result3 = (test1<2)?number;num;          -result3 = 49000000h = 2**19
 result4 = (test1<2)?(number+0):(num+0); }-result4 = 41200000h = 10 OK!
```

Signed off 01/14/88 in release Z02.10

Number: 5000206649 Product: 68000 C 64819 01.09

One-line description:
Two external declarations are made for one function.

Problem:
Compiler makes double external symbols.
Example

```
f1()
{ extern f();
     ----------
}
f2()
{ extern f();
    ----------
}
```

Compile this file,make

```
    EXTERNAL  F
    EXTERNAL  F
```

Temporary solution:
The file will still link with no errors so you can ignore the
problem. Or pull the declaration of "extern f()" outside
of the procedures which reference f() (in other words make
the declaration at the top of your source file.)

Signed off 01/14/88 in release Z02.10

Number: 5000223099 Product: 68000 C   64819   01.90

Keywords: ENHANCEMENT

One-line description:
Generate XREF from compiler which is readable by EDT.

Problem:
Antoinette Burkett sc:

Generate a listing with cross references from a C compiler program on the VAX. When editing that file using edt an error message will be encountered : error reading from input file filename : file specification %rms-w-rtb 512 byte record too large for users buffer, press return to continue. The edt editor will then show all parts of the file except the xref.

Signed off 01/14/88 in release Z02.10

Number: 5000234237 Product: 68000 C   64819   01.02

One-line description:
68008 libraries cause target processor disagree errors.

Problem:
Linker gives error message : target processors disagree file /usr/hp64000/lib/clib/168008/real_lib.R !

Signed off 01/14/88 in release Z02.10

Number: D200014340 Product: 68000 C   64819   01.07

Keywords: CODE GENERATOR

One-line description:
Bad code using $RANGE$ or $DEBUG$ with $CALL_PC_LONG$ or LIB_PC_LONG

Problem:
Bad code is generated when calling functions and the compiler directives $RANGE ON$ or $DEBUG ON$ are used in combination with the directives $CALL_PC_LONG$ or LIB_PC_LONG. For example,

```
   $DEBUG ON, LIB_PC_LONG$     int i;
   int f()  {  return 0; }
   main()   {  i = f() * 2;  /* Produces bad code */
        BSR   F          ;CALL F
        MULS  #2,D7      ;MULTIPLY RESULT IN D7 TIMES 2
        MOVE.L D7,-[A7] ;PUSH PARAMETER FOR Zoverflow_s16
        MOVE.L #Zoverflow_s16[PC],D7 ;ERROR!! D7 DESTROYED!!
        JSR   -6[PC,D7.L] ;CALL Zoverflow_s16 VIA PC LONG METHOD
        MOVE.W D7,I      ;WRONG VALUE STORED, D7 CONTAINS BAD DATA!!
```

Temporary solution:
Avoid the combination of functions, $RANGE$ or $DEBUG$, and $CALL_PC_LONG$ or LIB_PC_LONG. The example above may be rewritten to achieve the same functionality.

```
   i = f(); /*STATEMENT DOES NOT CAUSE CALL TO OVERFLOW ROUTINE*/
   i = i * 2;   /*OVERFLOW ROUTINE CALLED HERE BUT DATA IS NOT IN D7*/
```

Signed off 01/14/88 in release Z02.10

Number: D200069666 Product: 68000 C   64819   01.09

One-line description:
Compiler aborts when it encounters a legal structure declaration.

Problem:
The compiler aborts for a legal structure declaration.

```
"C"
"processor"

structure  tag { int var1; } A;

main() {

A.var1 = 1;

}
```

The compiler allocates 110H bytes of storage and then aborts.

Temporary solution:
No temporary solution at this time.

Signed off 01/14/88 in release Z02.10

Number: D200077065 Product: 68000 C   64819   01.10

Keywords: CODE GENERATOR

One-line description:
Floating point division of 2 constants generates incorrect result

Problem:
Compiler generates incorrect code for evaluation of double division:

```
"C"
"8088"
main()
{
      double   xx;
      xx = 2.0/3.0;
      xx = 2.0;
}
```

xx is assigned the value 2.0 by both statements.

This problem also occurs with other variable types such as float, long. Any constant divided by a constant will generate this error.

Temporary solution:

```
   xx = 2.0/y;   where y = 3.0;
```

Signed off 01/14/88 in release Z02.10

Number: D200079129 Product: 68000 C   64819   01.10

Keywords: PASS 1

One-line description:
DIV, MOD and COMParisons may do unsigned estend of signed values

Problem:
Conditionals that employ div, mod, or comparison operations may not correctly extend signed short values to int size if the other operand is an unsigned short or char. For example, in the following code s is extended as if it were declared an unsigned short.

```
$SHORT_ARITH OFF$

short s;
unsigned short us;

main()
{
   if ((s/us)^0xffff)      /* both s and us get unsigned extend */
      error();
   if ((us%s)^0x007f)      /* both s and us get unsigned extend */
      error();
   if (us==s)              /* both s and us get unsigned extend */
      error();
   if (s!=us)              /* both s and us get unsigned extend */
      error();
   if (s<us)               /* both s and us get unsigned extend */
      error();
   if (s>us)               /* both s and us get unsigned extend */
      error();
}
```

Signed off 01/14/88 in release Z02.10

Number: 1650009456 Product: 68000 PASCAL   64815   00.01

One-line description:
System reboot for syntax error.

Problem:
The following program will cause the 64000 station to reboot.

```
"68000"
PROGRAM  TASK4;
PROCEDURE TEST1;

VAR   X    :    REAL;
      A    :    BOOLEAN;

BEGIN

   IF X>0.0  THEN    A := FALSE    {MISSING SEMI-COLON CAUSES PROBLEM}
   IF X<0.0  THEN    A := FALSE    {THIS IF IS ALSO NEEDED. }
END;
.
```

Temporary solution:
If the 64000 station reboots during a compilation check your code for this type of syntax error.

Signed off 01/14/88 in release Z01.90

Number: 1650019224 Product: 68000 PASCAL   64815   01.10

One-line description:
Incorrect code generated for array which has boolean indices.

Problem:
Using boolean values as indices to an array will cause an incorrect offset to be generated.

```
"68000"

PROGRAM        bool;
$EXTENSIONS ON$

VAR   arr  :  ARRAY[1..4,BOOLEAN] OF BYTE;
        j  :  BOOLEAN;

BEGIN
   j := FALSE;
   arr[1,j] := 1;         { This is OK }
   arr[2,j] := 1;         { offset is calculated incorrectly
                            (200H + j). }

END.
```

Temporary solution:
Use an indice of type integer rather than boolean.

"68000"

```
$EXTENSIONS ON$

CONST    TRUE   =  1;
         FALSE  =  0;

VAR     arr : ARRAY[1..4,0..1] OF BYTE;
         j  : INTEGER;

BEGIN
  j := FALSE;
  arr[2,j] :=1;
END.
```

Signed off 01/14/88 in release Z01.90

Number: 1650019331 Product: 68000 PASCAL 64815 01.10

One-line description:
Set manipulation results in incorrect code being generated.

Problem:
The compiler appears to miss a syntax error and instead generates bad code.

```
"68000"
$EXTENSIONS ON$

PROGRAM  set_test;

TYPE  set_values        =       0..32;
      set_type          =       SET OF  set_values;

$EXTVAR ON$

VAR   operation      :  BYTE;
      x,y,z          :  INTEGER;
      set_var        :  set_type;
      values         :  set_values;
$EXTVAR OFF$

      set_member     :  INTEGER;

BEGIN

      set_member  := 04;
      set_var     := set_type[set_member];

{ I believe the customer wants to type cast set_member and therefore
  should use '()'.  The compiler generates code to clear a 5 byte
  block beginning at the address of the variable operation. }

END.
```

Temporary solution:
No temporary solution at this time.

Signed off 01/14/88 in release Z01.90

Number: D200063792 Product: 68000 PASCAL 64815 01.11

One-line description:
functional type change of a constant into multi-byte structure gen's err

Problem:
Functional type casting of a constant into a multi-byte structure generates bad data.

```
"processor"

PROGRAM BAD_DATA;

TYPE  EVENT = RECORD
        A   : BYTE;
        B   : BYTE;
        C   : INTEGER;
        D   : BYTE;
      END;

VAR   EVENT1   : EVENT;


PROCEDURE   GENERATOR();
   BEGIN
      EVENT1 := EVENT(0);  { THIS ASSIGNMENT RESULTS IN BAD DATA }
   END;

BEGIN
END.
```

Temporary solution:
No temporary solution at this time.

Signed off 01/14/88 in release Z01.90

Number: D200079202 Product: 68000 PASCAL 64815 01.12

One-line description:
Pascal does not report error for assignment of constant to structure

Problem:
The Pascal/64000 compiler fails to report an error when using the functional type change operator to attempt to assign an immediate constant to a multi-byte structure.

Since the Pascal/64000 compiler does not support structured constants, there is no meaningful way to assign a constant to a structure. Each element must be assigned individually.

The Pascal/64000 compiler does report an error 505 (Warning: type changes physical size), when it should generate a fatal error. It tries to generate code for the illegal statement which will not produce the results expected by the user.
The compiler should produce fatal Error #451: Structured constants not

implemented.

Here is a simple example and the workaround by explicit individual assignment statements.

```
"PASCAL" PREPROCESS
"6809"
{ Test program to demonstrate Pascal language defect }
{ Functional type change of constant to multi-byte variable }
PROGRAM PTEST101;
$EXTENSIONS ON$
 TYPE event = RECORD
               type     : BYTE;
               qualifier: BYTE;
               msg      : INTEGER;
               send_task: BYTE;
              END;
 VAR event1: event;
      i: INTEGER;
      R: REAL;

 BEGIN

{The following code is attempting to initialize}
{ the multibyte record event to zeros. }
{It should be interpreted as a Pass 1 error }
{ Error #451: Structured constants not implemented}
{ The code produced will be processor dependent }

     event1 := event(0);  {This code is incorrect Pascal}


 {Correct Pascal using individual assignments}
     event1.type:=0;
     event1.qualifier:=0;
     event1.msg:=0;
     event1.send_task:=0;
 END.
```

Signed off 01/14/88 in release Z01.90

Number: D200079269 Product: 68000 PASCAL   64815   01.12

Keywords: PASS 1

One-line description:
Functional type changes not always evaluated correctly

Problem:
Some functional type changes are not correctly evaluated. For example, the following code illustrates the problem.

```
$EXTENSIONS ON$
PROGRAM PTEST;
```

```
VAR
    S8  : SIGNED_8 ;
    U8  : UNSIGNED_8 ;
    S16 : SIGNED_16 ;
    U16 : UNSIGNED_16 ;

BEGIN
    U16 := UNSIGNED_16(S8);  (* signed extension of S8 - correct *)
    U16 := UNSIGNED_8(S8);   (* signed extension of S8 - incorrect *)

    S16 := SIGNED_16(U8);    (* unsigned extension of U8 - correct *)
    S16 := SIGNED_8(U8);     (* unsigned extention of U8 - incorrect *)
END.
```

Signed off 01/14/88 in release Z01.90

Number: D200076844 Product: 6809 C 64822 01.08

One-line description:
Array is being placed in the PROG section rather than data.

Problem:
Compiler puts array that should be in DATA section in PROG section
Example:

```
"C"
"Z80"
char  array[12];
```

The above code when compiled creates an array of twelve bytes that will reside in the PROG section. This should be placed in the DATA section.

Temporary solution:
Generate an ASM_FILE and edit the ASMProcessor file to place the array under the DATA counter.

Signed off 01/14/88 in release Z01.80

Number: D200077180 Product: 6809 C 64822 01.08

Keywords: CODE GENERATOR

One-line description:
Floating point division of 2 constants generates incorrect result

Problem:
Compiler generates incorrect code for evaluation of double division:

```
"C"
"8088"
main()
{
      double   xx;
      xx = 2.0/3.0;
      xx = 2.0;
}
```

xx is assigned the value 2.0 by both statements.

This problem also occurs with other variable types such as float, long. Any constant divided by a constant will generate this error.

Temporary solution:

```
   xx = 2.0/y;   where y = 3.0;
```

Signed off 01/14/88 in release Z01.80

Number: D200079152 Product: 6809 C 64822 01.08

Keywords: PASS 1

One-line description:
DIV, MOD and COMParisons may do unsigned estend of signed values

Problem:
Conditionals that employ div, mod, or comparison operations may not correctly extend signed short values to int size if the other operand is an unsigned short or char. For example, in the following code s is extended as if it were declared an unsigned short.

```
$SHORT_ARITH OFF$

short s;
unsigned short us;

main()
{
   if ((s/us)^0xffff)     /* both s and us get unsigned extend */
      error();
   if ((us%s)^0x007f)     /* both s and us get unsigned extend */
      error();
   if (us==s)             /* both s and us get unsigned extend */
      error();
   if (s!=us)             /* both s and us get unsigned extend */
      error();
   if (s<us)              /* both s and us get unsigned extend */
      error();
   if (s>us)              /* both s and us get unsigned extend */
      error();
}
```

Signed off 01/14/88 in release Z01.80

Number: D200080366 Product: 6809 C 64822 01.08

One-line description:
Warning message text is incorrect.

Problem:
68000 C compiler, Just updated to 2.07.

Warning 521: Unsigned integer to real conversion treated as signed.
Is incorrect.
The wording should imply that the conversion should be going the other w
ay, from real to unsigned integer.

To get the error:

```
"C"
"68000"
unsigned int a;
main()
{
a=0.0;
}
```

NOTE: this error message is not in the manuals.

Temporary solution:
If you do not want to see this message you may specify
$WARN OFF$. This will turn off all warning messages.

Signed off 01/14/88 in release Z01.80

Number: D200063719 Product: 6809 PASCAL 64813 01.10

One-line description:
functional type change of a constant into multi-byte structure gen's err

Problem:
Functional type casting of a constant into a multi-byte structure generates bad data.

```
"processor"

PROGRAM BAD_DATA;

TYPE  EVENT = RECORD
         A   : BYTE;
         B   : BYTE;
         C   : INTEGER;
         D   : BYTE;
      END;

VAR   EVENT1   : EVENT;


PROCEDURE   GENERATOR();
   BEGIN
      EVENT1 := EVENT(0);  { THIS ASSIGNMENT RESULTS IN BAD DATA }
   END;

BEGIN
END.
```

Temporary solution:
No temporary solution.

Signed off 01/14/88 in release Z01.60

Number: D200075911 Product: 6809 PASCAL 64813 01.11

One-line description:
Unsigned_8 treated as signed value in FOR loop test.

Problem:
Assigning a constant to an unsigned_8 variable whose upper bit is set causes problems. Specifically, when the unsigned_8 var is used later it is treated as a signed value. In the example below, an unsigned_8 is assigned 247 decimal at the top of a FOR loop. When the compiler compares it is does a byte compare and therefore interprets the unsigned_8 as a signed quantity.

```
"processor"

$EXTENSIONS ON$

PROGRAM DOLOOP;

VAR   SECTORNUM,STOPSECTOR    : UNSIGNED_8;
      A                       : INTEGER;
```

```
BEGIN
    STOPSECTOR := UNSIGNED_8(247);

    FOR  SECTORNUM := UNSIGNED_8(0) TO STOPSECTOR DO BEGIN

         A := 5;

    END;

END.
```

Temporary solution:
USE AN UNSIGNED_16 FOR THE CONTROLLING VAR.

```
"PROCESSOR"

$EXTENSIONS ON$

PROGRAM DOLOOP;

VAR    SECTORNUM,STOPSECTOR    :  UNSIGNED_16;
       A                       :  INTEGER;


BEGIN

     STOPSECTOR := UNSIGNED_16(247);

     FOR SECTORNUM := UNSIGNED_16(0) TO STOPSECTOR DO BEGIN

     A := 5;
     END;
END.
```

This works for values up to 8000H.

Signed off 01/14/88 in release Z01.60

Number: D200079186 Product: 6809 PASCAL 64813 01.11

One-line description:
Pascal does not report error for assignment of constant to structure

Problem:
The Pascal/64000 compiler fails to report an error when using the functional type change operator to attempt to assign an immediate constant to a multi-byte structure.

Since the Pascal/64000 compiler does not support structured constants, there is no meaningful way to assign a constant to a structure. Each element must be assigned individually.

The Pascal/64000 compiler does report an error 505 (Warning: type changes physical size), when it should generate a fatal error. It

tries to generate code for the illegal statement which will not produce the results expected by the user.
The compiler should produce fatal Error #451: Structured constants not implemented.

Here is a simple example and the workaround by explicit individual assignment statements.

```
"PASCAL" PREPROCESS
"6809"
{ Test program to demonstrate Pascal language defect }
{ Functional type change of constant to multi-byte variable }
PROGRAM PTEST101;
$EXTENSIONS ON$
 TYPE event = RECORD
                type     : BYTE;
                qualifier: BYTE;
                msg      : INTEGER;
                send_task: BYTE;
              END;
 VAR event1: event;
      i: INTEGER;
      R: REAL;

 BEGIN

{The following code is attempting to initialize}
{ the multibyte record event to zeros. }
{It should be interpreted as a Pass 1 error }
{ Error #451: Structured constants not implemented}
{ The code produced will be processor dependent }

     event1 := event(0);  {This code is incorrect Pascal}


 {Correct Pascal using individual assignments}
     event1.type:=0;
     event1.qualifier:=0;
     event1.msg:=0;
     event1.send_task:=0;
 END.
```

Signed off 01/14/88 in release Z01.60

Number: D200079244 Product: 6809 PASCAL 64813 01.11

Keywords: PASS 1

One-line description:
Functional type changes not always evaluated correctly

Problem:
Some functional type changes are not correctly evaluated. For example, the following code illustrates the problem.

```
$EXTENSIONS ON$
PROGRAM PTEST;

VAR
    S8  : SIGNED_8 ;
    U8  : UNSIGNED_8 ;
    S16 : SIGNED_16 ;
    U16 : UNSIGNED_16 ;

BEGIN
    U16 := UNSIGNED_16(S8);  (* signed extension of S8 - correct *)
    U16 := UNSIGNED_8(S8);   (* signed extension of S8 - incorrect *)

    S16 := SIGNED_16(U8);    (* unsigned extension of U8 - correct *)
    S16 := SIGNED_8(U8);     (* unsigned extention of U8 - incorrect *)
END.
```

Signed off 01/14/88 in release Z01.60

Number: D200068825 Product: 80286B ASSEMB 64859 01.02

One-line description:
Assembling on 64100 & linking on VAX generates erroneous absolute file

Problem:
If all programs are assembled and linked on the VAX and then downloaded to the 64100, the execution in the emulator is fine. But if the monitor is assembled on the 64100 and uploaded where it is linked on the VAX and downloded back to the 64100, the program runs off in the weeds.

Temporary solution:
There is no known work around at this time.

Signed off 01/14/88 in release Z01.30

Number: D200063909 Product: 8085 B PASCAL 64825 01.03

One-line description:
functional type change of a constant into multi-byte structure gen's err

Problem:
Functional type casting of a constant into a multi-byte structure generates bad data.

```
"processor"

PROGRAM BAD_DATA;

TYPE  EVENT = RECORD
         A   : BYTE;
         B   : BYTE;
         C   : INTEGER;
         D   : BYTE;
      END;

VAR   EVENT1   : EVENT;


PROCEDURE   GENERATOR();
   BEGIN
      EVENT1 := EVENT(0);  { THIS ASSIGNMENT RESULTS IN BAD DATA }
   END;

BEGIN
END.
```

Temporary solution:
No temporary solution at this time.

Signed off 01/14/88 in release Z01.90

Number: D200064212 Product: 8085 B PASCAL 64825 01.03

Keywords: PASS 2

One-line description:
Incorrect code generated when set elements are passed as parameters.

Problem:
Incorrect code is generated when sets are passed as parameters. The stack pointer is manipulated so that the program "goes in the weeds" after the call to the procedure. The following code is an example:

```
"processor name"
$SEPARATE ON$
$EXTENSIONS ON$
TYPE
  Letters = (a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,r);
  Set_of_Letters = SET OF Letters;
$GLOBPROC ON$
PROCEDURE Letters_Pas(Received:Set_of_Letters):EXTERNAL;
```

```
PROCEDURE Init_Set;
BEGIN
  Letters_Pas([]);    (*Code generates an extra INC SP after the
END;                     call to Letters_Pas*)
$GLOBPROC OFF$
.
```

Temporary solution:
Any set size other than 3 bytes will work correctly.

Signed off 01/14/88 in release Z01.90

Number: D200076109 Product: 8085 B PASCAL 64825 01.04

One-line description:
Unsigned_8 treated as signed value in FOR loop test.

Problem:
Assigning a constant to an unsigned_8 variable whose upper bit is set causes problems. Specifically, when the unsigned_8 var is used later it is treated as a signed value. In the example below, an unsigned_8 is assigned 247 decimal at the top of a FOR loop. When the compiler compares it is does a byte compare and therefore interprets the unsigned_8 as a signed quantity.

```
"processor"

$EXTENSIONS ON$

PROGRAM DOLOOP;

VAR   SECTORNUM,STOPSECTOR    : UNSIGNED_8;
      A                       : INTEGER;

BEGIN
    STOPSECTOR := UNSIGNED_8(247);

    FOR  SECTORNUM := UNSIGNED_8(0) TO STOPSECTOR DO BEGIN

         A := 5;

    END;

END.
```

Temporary solution:
USE AN UNSIGNED_16 FOR THE CONTROLLING VAR.

```
"PROCESSOR"

$EXTENSIONS ON$

PROGRAM DOLOOP;

VAR    SECTORNUM,STOPSECTOR   :  UNSIGNED_16;
       A                      :  INTEGER;
```

```
BEGIN

     STOPSECTOR := UNSIGNED_16(247);

     FOR SECTORNUM := UNSIGNED_16(0) TO STOPSECTOR DO BEGIN

     A := 5;
     END;
END.
```

This works for values up to 8000H.

Signed off 01/14/88 in release Z01.90

Number: D200079228 Product: 8085 B PASCAL 64825 01.04

One-line description:
Pascal does not report error for assignment of constant to structure

Problem:
The Pascal/64000 compiler fails to report an error when using the functional type change operator to attempt to assign an immediate constant to a multi-byte structure.

Since the Pascal/64000 compiler does not support structured constants, there is no meaningful way to assign a constant to a structure. Each element must be assigned individually.

The Pascal/64000 compiler does report an error 505 (Warning: type changes physical size), when it should generate a fatal error. It tries to generate code for the illegal statement which will not produce the results expected by the user.
The compiler should produce fatal Error #451: Structured constants not implemented.

Here is a simple example and the workaround by explicit individual assignment statements.

```
"PASCAL" PREPROCESS
"6809"
{ Test program to demonstrate Pascal language defect }
{ Functional type change of constant to multi-byte variable }
PROGRAM PTEST101;
$EXTENSIONS ON$
 TYPE event = RECORD
               type     : BYTE;
               qualifier: BYTE;
               msg      : INTEGER;
               send_task: BYTE;
              END;
 VAR event1: event;
      i: INTEGER;
      R: REAL;
```

```
 BEGIN

{The following code is attempting to initialize}
{ the multibyte record event to zeros. }
{It should be interpreted as a Pass 1 error }
{ Error #451: Structured constants not implemented}
{ The code produced will be processor dependent }

    event1 := event(0);  {This code is incorrect Pascal}


 {Correct Pascal using individual assignments}
    event1.type:=0;
    event1.qualifier:=0;
    event1.msg:=0;
    event1.send_task:=0;
 END.
```

Signed off 01/14/88 in release Z01.90

Number: D200079285 Product: 8085 B PASCAL 64825 01.04

Keywords: PASS 1

One-line description:
Functional type changes not always evaluated correctly

Problem:
Some functional type changes are not correctly evaluated. For example, the following code illustrates the problem.

```
$EXTENSIONS ON$
PROGRAM PTEST;

VAR
    S8  : SIGNED_8 ;
    U8  : UNSIGNED_8 ;
    S16 : SIGNED_16 ;
    U16 : UNSIGNED_16 ;

BEGIN
    U16 := UNSIGNED_16(S8);  (* signed extension of S8 - correct *)
    U16 := UNSIGNED_8(S8);   (* signed extension of S8 - incorrect *)

    S16 := SIGNED_16(U8);    (* unsigned extension of U8 - correct *)
    S16 := SIGNED_8(U8);     (* unsigned extention of U8 - incorrect *)
END.
```

Signed off 01/14/88 in release Z01.90

Number: 5000172742 Product: 8085 C 64826 01.03

Keywords: CODE GENERATOR

One-line description:
bad code generated with *pointer++ operation

Problem:
The example provided produces the following code:

```
*b++ = *c++;
    LXI   H,00002H
    DAD   SP
    MOV   E,M
    INX   H
    MOV   D,M
    XCHG
    SHLD  Dfunc+0002H
    INX   H
    XCHG
    LXI   H,00002H
    DAD   SP      /* HL = SP+2  */
    MOV   M,E    <--puts address of what b points to + 1 in
    INX   H      <---|---address of b,instead of *b++
    MOV   M,D    <---|
    LHLD  Dfunc
     .
     .
```

Temporary solution:
Use $RECURSIVE ON$ directive, or increment the pointer in a seperate operation

Signed off 01/14/88 in release Z02.10

Number: 5000202846 Product: 8085 C 64826 01.04

Keywords: CODE GENERATOR

One-line description:
> = does not work with float type

Problem:
Comparison of real numbers using "less than or equal" (LEQ) libraries may fail.

Temporary solution:
Break the comparison into two separate tests

Signed off 01/14/88 in release Z02.10

Number: 5000220186 Product: 8085 C 64826 01.04

Keywords: CODE GENERATOR

One-line description:
When subtract an integer from a pointer, get unnecessary warning message

Problem:
Unnecessary warning message is given when subtracting an integer from a pointer.

Example:

```
"C"
"8085"
unsigned short var,*ptr;
main()
{
  var=(*(ptr - 1));
}
  this generates an unnessary message about the pointer and the integer
not being the same size
```

Temporary solution:
A workaround is to add a negative integer and no warning message will be generated. Example var=(*(ptr + -1)).

Signed off 01/14/88 in release Z02.10

Number: D200068403 Product: 8085 C 64826 01.03

One-line description:
Expression used as array index generates incorrect code.

Problem:
Incorrect code is generated if an array index is an expression of the form [i+1] for example. The following program demonstrates the problem:

```
"C"
"8085"
$RECURSIVE OFF$
$SEPARATE ON$
$EXTENSIONS ON$
$INIT_ZEROES OFF$
#define number_of_pages 100
int program_page[];
delete_page(page_number)
int page_number;
{
  int i,j;

  for (i=page_number; i < (number_of_pages - 2); ++i)
  {
    program_page[i] = program_page[i+1];  (*This statement causes the
}                                                   problem*)
```

The code generated by the assignment statement is

```
   LXI      D,Istatic    (*???*)
   DAD      H
   DAD      D
   SHLD     Ddelete_page+0004H
   LHLD     Ddelete_page
   INX      H
   LXI      D,Istatic    (*???*)
   DAD      H
   DAD      D
   MOV      E,M
   INX      H
   MOV      D,M
   LHLD     Ddelete_page+0004H
   MOV      E,M
   INX      H
   MOV      D,M
```

Temporary solution:
Use a temporary variable as the array index:

```
delete_page(page_number)
int page_number;
{
  int i,j;

  for (i = page_number; i < (number_of_pages - 2); ++i)
  {
    j + i + 1;
    program_page[i] = program_page[j];
   }
}
```

Signed off 01/14/88 in release Z02.10

Number: D200076919 Product: 8085 C   64826   01.04

One-line description:
Array is being placed in the PROG section rather than data.

Problem:
Compiler puts array that should be in DATA section in PROG section
Example:

```
"C"
"Z80"
char  array[12];
```

The above code when compiled creates an array of twelve bytes that will reside in the PROG section. This should be placed in the DATA section.

Temporary solution:
Generate an ASM_FILE and edit the ASMProcessor file to place the array under the DATA counter.

Signed off 01/14/88 in release Z02.10

Number: D200077263 Product: 8085 C   64826   01.04

Keywords: CODE GENERATOR

One-line description:
Floating point division of 2 constants generates incorrect result

Problem:
Compiler generates incorrect code for evaluation of double division:

```
"C"
"8088"
main()
{
      double   xx;
      xx = 2.0/3.0;
      xx = 2.0;
}
```

xx is assigned the value 2.0 by both statements.

This problem also occurs with other variable types such as float, long. Any constant divided by a constant will generate this error.

Temporary solution:

```
   xx = 2.0/y;   where y = 3.0;
```

Signed off 01/14/88 in release Z02.10

Number: D200079087 Product: 8085 C   64826   01.04

Keywords: CODE GENERATOR

One-line description:
+=, -=, *=, & /= may fail to auto vars with $RECURSIVE ON$

Problem:
Composite assignment operators may fail to automatic variables when $RECURSIVE ON$ is in effect.
The following program segment illustrates this problem.

```
"C"
"8085"
$RECURSIVE ON$

func(i1,i2,doub)
int i1,i2;
double doub;
{
   int answer;

   answer = 1;

   answer += i2*x;  /* after this statement answer still is 1 */
                    /* however i1 = i2 * x                    */
}
```

Temporary solution:
There is no known fix at this time.

Signed off 01/14/88 in release Z02.10

Number: D200079160 Product: 8085 C 64826 01.04

Keywords: PASS 1

One-line description:
DIV, MOD and COMParisons may do unsigned estend of signed values

Problem:
Conditionals that employ div, mod, or comparison operations may not correctly extend signed short values to int size if the other operand is an unsigned short or char. For example, in the following code s is extended as if it were declared an unsigned short.

```
$SHORT_ARITH OFF$

short s;
unsigned short us;

main()
{
   if ((s/us)^0xffff)     /* both s and us get unsigned extend */
      error();
   if ((us%s)^0x007f)     /* both s and us get unsigned extend */
      error();
   if (us==s)             /* both s and us get unsigned extend */
      error();
   if (s!=us)             /* both s and us get unsigned extend */
      error();
   if (s<us)              /* both s and us get unsigned extend */
      error();
   if (s>us)              /* both s and us get unsigned extend */
      error();
}
```

Signed off 01/14/88 in release Z02.10

Number: D200080382 Product: 8085 C 64826 01.04

One-line description:
Warning message text is incorrect.

Problem:
68000 C compiler, Just updated to 2.07.

Warning 521: Unsigned integer to real conversion treated as signed.
Is incorrect.
The wording should imply that the conversion should be going the other w
ay, from real to unsigned integer.

To get the error:

```
"C"
"68000"
unsigned int a;
main()
{
a=0.0;
}
```

NOTE: this error message is not in the manuals.

Temporary solution:
If you do not want to see this message you may specify
$WARN OFF$. This will turn off all warning messages.

Signed off 01/14/88 in release Z02.10

Number: 5000152918 Product: 8085 EMULATION 64203 01.06

One-line description:
Can't use 9.5" paper to print mem map, due to centering of printout.

Signed off 01/14/88 in release Z01.07

Number: 1650004598 Product: 8086/8 ASSEMB 64853 02.01

One-line description:
Wrong values during EQU from externals.

Problem:
The 8086 assembler assigns wrong values during equ from externals.

Following program will show the bug :
```
      ext  a1,a2       (assigned somwhere else e.g. to 1 and 2)
      x    equ        a2
      ^    here  the value of a1 is assigned to x !!!!
```

Temporary solution:
No known workaround at this time.

Signed off 01/14/88 in release Z02.70

Number: 1650013235 Product: 8086/8 ASSEMB 64853 02.03

One-line description:
Tabs in source file are expanded to 6 spaces instead of 8 spaces

Problem:
The first tab encountered in the source code is expanded
to 6 characters instead of the expected 8 blanks, causing
unaligned fields in the assembly listing output.

Temporary solution:
Do not put tabs in the source code.

Signed off 01/14/88 in release Z02.70

Number: 5000136085 Product: 8086/8 ASSEMB 64853 02.00

One-line description:
Assembler/linker does not correctly handle EQU <EXT_LABEL> statement.

Problem:

Temporary solution:
Don't use the statement

```
        F001    EQU    OFFSET LAB1
```

Put the address calculation part of the expression in the MOV statement
something like

```
                MOV    AX,OFFSET LAB1
```

In other words the EQU statement is not correctly resolved by the
assembler/linker.

Signed off 01/14/88 in release Z02.70

Number: 5000170415 Product: 8086/8 ASSEMB   64853   02.03

One-line description:
The noload files aren't showing up in the listing; absolute correct

Problem:
Noload files are being linked correctly. They do not appear in the absolute file. However, the listing shows these files as loaded. The expected parenthesis around the no load file is not present.

Temporary solution:
No known temporary solution for the listing problem. Emulation can verify that the absolute is correct.

Signed off 01/14/88 in release Z02.70

Number: 5000172593 Product: 8086/8 ASSEMB   64853   02.03

Keywords: LINKER

One-line description:
Label preceeded with WORD PTR,NEAR PTR, etc. will not appear in the xref

Problem:
A label which is preceeded by a psuedo like NEAR PTR, WORD PTR, etc. does not appear in the assembler's xref in the references column.

```
"processor name"
1  LAB1      MOV AW,#0H
2            BZ   NEAR PTR LAB1
3            BR LAB1
4            END
```

Cross Reference table:

```
LINE#   SYMBOL    TYPE     REFERENCES
  1      LAB1      P        3  <----should also have 2
```

Temporary solution:
Do xref on the 64100. Problem only occurs on host computers.

Signed off 01/14/88 in release Z02.70

Number: 5000201012 Product: 8086/8 ASSEMB   64853   02.02

Keywords: CODE GENERATOR

One-line description:
Incorrect Object code generated

Problem:

```
                        1 "8086"
                        2      ASSUME  CS:PROG
                        3      PROG
 0000  2E8A84000C       4      MOV     AL,FIX[SI]
```

```
 0005  2E8A8C000C       5      MOV     CL,FIX[SI]
 000A  2E8B84000C       6      MOV     AX,WORD PTR FIX[SI]
 000F  0B1621           7FIX   DB      11,22,33

   ERRORS=  0
```

IN ABOVE PROGRAM,ASSEMBLER DOES NOT COUNT ADDRESS CORRECTRY.
000F SHOUD BE GENERATED.(NOW 000C)

The address 0C that is used for addressing FIX[SI] points to the middle of line 000A. The three addresses 0C should point to line 000F. This is a bug in the most recent SMS.

Signed off 01/14/88 in release Z02.70

Number: D200060509 Product: 8086/8 ASSEMB   64853   02.01

Keywords: LINKER

One-line description:
Linker generates error if COMN segment is not 0000H

Problem:
This SR was originally entered under the operating system, SR#5000-143487.

The linker generates a "Max addr or seg boundry exceed" when the COMN area is used and when its segment is not 0000H.

For Example: Linking a file that uses the COMN psuedo instruction at 010001000,010002000,010003000, will result in this error.

Temporary solution:
No known temporary solultion.

Signed off 01/14/88 in release Z02.70

Number: D200077768 Product: 8086/8 ASSEMB   64853   02.01

One-line description:
reusing accumulator

Problem:
The AX register is being destroyed:

```
"80186"
$EXTENSIONS ON$
$POINTER_SIZE=32$
$SEPARATE_CONST OFF$
PROGRAM CL;

TYPE

U_8  = UNSIGNED_8;
U_16 = UNSIGNED_16;
```

```
U_32   = UNSIGNED_32;

IDS    = U_8(0)..U_8(9);

BASE_CLK = RECORD
  HEAD    : U_8;
  TAIL    : U_8;
  SIZE    : U_8;
END;

CLSREC = RECORD
   TOTCNT  : U_32;
 Z PAD     : ARRAY[U_16(0)..U_16(500)] OF U_8;
ND;

AR

CLSSR   : ARRAY[U_8(0)..U_8(1),IDS] OF BASE_BLK;
CR      : ARRAY[U_8(1)..U_8(48)] OF CLSREC;

FUNCTION GCNT(ID:IDS):U_16;
VAR
CH      : U_8;
BEGIN
$LIST_CODE ON$
    GCNT := CR[CLSSR[CH,ID].HEAD].TOTCNT;
    .
    .
    .
    MOV    AL,#+00003H
    MUL    SS:BYTE PTR [BP+00004H]
    ADD    BX,AX
    MOV    AL,DS:BYTE PTR [BX]
    MOV    AH,#0
    MUL    AS                    AX GETS DESTROYED HERE
    ADD    SI,AX
    MOV    AX,DS:WORD PTR [SI]
    .
    .
    .
$LIST_CODE OFF$
END;
.
```

Temporary solution:
There is no known work around at this time.

Signed off 01/14/88 in release Z02.70

Number: 5000226613 Product: 8086/8 ASSEMB 300 64853S004 02.20

Keywords: LINKER

One-line description:
Linker can generate invalid DATE on listing file.

Problem:
Very infrequently, the linker generates an invalid DATE on the listing file. When file are compiled or assembled on 9/01, the DATE file shows as 32 Aug 1987.

```
FILE/PROG NAME        PROGRAM DATA COMMON ABSOLUTE DATE    TIME
-----------------------------------------------------------------
boot.R               00000400                     TUE, 32 Aug 1987,
                                                        ^
```

Temporary solution:
There is no work around at this time.

Signed off 01/14/88 in release Z02.70

```
Number: D200079335  Product: 8086/8 ASSEMB    VAX 64853S003       02.50

Keywords: CODE GENERATOR

One-line description:
VMS Hosted linker does not recognize logical names

Problem:
Detailed Listing for Defect Number LSDqf00689

  Submission Number: 00663LSDqf   Date Found:     870817
  Defect Status: OPEN             Date Arrived:   870817
  Prod/SCMS:/lsd/pplus/cmd/lnk    Date Received:  870820
  Version : current               Date Resolved:          (estimated)
  Severity: 1
  Showstopper: No                 Number of Duplicates:
  Workaround: No                  Additional Files: 1
  Defect/Enhancement:
   * defect

Text:
  VMS hosted linker does not recognize logical names



Submitter Supplied Information

  Submitter name:      Lee Jackson
  Submitter phone:
  Submitter address:   lee
  Activity used to find defect: casual use

Responder Supplied Information

  Responsible site:     LSD
  Responsible project:  stars
  Responsible engineer: STARS II



.submitter

When the linker is given a file name it does not test to see that the
name is a logical name,  thus if the name is a logical name, the linker
will not open the appropriate file.

Temporary solution:
There is no known work around at this time.

Signed off 01/14/88 in release Z02.70
```

```
Number: 1650026708  Product: 8086/8 C             64818            03.01

Keywords: CODE GENERATOR

One-line description:
Right shift using var. for # of places to shift generates bad code

Problem:
The following program generates incorrect code:

"C"
"8086"

main() {
   int a;
   unsigned b;
   a = 5 >> b;
     /* negates and then does a left shift instead of a right shift */
   a = 5 >> 3;
     /* works fine */
   a = 5 << 3; a = 5 << b;
     /* both work fine */
      }

Temporary solution:
Manually edit the ASM8086 file, generated using $ASM_FILE$, and
assemble.

Signed off 01/14/88 in release Z03.70
```

```
Number: 2700005520  Product: 8086/8 C             64818            00.00

Keywords: RUN-TIME LIBRARY

One-line description:
REAL NUMBER COMPARISONS MAY NOT EVALUATE CORRECTLY.

Problem:
There is a problem with REAL__COMP in the 8086 C real number library
when mantissas are compared.  Real numbers declared as float or double
do not compare correctly when they differ in the sixth figure.

   double a, b;      /* can also be declared float */
   a = 12.3456;
   b = 12.3455;

     if (a > b)
        result = 1;
     else if (a == b)
        result = 0;
     else result = 2;    /* result = 0 after this code is executed */


Temporary solution:
No known temporary solution at this time.
```

Signed off 01/14/88 in release Z03.70

Number: 5000134593 Product: 8086/8 C 64818 02.01

Keywords: CODE GENERATOR

One-line description:
1102 error generated - register needed but not availiable

Problem:
The following program generates a 1102 -register needed but not available error:

```
  "C"
  "8086"
  struct test {
    int bbb;
    short aaa;
};
bbb(ptr)
struct test *ptr;
{
short x;
x = ptr -> aaa >> 4;
}
main()
{}
```

The assembly code mneumonics generated for the expression, x = ptr -> aaa >> 4, included:

```
    mov  cl, #+00004H
    push  cx
    mov  ch, (some variable)
    pop cx
    shr ch, cl      (ch is unknown after  the pop)
```

This same code was genrated by the Pascal compiler, SR#5000-138388.

Signed off 01/14/88 in release Z03.70

Number: 5000134601 Product: 8086/8 C 64818 02.01

Keywords: CODE GENERATOR

One-line description:
Incorrect segment of array transfered to pointer

Problem:
ES is used to store the segment of an array but DS is loaded into the pointer+2H.

```
  "C"
  "8086"
  $FAR_EXTVARS$
  $POINTER_SIZE 32$
  extern struct {int a,b[3],c;} ary[5];
```

```
  int *p,i;
  foo1()
  {  p=&ary[i].b[i];}
     :
   MOV AX,SEG ary
   MOV ES,AX          {segment of ary in ES}
     :
   MOV DS:WORD PTR Dstatic+2H,DS  {moves incorrect segment into ptr}

  main(){}
```

Temporary solution:
The following program generates the correct code:

```
   extern struct {int a,b[3],c;} *p,ary[5];
   int *t,i;
   f() { p=&ary[i];
         t=&(*p).b[i];}
```

Signed off 01/14/88 in release Z03.70

Number: 5000136267 Product: 8086/8 C 64818 02.01

Keywords: CODE GENERATOR

One-line description:
ES register corrupt when used to get address of array to place in ptr.

Problem:
The following program places the incorrect segment of an array address into a pointer.

```
   "C"
   "8086"
   $POINTER_SIZE 32$
   $FAR_EXTVARS$
   extern int var [10][10];
   extern int *point;

   main() { int x,y;
            point = &var[x][y];
         MOV AX,#+0014H
              :
         MOV  AX,SEG var
         MOV  ES,AX          {ES contains the segment value of var}
              :
         MOV  AX,SEG point
         MOV  ES,AX          {ES contains the segment value of point}
              :
         MOV ES:WORD PTR point+00002H,DS {DS is unknown}
          }
```

The segment value for var should have been loaded into DS, or PUSH on th e stack.

Temporary solution:
Temporary solution:

```
    extern int var [10*10];
    extern int *point;

    main () { int x,y;
              int p;
              p= x*10+y;
              point = &var[p];
            }
```

Signed off 01/14/88 in release Z03.70

Number: 5000149229 Product: 8086/8 C   64818   03.00

Keywords: CODE GENERATOR

One-line description:
Return statement not putting value on BX register

Problem:
In the following program, the code generated for case 4 does not return a value in the "BX" register. A "12" is put in the accumulator, but nothing ever happens to it. When the program is getting ready to return we need the following command:
MOV   BX,SS:WORD PTR [BP_00008H]

```
/*******Sample Program***********/

"C" "8086"
$SEPARATE_CONST OFF$ $POINTER_SIZE=32$ $FAR_EXTVARS$
$FULL_LIST OFF$ $AMNESIA ON$ $EXTENSIONS ON$ $INIT_ZEROES OFF$
$FAR_PROC ON$  $FAR_LIBRALIES ON$
struct
{
      int   ino;
}  index[160];
sp10main()
{
         int    l,c;
         while(1)  {
               switch(c)  {
                  case 4:    return(12);
                  case 1:    for (l=15; index[(2*l+c)*16+1].ino<0;l--);
               }
         }
}
```

Temporary solution:
Breaking up the expression for the array value of index[] causes the compiler to generate the correct code. Create an "int" type variable:

```
int k
k=(2*l+c)*16+1
```

and use this inside the 'for' loop

Signed off 01/14/88 in release Z03.70

Number: 5000149757 Product: 8086/8 C   64818   03.02

One-line description:
Code generated for illegal C statement - POP BH generated

Problem:
The C compiler generates invalid code for the following program.

```
"C"
"80188"
$FAR_EXTVARS, POINTER_SIZE 32$
struct Button_def {
           char *(*labels)[];
                  };

extern struct Button_def Button_List[];

Draw_Button(but)
char but;

{
  struct Button_Def *but_p;
  char *lab_p[];
  char bindex;

  but_p = &Button_List[bindex];
  lab_p = *but_p->labels;
       /*generates invalid code including a POP BH*/
}
```

The compiler does not flag an error when a pointer is being assigned to a constant address with no memory associated with it. For example, lab_p is the name of an array. There is memory allocated for each of the array elements (i.e. lab_p[2]), but the name lap_b has no memory associated with it. THerefore, you should not be able to write "lab_p = whatever". Our compiler, however, attempts to generate code for this statement. (Note that the s/w going out in the October suds generates a "60:Lvalue expected" error for this statement).

Temporary solution:
One possible way to get the desired results is:

```
"C"
"80188"
$FAR_EXTVARS, POINTER_SIZE 32$
struct Button_Def
     { char *(*labels)[];
     };

extern struct Button_def Button_List[];

Draw_Button(but)
char but;
{   struct Button_Def  *but_p;
    char **lab_p;
    char bindex;

    but_p = &Button_List[bindex];
    lab_p = *but_p->labels;
}
```

labels is a ptr to an array of ptrs. to chars.
lab_p is now a ptr to a ptr to chars.

These two can now be equated.

Signed off 01/14/88 in release Z03.70

Number: 5000149765 Product: 8086/8 C 64818 03.02

One-line description:
Address of array element incorrectly calculated

Problem:
The following program causes the processor to go into the weeds.

```
"C"
"80188"
$FAR_EXTVARS, POINTER_SIZE 32$
struct Button_Obj {
       char butx1, buty1, butx2, buty2;
       char button_code, label_code;
       char button_parm, button_attrib; };
#define diasable_bit   0x08
extern int Button;
extern struct Button_Obj  Current_Buttons[];
extern char obj_index;

State_Machine() {
    char B_code;
    int Error;
    if ( Button != 0xFFFF)
       {
SM1:      obj_index = ( Button & 0xFF );
          Button = 0xFFFF;
          if ((Current_Buttons[obj_index].button_attrib & disable_bit) =
= 0)
  { }
   }
}
```

The code generated for the last if statement looks like:

```
        MOV AX,SEG obj_index
        MOV ES,AX
        MOV BL,ES:BYTE PTR obj_index
        MOV BH, #0
        SHL BX,#+00003H
        MOV AX,SEG Current_Buttons
        MOV ES,AX
        MOV AL, ES:BYTE PTR Current_Buttons[BX+07H]
          etc.
```

WHen tracing the emulation the same statement as MOV AL,ES:BYTE PTR Current_Buttons[BX+07H] becomes BX+08H.
This may be incorrect.

The program was loaded at 1000h,2000h,3000h. The monitor was loaded at 4000H,5000H,6000h.

The second file consists of :

```
"C"
"80188"
"$FAR_EXTVARS, POINTER_SIZE 32$
struct Button_Obj {
```

Temporary solution:
No known temporary solution.

Signed off 01/14/88 in release Z03.70

Number: 5000162487 Product: 8086/8 C 64818 03.02

Keywords: CODE GENERATOR

One-line description:
Vax not creating same code as the 64000

Problem:
8086 C compiler rev 3.4 on Vax does not create the same code as the comp iler on the 64000. The code on the 64000 is correct. The code on the VAX is not.

```
"C"              64000 creates the following      VAX creates these:
"8086"           for both assignment statements:
#$LIST_CODE$     LEA  SI,DS:CONST_data          <= that for the temp=0.5
#$LIST_ON$       LEA DI,.......
test();              .                             this for the 1.0/2.0
{                    .                          LEA SI,DS:CONST_data+0008H
double temp;         .                               .
temp=0.5;        DATA                                .
temp=1.0/2.0;    CONST_data                          .
}                DB  000H,000H,000H,000H        DB 000H,000H,000H,000H
                     000H,000H.0E0H,03FH           000H,000H,0E0H,03FH
```

Temporary solution:
There is no known work around at this time.

Signed off 01/14/88 in release Z03.70

```
Number: 5000163410  Product: 8086/8 C            64818              03.02
```

Keywords: PASS 1

One-line description:
compiler using DS segment rather than ES segment for 32 bit pointers

Problem:
When 32 bit pointers are used with structures the DS segment is moved rather than the ES segment. This occurs if arithmatic is done in a parentehsis.

EXAMPLE:

```
"C"
"80186"
$POINTER_SIZE 32$
$FAR_EXTVARS ON$

struct cmd_exe_struct{
       int test;
       int opt_parms[16];
       };

extern struct cmd_exe_struct cmd_exe[];

main()
{
int dev,*ptr;
ptr = cmd_exe[dev-1].opt_parms;
}
```

Temporary solution:
Assign arithmatic operations within parenthesis to a temporary variable.

Signed off 01/14/88 in release Z03.70

```
Number: 5000165134  Product: 8086/8 C            64818              03.02
```

Keywords: CODE GENERATOR

One-line description:
BX register overwritten with a switch statement

Problem:
The following program causes the BX register to be overwritten while calculating the index of the array.

```
"C"
"80188"
$EXTENSIONS ON$
$FAR_EXTVARS$
$POINTER_SIZE=32$
```

```
extern char Channel_Data[6][9], SelectedTrace, SIchnl_type;
#define ECG  0
#define SI_ECG_btns 0
char btns;

SIchannel_sel(chnl_type, chnl-index)
  char   chnl_type, chnl_index;
{
 switch (Channel_Data[SelectedTrace][SIchnl_type] )
   MOV  AL,#+00009H
   MOV CS,SEG SelectedTrace
   MOV  ES,CX
   MUL  ES:BYTE PTR SelectedTrace
   MOV  BX,AX
   LEA  BX,DS:Channel_Data[BX]     puts address of Channel_Data in BX
   .
   .
   .
   MOV  BL,ES:BYTE PTR SIchnl_Type  reuses BX - Channel_Data address
   MOV  BH,#0                       lost!!!
   ADD  BX,BX
   .
   .
   .

{
    case(ECG):
      btns = SI_ECG_btns;
      break;
}
}
```

Temporary solution:
There is no known work around at this time.

Signed off 01/14/88 in release Z03.70

```
Number: 5000172239  Product: 8086/8 C            64818              03.02
```

One-line description:
external used 2x in same pgm, w/ ASM_FILE ON, get 2 EXT stmts in ASMfile

Problem:
The following C program, when, using ASM_FILE ON, puts 2 EXTERNAL zzz statements, and then the ASM70108 file will not assemble.

```
"C"
"8086"
$ASM_FILE ON$

b()
{
extern zzz;
}

c()
```

```
{
extern zzz;
}
```

The ASM70108 file looks like this:

```
"70108"
;  1  0000  0  "C"
   EXTERNAL zzz
   EXTERNAL zzz
^ ERROR-ET
```

ET - Expression Type

```
07/24/87  LSD STARS DTS LINK      COPIED TO D200078766  64818
07/24/87  LSD STARS DTS LINK      COPIED TO D200078774  64818S001
07/24/87  LSD STARS DTS LINK      COPIED TO D200078782  64818S004
```

Temporary solution:
No known solution at this time.

Signed off 01/14/88 in release Z03.70

Number: 5000186718 Product: 8086/8 C   64818   03.01

Keywords: CODE GENERATOR

One-line description:
Floating point division of 2 constants generates incorrect result

Problem:
Compiler generates incorrect code for evaluation of double division:

```
"C"
"8088"
main()
{
      double   xx;
      xx = 2.0/3.0;
      xx = 2.0;
}
```

xx is assigned the value 2.0 by both statements.

This problem also occurs with other variable types such
as float, long. Any constant divided by a constant will
generate this error.

Temporary solution:

```
   xx = 2.0/y;   where y = 3.0;
```

Signed off 01/14/88 in release Z03.70

Number: 5000193466 Product: 8086/8 C   64818   03.01

Keywords: CODE GENERATOR

One-line description:
When using calculated value for array index, uses BX register twice

Problem:
When compiling the next source, compiler generates incorrect codes.

```
  "C"
  "8086"
  $FAR_EXTVARS$ $POINTER_SIZE=32$
  $FAR_LIBRARIES+$ $SEPARATE_CONST+$
  extern long CPOS[4],PCMDTBL[6][50][8];
  main(){  int stepno;
           stepno=1;
           PCMDTBL[0][stepno-1][1]=CPOS[1];}
```

The assembler listing file is as follows.

```
      :
  LES BX,SS:DWORD PTR[BP-00006H]   -----(1)
  MOV BX,SEG PCMDTBL               -----(2)
  MOV ES,BX                        -----(3)
  POP ES:[BX+00004H]
  POP ES:[BX+00006H]
}                                                   END OF 1/2
```

Compiler sets the offset address of PCMDTBL[0][stepno-1][1] to BX
register (line(1)).
But BX register is set the segment address of PCMDTBL at line (2).
Temporary solution is as follows.

```
    :
  int stepno, X;
  stepno=1;
  X=stepno-1;
  PCMDTBL[0][X][1]=CPOS[1];
    :
```

END OF 2/2

8086 C generates incorrect codes.
Array's address isn't correct.

END OF 1/2

Please refer to the verifier text No. 1/2.

END OF 2/2

Temporary solution:
No know solution at this time.

Signed off 01/14/88 in release Z03.70

Number: 5000195628 Product: 8086/8 C 64818 03.01

One-line description:
ES reg overwritten when assign char array to complex data structure

Problem:
8086 C produces wrong code for assigning a character array to a complex data structure. Example Prog:

```
"C"
"8086"
$OPTIMIZE OFF$
$FIXED_PARAMETERS ON,EXTENSIONS ON,FAR_LIBRARIES ON$
$FAR_PROC ON,POINTER_SIZE 32,SEPARATE_CONST OFF,RECURSIVE ON$
$FAR_EXTVARS ON$
struct fibtab {
char name[20];
char typ;
char att;
int first;
int max;
int last;
int byte,date,use,reserve; };

char directory[64];

struct dir {
int maxanz;
struct fibtab fib[1];
int link;
};

main()
{
 int i,wert;
 char *string1,*string2,zei1[10];
 struct dir *zei2;
```

```
 zei2 = (struct dir *) directory;

 for (i=0;i<10;i++)
     zei1[i] = '\0' ;

 for (i=0;i<4;i++)
     zei1[i] = 'A' ;

 string1 = zei1 ;
 string2 = zei2 ;
wert = 0;
i = 0;

while ((zei2->fib[wert].name[i] = zei1[i] ) != '\0' )
                i++ ;    /* this works fine ! */


while ((zei2->fib[wert].name[i] = string1[i] ) != '\0' )
                i++ ;    /* this should do alike, its just using */
                         /* a pointer instead of an array name   */
                         /* however with the 8086 C compiler it  */
                         /* produces bad code. The ES register is    */
                         /* used to addres zeis-> fib[wert], then to */
                         /* address string1[], then it is assumed */
                         /* that ES register is still loaded w/   */
                         /* fib[wert] addresses for adding .name[i]*/
}
```

Temporary solution:
Use an array name instead of a pointer.

Signed off 01/14/88 in release Z03.70

Number: 5000201749 Product: 8086/8 C 64818 03.20

Keywords: CODE GENERATOR

One-line description:
compiler reusing CX register

Problem:
This program causes the CX register to be used twice, without being reintialized in between uses.

```
"C"
"8086"

struct struct3 {char ele1; char ele2; char ele3;};
struct struct8 {char ele1; char ele2; char ele3; char ele4; char ele5;
                char ele6; char ele7; char ele8;};

func()
{
   char c;
   int  i;
struct struct8 (*src)[100];
struct struct3 (*dest)[4];

c=(*src)[i].ele3;

(*dest)[i].ele2=c;
}
```

reinitialized inbetween uses.

Temporary solution:
Create ASM file and modify

Signed off 01/14/88 in release Z03.70

Number: 5000203596 Product: 8086/8 C 64818 03.30

Keywords: CODE GENERATOR

One-line description:
Problem w/ unreleased Rev. Dx Register destroyed

Problem:
DX register using temporary data buffer was destroyed by calculateing the address of external variable, before using this temporary data.

example:

```
func(lncin,lncout)
   unsigned short lncin;
   unsigned short lncout;

   struct tgcntb {
    { char *cinpa,
           *couta;
      unsigned int cinpb
                   coutb;
    }
   extern struct centb[8] ;
   extern unsigned int centp,centc;

   if (centc < 8)
$OPTIMIZE OFF$
   {
$OPTIMIZE ON$
   centb[centp].cinpb=lncin;
   centb[centp].coutb=lncout ;
           MOV    DX,SS:WORD PTR [BP+00012H]
           MOV    AX,#+0000CH
           MOV    DX,SEG  centp     ; DX IS DESTROY
                  .
                  .
```

Temporary solution:
There is no know work around at this time.

Signed off 01/14/88 in release Z03.70

Number: 5000216036 Product: 8086/8 C 64818 03.10

Keywords: LINKER

One-line description:
The noload files aren't showing up in the listing, absolute correct

Problem:
Noload files are being linked correctly. They do not appear in the absolute file. However, the listing shows these files as loaded. The expected parenthesis around the no load file is not present. This problem is also found on the Series 300

Temporary solution:
There is no known solution at this time.

Signed off 01/14/88 in release Z03.70

Number: 5000223800 Product: 8086/8 C 64818 03.02

Keywords: CODE GENERATOR

One-line description:
Bad code generated for address of external character if for loop

Problem:
The following program generated code that did not reference the external variable correctly:

```
  "C"
  "8086"
  $FAR_PROC  ON$ $FAR_LIBRARIES ON$ $POINTER_SIZE =32$
  $SEPARATE_CONST  OFF$  $INIT_ZEROES OFF$  $FAR_EXTVARS  ON$
  extern  char ** DISPJ2;
  extern  char ** DISPJ2end;
  char  *dme  ;
 int table_set (japan)
 int japan;
{
  char **cp1,**cp2;
  if (japan) {
      for(cp1=&dme,cp2=&DISPJ2; cp2 < &DISPJ2end  ;)
           *cp1++ =*cp2++;
              }
}
```

The comiler should have pushed ES, instead of DS.

Temporary solution:
  Use a temporary varible buf, to hold &DISPJ2end:

```
  char **cp1,**cp2;
  char ***buf;   buf=&DISPJ2end;
  if (japan){
     for (cp1=&dme,cp2=&DISPJ2;cp2 < buf)
```

Signed off 01/14/88 in release Z03.70

Number: 5000229476 Product: 8086/8 C 64818 03.02

Keywords: SF1001

One-line description:
When casting unsigned ints to floats using +=, generates a error #1001

Problem:
Program generates error #1001 when using summation "+=" and casting the result of a multiplication of 2 unsigned integers into a double.

```
 "8086"
 "C"
 unsigned int b[2][5] = { {1, 1, 1, 1, 1},{1, 1, 1, 1, 1} };
 unsigned int a[5] = {1, 1, 1, 1, 1};
 double x;
 int CNTR;
 int XPIX;
 main()
    {
      for (CNTR=0;CNTR=4;++CNTR)
           x += b[CNTR][XPIX] * a[CNTR];
    }
 THIS GENERATES AN ERROR   #1001
```

Temporary solution:
Use:
x = x + b[CNTR][XPIX] * a[CNTR]

or cast the right side of the equation into a float

Signed off 01/14/88 in release Z03.70

Number: D200025858 Product: 8086/8 C 64818 01.06

Keywords: CODE GENERATOR

One-line description:
Argument to switch statement may be doubled.

Problem:
Switching on a dereferenced pointer to a structure field or on a multi-dimensional array field generates incorrect code which doubles the switch argument. The following is an example of this:

```
"C"
"PROCESSOR NAME"
unsigned short i;
struct { short z; short y[]; } *x;
main() {
```

```
  i = 1;
  switch(x -> y[i]) {    /*Generates code which doubles the argument*/
     case 0: break;
    }
}
```

Temporary solution:
Set up a temporary variable of the appropriate type and assign the expression to it. Use the temporary in the switch statement:

```
int temp;
temp = x -> y[i];
switch(temp) {
 . . . }
```

Signed off 01/14/88 in release Z03.70

Number: D200031476 Product: 8086/8 C 64818 02.00

One-line description:
Using a postfix decrement operator in a conditional statement fails.

Problem:
Using a postfix decrement operator in a conditional statement generates an incorrect comparison. When a value is supposed to be compared to zero, it is instead compared to -1. If the value is declared unsigned then this will never be true. The following code is an example:

```
"C"
"processor name"
unsigned short i,j;
char s[20],d[20];
main()  {
  j = 10;
  for (i = 0; j--; s[i++] = d[j]); /*compares j to -1, which it will
}                                           never be*/
```

The order of evaluation of the decrement operator is also incorrect, which is documented in SR #D200-031294.

Temporary solution:
Rearrange the expression so that the postfix decrement operator is not used:

```
   for (i = 0; j; s[i++] = d[--j]);
```

Signed off 01/14/88 in release Z03.70

Number: D200042606 Product: 8086/8 C 64818 02.00

One-line description:
Compiler uses wrong segment register.

Problem:
In the following example, the compiler forgets which segment register to use after several expressions involving pointers.

```
"C"
"processor name"
$POINTER_SIZE=32$
test()
{
int *p,*q,i,j;
     j=*(p+1);
     i=*p;
     q=p+2;   /*listing looks like this:
                ADD      BX,#+04H
                MOV      SS:WORD PTR [BP-00008H],BX
                MOV      SS:WORD PTR [BP-00006H],DS
                                                  ^should be ES */
}
```

Temporary solution:
Turn $AMNESIA ON$ around that expression.

Signed off 01/14/88 in release Z03.70

Number: D200049916 Product: 8086/8 C 64818 03.00

One-line description:
DX register is used although it is overwritten by IMUL instruction

Problem:
The value of the DX register is incorrect because it has been destructed by the IMUL instruction.

Example:

```
struct {
       char dummy1;
       int  var1;
       } block [12];
ROUTINE(param1,param2)
 int param1;
 long param2;
{
  block[param1].var1 = param2 /0x10000;
  .
  .
              IMUL SS:WORD PTR [BP+00008H]
              MOV SI,AX
              mov ES:WORD PTR [SI-00FFFH],DX  ; DX has been overwritten
  .                                           ; by IMUL
  .
  return;
}
```

Temporary solution:
No known temporaray solution.

Signed off 01/14/88 in release Z03.70

Number: D200057802 Product: 8086/8 C 64818 03.00

Keywords: CODE GENERATOR

One-line description:
Nonsense code generated by dynamic struc declaration in a funct.

Problem:
Dynamic data structures which access an array element of an unknown sized array cause the compiler to generate bad code.

Temporary solution:
No known solution at this time.

Signed off 01/14/88 in release Z03.70

Number: D200068684 Product: 8086/8 C 64818 03.01

Keywords: CODE GENERATOR

One-line description:
Compile incorrect when a ptr to an int is casted as a short and incremen

Problem:
The following table describes the compiled files and their results on 64100.

test case	"if" used	Ptr size	Number of increments; statemnt separation	increment and gets separate statements	BUG DESCRIPTION
TEST1	yes	32	2 ;	no	Reboots system
TEST2	no	32	2 ;	no	No increments in listing
TEST3	yes	32	2 ,	no	No increments in listing
TEST4	yes	16	2 ;	no	Reboots system
TEST5	no	16	2 ;	no	compiles correctly
TEST6	yes	16	2 ,	no	Reboots system
TEST7	yes	32	1	no	No increments in listing
TEST8	yes	16	1	no	Reboots system
TEST9	no	32	1	yes	error in factor message
TEST10	no	16	1	yes	error in factor message
TEST11	no	32	1	yes	No increments in listing
TEST12	no	16	1	yes	No increments in listing

Temporary solution:
No known temporary solution.

Signed off 01/14/88 in release Z03.70

Number: D200070532 Product: 8086/8 C 64818 03.02

Keywords: CODE GENERATOR

One-line description:
Incorrect segment passed to external function

Problem:
The compiler passes the incorrect segment to an external function.

```
"C"
"8088"

#define ushort  unsigned short
#define uint    unsigned

$POINTER_SIZE 32$
$FAR_EXTVARS$
$INIT_ZEROES OFF$
$SEPARATE_CONST OFF$
$FAR_LIBRARIES ON$
$FAR_PROC ON$
$ENTRY OFF$

extern char m[2][4][8];
extern ushort a[5];
extern movb();

char *p;

proc(ptr,num)
   char *ptr;
   uint num;
   {  uint r,j,k;

      movb( m[k][r]);
           /* MOV CL,#+00005H
              MOV BX,SS:WORD PTR [BP-00002H]
              SHL BX,CL
              LEA BX,DS:m[BX]     offset loaded into bx
              MOV AX,SEG m
              MOV ES,AX           segment for m into ES
              :
              PUSH DS             DS passed to movb() not ES
              PUSH BX             WRONG
              CALL FAR PTR movb
              :     */
   }
```

This problem occurs on version 3.20 of the compiler. ES should have been pushed on the stack instead of DS.

Signed off 01/14/88 in release Z03.70

Number: D200070615 Product: 8086/8 C   64818   03.01

Keywords: CODE GENERATOR

One-line description:
Incorrect segment passed to external function

Problem:

The compiler passes the incorrect segment to an external function.

```
"C"
"8088"

#define ushort  unsigned short
#define uint    unsigned

$POINTER_SIZE 32$
$FAR_EXTVARS$
$INIT_ZEROES OFF$
$SEPARATE_CONST OFF$
$FAR_LIBRARIES ON$
$FAR_PROC ON$
$ENTRY OFF$

extern char m[2][4][8];
extern ushort a[5];
extern movb();

char *p;

proc(ptr,num)
   char *ptr;
   uint num;
   {  uint r,j,k;

      movb( m[k][r]);
           /* MOV CL,#+00005H
              MOV BX,SS:WORD PTR [BP-00002H]
              SHL BX,CL
              LEA BX,DS:m[BX]     offset loaded into bx
              MOV AX,SEG m
              MOV ES,AX           segment for m into ES
              :
              PUSH DS             DS passed to movb() not ES
              PUSH BX             WRONG
              CALL FAR PTR movb
              :     */
   }
```

This problem occurs on version 3.20 of the compiler. ES should have been pushed on the stack instead of DS.

Signed off 01/14/88 in release Z03.70

Number: D200072371 Product: 8086/8 C   64818   03.01

Keywords: CODE GENERATOR

One-line description:
Assignment to ptr var. (w/ Separate_const off) causes corrupt stack

Problem:
The following program generates an incorrect number of PUSH's

and POP's. The problem did not occur on rev. 3.01.

```
"C"
"8086"

$POINTER_SIZE 16$  /* pointer_size 32 has this problem also */
$SEPARATE_CONST OFF$    /* required for problem to occur  */

main ()
  {  double  *a;
     *a++ = 0.0;
         /* MOV AX,SS:WORD PTR [BP-00002H]
            ADD SS:WORD PTR [BP-00002H],#+00008H
            LEA SI,DS:Const_prog
            PUSH DS   - saves the value of ds
            missing a PUSH CS here to set up for the MOVSB
            MOV BX,AX
            LEA DI,DS:[BX]
            MOV CX,#+00008H
            PUSH DS
            POP ES
            CLD
            POP DS
            REP MOVSB    - cs should have been loaded into ds but wasn;t
            POP DS
                Nothing left on the stack from this routine

          */
    }
```

This problem also occurs without the increment. Any constant assignment to a dereferenced pointer that generates a MOVSB instruction will cause the problem (i.e. pointers to long,double,strings). This problem is caused only when the constants are being stored in the code segment (CONST_prog).

Temporary solution:
Modify the assembly file, generated with the $ASM_FILE ON$ directive, to include the required PUSH CS and assemble.

Signed off 01/14/88 in release Z03.70

Number: D200074237 Product: 8086/8 C   64818   03.02

One-line description:
PROGRAMS WITH DUPLICATE GOTO LABELS MAY FAIL IN PASS 3

Problem:
C programs with duplicate user labels(for goto's)may fail in pass3.

The current SUDS C compilers may produce the error
   "comp: failed; too many errors in pass 3."
   from some C programs which previously compiled correctly.

This problem did not appear in any C compilers before April 1987.

In C it is valid to use the same goto label symbol in different

functions, since they have a logical different scope.

However, the HP64000 C cross will inform the user that these symbols are duplicate in the pass3 on the compiler. These symbols would produce duplicate label definitions when defined the ASM_FILE output is assembled. In addition the emulation products will only find one of these symbols.

The duplicate symbol detection algorithm on the HPUX/300, HPUX/500 and VAX/VMS C language compilers has an error which causes the compiler to fail.

However, the duplicate symbol checking is done after all of the relocatable and asmb_sym files have been produced. These output files are equivalent to those produced in the HP64000 version compilers. Thus, the output of the compilers is still correct, except for some trailing lines in the listing file.

The following program will cause this defect to occur:

```
"C"
"6800"
/************************************************************/
/*   TEST  file for problem with duplicate local labels     */
/*----------------------------------------------------------*/
/*   This program fails in pass 3 on VAX & HPUX/500 &/300   */
/*      While checking for duplicate asmb_sym symbols       */
/*      due to the "duplicate" error_exit labels            */
/*----------------------------------------------------------*/
/*   The workaround                                         */
/*     is to use the same local symbol only once per module */
/************************************************************/

int i;
test1()
{
     /* ... */
  if (i == 77) goto error_exit;
     /* ... */
  error_exit:

       i = -1;
     /* ... */
}
/* duplicate symbol should be created */

test2()
{
     /* ... */
  if (i == 137) goto error_exit;
     /* ... */
  error_exit:

       i = -1;
     /* ... */
}
```

Signed off 01/14/88 in release Z03.70

Number: D200077396 Product: 8086/8 C 64818 03.02

Keywords: CODE GENERATOR

One-line description:
VAX and 64100 generate different constants. VAX is incorrect.

Problem:
There is a problem with the double constant divide:

```
"C"
"8088"
main()
{
double x;
x=3.1415926535898/180.0;
}
```

The program generates an incorrect constant on the VAX, the 64K
code is fine.

Temporary solution:
There is no known solution at this time.

Signed off 01/14/88 in release Z03.70

Number: D200077727 Product: 8086/8 C 64818 03.02

Keywords: CODE GENERATOR

One-line description:
CL register being used twice

Problem:
Compiler uses CX register for two different values
example:

```
"C
"8088"

#define ushort   unsigned short
#define uint     unsigned

$FIXED_PARAMETERS ON$
$POINTER_SIZE=32$
$SEPARATE_CONST OFF$
$FAR_LIBRARIES ON$
$FAR_PROC ON$
$FAR_EXTVARS$

uint arr1[3][2],arr2[40];
ushort i;

main()
```

```
{
$LIST_CODE ON$
arr2[20] = arr1[i][0]/100+30;
       MOV   CX,#+00064H                  <---load CL w/ 100 decimal
       MOV   BL,DS:BYTE PTR Dstatic+0005CH
       MOV   BH,#0
       MOV   CL,#+00002H                  <---CL loaded w/ 2H before
       SHL   BX,CL                         it can be used for divide
       MOV   AX,DS:WORD PTR Dstatic[BX]
       SUB   DX,DX
       DIV   CX                           <---dividing by 2 not 64H
       ADD   AX,#+0001EH
       MOV   DS:WORD PTR Dstatic+00034H,AX
$LIST_CODE OFF$
```

Temporary solution:
There is no know work around at this time.

Signed off 01/14/88 in release Z03.70

Number: D200079111 Product: 8086/8 C 64818 03.02

Keywords: PASS 1

One-line description:
DIV, MOD and COMParisons may do unsigned estend of signed values

Problem:
Conditionals that employ div, mod, or comparison operations may not
correctly extend signed short values to int size if the other operand
is an unsigned short or char. For example, in the following code s
is extended as if it were declared an unsigned short.

```
$SHORT_ARITH OFF$

short s;
unsigned short us;

main()
{
   if ((s/us)^0xffff)     /* both s and us get unsigned extend */
      error();
   if ((us%s)^0x007f)     /* both s and us get unsigned extend */
      error();
   if (us==s)             /* both s and us get unsigned extend */
      error();
   if (s!=us)             /* both s and us get unsigned extend */
      error();
   if (s<us)              /* both s and us get unsigned extend */
      error();
   if (s>us)              /* both s and us get unsigned extend */
      error();
}
```

Signed off 01/14/88 in release Z03.70

Number: D200080051 Product: 8086/8 C 64818 03.02

Keywords: CODE GENERATOR

One-line description:
Cannot prevent adding Esymbol and Rsymbol info to global symbol table

Problem:
When using the linker on the VAX one does not have the capability to prevent adding Esymbol and Rsymbol information to the global symbol table. This presents a problem for me because I currently have approximately 10,000 global symbols in my source code and when I link the files this grows to approximately 30,000 symbols because the E and R values are added to the linklisting. It becomes very difficult to deal with this much information especially since the E and R values are of no use to me. I need the capability to turn off the calculation of the Entry and Return values for global symbols.

Temporary solution:
There is no known solution at this time.

Signed off 01/14/88 in release Z03.70

Number: D200080333 Product: 8086/8 C 64818 03.02

One-line description:
Warning message text is incorrect.

Problem:
68000 C compiler, Just updated to 2.07.

Warning 521: Unsigned integer to real conversion treated as signed.
Is incorrect.
The wording should imply that the conversion should be going the other w
ay, from real to unsigned integer.

To get the error:

```
"C"
"68000"
unsigned int a;
main()
{
a=0.0;
}
```

NOTE: this error message is not in the manuals.

Temporary solution:
If you do not want to see this message you may specify
$WARN OFF$. This will turn off all warning messages.

Signed off 01/14/88 in release Z03.70

Number: 1650038430 Product: 8086/8 C 300 64818S004 03.02

Keywords: CODE GENERATOR

One-line description:
printer arithmetic gives warning "integer not pointer size"

Problem:
Incorrect warning message on instructions using pointers:

```
     char *p;
     *(p-2)=5;
gives the message:
     515: Warning: integer not pointer size
```

Temporary solution:
Add, rather than subract from the pointer:

```
*(p+(-2)) = 5;
```

Signed off 01/14/88 in release Z03.70

Number: 5000221788 Product: 8086/8 C   500 64818S001   03.30

Keywords: CODE GENERATOR

One-line description:
bade code gen if local ptr to extrnl strcture is assgn vlu frm extrn ary

Problem:
Bad code generated when local pointer to external structure is
assigned a value from an external array. Example:

```
"C"
"80186"
$POINTER_SIZE 32$
$FAR_EXTVARS$
struct update_msg { char node_id; short neigh_devid[16]; } ;
extern int tdb[100];
main() { int temp_devid; struct update_msg *buffer;
         buffer->neigh_devid[temp_devid] = tdb[3];
}
```

The problem is that ES:BX is set up to point to buffer->neigh_devid[0],
then the value tdb[3] is put in AL, which requires that ES be loaded
with the segment of tdb. Then the value temp_devid is added to BX, and
finally ES:BX is used to load AL into what should be
buffer->neigh_devid[temp_devid], but is not.

Temporary solution:
Break the equation up into smaller pieces.

create temp variable hold1 of integer type. Then do:

```
 hold1 = tdb[3];
 buffer->neigh_devid[temp_devid] = hold1;
```

Signed off 01/14/88 in release Z03.70

Number: 1650004705 Product: 8086/8 PASCAL   64814   02.01

Keywords: CODE GENERATOR

One-line description:
Using ES register without initalization - REP MOVSB.

Problem:
The following programs demonstrates a code generation problem:
The ES register is used without initalization.

```
TYPE
  Pointer  =  ^Record;
  Record   =  RECORD
                first  : BYTE;
                last   : ARRAY [0..40]  OF BYTE;
              END;

VAR  A, B  :  Pointer;

BEGIN
  A^.last  :=  B^.last;
END.
```

The expanded listing shows that the DS and ES registers are
pushed, then DS is popped. The following REP MOVSB
instruction does therefore use the contents of the ES
register, which was never intialized.

Signed off 01/14/88 in release Z03.50

Number: 1650018689 Product: 8086/8 PASCAL   64814   00.01

Keywords: CODE GENERATOR

One-line description:
Stack POP'S exceed Stack PUSH'S when assignment made to ext var.

Problem:
The following program loaded character constants into Const_PROG
but fails to load the DS segment with the value of the CS segment
before a REP MOVSB. This instruction requires the source address to
be in DS and SI and the destination addres to be in ES and DI.
WHen CS is not equal to DS the program fails.

```
"80186"
$separate_const ON$
$EXTENSIONS ON$
$RECURSIVE ON$
$FAR_LIBRARIES ON$
$POINTER_SIZE 32$
$FAR_EXTVARS$
$GLOBPROC ON$

PROGRAM TEST;

TYPE
```

```
   CHARSET = SET OF CHAR;
VAR
   SET1 : CHARSET;
$EXTVAR +$
   SET2 : CHARSET;
$EXTVAR -$

BEGIN
   SET1 := ['a','b','c'];
   SET2 := ['e','f','g'];
      (* LEA SI,DS:CONST_prog+014H *)
      (* PUSH DS *)
      (* missing PUSH CS here *)
      (* MOV AX,SEG SET2 *)
      (* MOV ES,AX *)
      (* LEA DI,DS:SET2 *)
      (* MOV CX,020H *)
      (* CLD *)
      (* POP DS - this should load CS into DS, but instead it loads *)
               (* DS into DS *)
      (* REP MOVSB *)
      (* POP DS  - nothing on the stack from this procedure to pop *)
END.
```

Temporary solution:
No known temporary solution other than identifying the problem
and editing manually the ASM8086 file, then assembling the ASM8086
file.

Signed off 01/14/88 in release Z03.50

Number: 5000103432 Product: 8086/8 PASCAL   64814   02.01

One-line description:
Incorrect code generated in FOR loop.

Problem:
8086/8 Pascal compiler generates incorrect code when an mixed
mode arithmatic is done using array elements indexed by a loop
variable of type BYTE.

Temporary solution:
Use a loop variable of type SIGNED_16.

Signed off 01/14/88 in release Z03.50

Number: 5000118844 Product: 8086/8 PASCAL   64814   02.00

Keywords: CODE GENERATOR

One-line description:
Wrong code generated for expression in 'FOR' loop

Problem:
The following program creates bad code:

```
   "8086"
   $EXTENSIONS$
   TYPE
     INT = SIGNED_16;
     STRUC = RECORD  A : SIGNED_16;
                     B : SIGNED_16;
             END;
   VAR
   $GLOBVAR$
   TABLE_CADRAGE : ARRAY [1..12] OF STRUC;
   TAB_PAR_SELEC : ARRAY [1..14] OF SIGNED_16;
   PT_TAB_PAR_SELECT : INT;
   $GLOBVAR OFF$
   $GLOBPROC$
   PROCEDURE CADRER;
   VAR
      VAL : SIGNED_16;
   BEGIN
   FOR PT_TAB_PAR_SELEC := 1 TO 12 DO BEGIN
   VAL := VAL * TABLE_CADRAGE[PT_TAB_PAR_SELEC].A;
   END;
   END;
   .

   compiler puts the limit of the FOR loop in CX
   then moves CX into DX
   then moves DX into BX
   but BX has the pointer of the array stored in it.
```

Signed off 01/14/88 in release Z03.50

Number: 5000124313 Product: 8086/8 PASCAL   64814   02.01

One-line description:
The library routine, DISPOSE, overwrites the ES register

Problem:
The library routine, DISPOSE, overwrites the ES register with out
restoring it. For example:

```
  "8086"
  $POINTER_SIZE 32$
  $FAR_LIBRARIES$
  $FAR_PROC ON$

  TYPE
         A = ARRAY[1..6] OF BYTE;
         REC : ^A;

   VAR   P : REC;

  PROCEDURE TEST:
  BEGIN
   NEW (P);
   DISPOSE(P);
  END;
```

This problem was also reported on the 6809 (sr# 5000-124065).

Temporary solution:
No known temporary solution.

Signed off 01/14/88 in release Z03.50

Number: 5000134817 Product: 8086/8 PASCAL 64814 02.01

Keywords: CODE GENERATOR

One-line description:
Incorrect address calculated for beginning of ary in WITH stamnt

Problem:
The following program generates incorrect code:
```
   "processor name"
   $EXTENSIONS ON$
   PROGRAM TEST;
   TYPE NUM_REC =RECORD  NUM_BUF : ARRAY [1..24] OF BYTE;
                         TOT_NUM : BYTE; END;
        PTR = ^INTEGER;
   VAR  KEY:BYTE; NUM_INP : NUM_REC; POINTER: PTR;

   PROCEDURE DISPLAY(ROW,COLUMN,LENGTH : BYTE; START:PTR;); EXTERNAL;

   PROCEDURE IN;
   BEGIN
     WITH NUM_INP DO BEGIN
       NUM_BUF[TOT_NUM] := KEY;
                 :
           ADD   BX,AX          {BX WILL HOLD ADDR OF TOT_NUM}
                 :
       POINTER:=  ADDR(NUM_BUF);
                 :
           MOV    DS:WORD PTR DTEST+01AH,BX {Assumes BX contains addr of
                 :                              NUM_BUF, IT DOESN'T}
       DISPLAY  (5,25-TOT_NUM,TOT_NUM,POINTER);
                 :
          MOV    AL,DS:BYTE PTR [BX+00018H] {also assumes this. wrong!}
                 :
     END;
   END; .
```

Temporary solution:
Do not use the WITH statement. Reference all record members directly.

Signed off 01/14/88 in release Z03.50

Number: 5000138388 Product: 8086/8 PASCAL 64814 03.00

Keywords: CODE GENERATOR

One-line description:
Incorrect code gener. when shift function operand is mult. dimen. array

Problem:
The following code genrates an "1102 - register needed but not available" error:

```
   "8086"
   PROGRAM TEST;
   $EXTENSIONS ON$
   VAR  TABLE : ARRAY [0..3,0..15] OF SIGNED_8;

   BEGIN
     TABLE[2,3] := SHIFT(TABLE[2,3],4);
   END.
```

Part of the genrated code looks like:
```
         MOV          CL, #+00004H         ;Loads 4 into the counter
         PUSH         CX                   ;Puts 16-bit reg. onto stack
         MOV        CH, DS:BYTE PTR DTEST+000023H  ;Loads high byte with
                                                     TABLE[2,3]
         POP          CX                   ;Takes 16-bit reg. off stack,
                                            overrides the address of
                                            TABLE[2,3] in CH.
         SHL        CH,CL                  ;CH not valid.
```

If AL had been used instead of CH, the problem would not occur.

Temporary solution:
Use of a dummy variable in the shift function instead of the array element will generate the correct code.
For example:

```
    "8086"
    PROGRAM TEST;
    $EXTENSIONS ON $
    VAR TABLE : ARRAY[0..3,0..15] OF SIGNED_8;
        X : SIGNED_8;
    BEGIN
      X := TABLE[2,3];
      TABLE[2,3] := SHIFT(X,4);
    END.
```

Signed off 01/14/88 in release Z03.50

Number: 5000163824 Product: 8086/8 PASCAL 64814 03.01

Keywords: CODE GENERATOR

One-line description:
Multiplication result stored in CX and overwritten when counter reg need

Problem:
Incorrect code generated when 80186 Pascal compiler sees this code:

```
"80186"
$EXTENSIONS ON$
PROGRAM TRY;
 CONST COUNT = UNSINGED_8(4);
VAR
```

```
 V : ARRAY[UNSIGNED_8(0) .. (COUNT-UNSIGNED_8(1))] OF UNSIGNED_16;
 I,X : UNSIGNED_8;
BEGIN
 X := UNSIGNED_8(2);
 FOR I := UNSIGNED_8(0) TO (COUNT-UNSIGNED_8(1)) DO
   V[I] := UNSIGNED_16((UNSIGNED_8(4))*X);            <= Incorrect code
END.                                                     generated here!
```

See verifier text for details.

Signed off 01/14/88 in release Z03.50

Number: 5000171876 Product: 8086/8 PASCAL   64814   03.01

Keywords: CODE GENERATOR

One-line description:
Code produces an #1102 error - reg. needed but not available

Problem:
The following code produces compile time error #1102 "register needed bu not available" for the 80186 Pascal Cross Compiler (compiler rev #3.01 and Op Sys rev #2.04).

```
"80186"                       ---> continued from right column
$EXTENSIONS ON$
PROGRAM MISC;                 VAR
TYPE                          R1:ARRAY[U_8(0)..U_8(1)] OF REC1;
U_8=UNSIGNED_8;               X:U_8;
U_16=UNSIGNED_16;             PROCEDURE TEST (N:U_8);
REC1=RECORD                   BEGIN
A:U_16;                       X:=R1[R1[N].A].A
B:U_8;                        END;
END;                          .

---> continued in left column
```

Signed off 01/14/88 in release Z03.50

Number: 5000171884 Product: 8086/8 PASCAL   64814   03.01

Keywords: CODE GENERATOR

One-line description:
BX register gets overwritten when accessing arrays of records

Problem:

The following program overwrites the value originally stored in BX and then attempts to use BX for the original value.
See submitter/verifier text for declarations.

```
FUNCTION TEST:BOOLEAN;
BEGIN
TEST := (U_16(5)*R1[N2[N1]].B) > (U_16(2) *R2[N1].B);
                    :
          MUL CL
          MOV BX,AX
```

```
          MOV AX,#00002H
          MOV BX,DX       *BX register gets overwritten here
                          *which did contain 3*N1
          MUL DS:WORD PTR DMISC[BX+000AH]  *WRONG value now in BX
                    :
 END;
 .
```

Temporary solution:
Using the compiler option $AMNESIA ON$ will force the compiler to correct this situation.
OR
The ASM8086 file can be edited (generated by using $ASM_FILE on$) and the line MOV BX,DX can be changed to MOV CX,DX. Also, the line CMP BX,AX should be changed to CMP CX,AX.

Signed off 01/14/88 in release Z03.50

Number: 5000171900 Product: 8086/8 PASCAL   64814   03.01

Keywords: CODE GENERATOR

One-line description:
Contents of register A gets overwritten when accessing mult. arrys of rd

Signed off 01/14/88 in release Z03.50

Number: 5000197624 Product: 8086/8 PASCAL   64814   03.01

Keywords: PASCAL

One-line description:
for loop w/ counter = unsignd_8 type uses BX twice

Signed off 01/14/88 in release Z03.50

Number: 5000207845 Product: 8086/8 PASCAL   64814   03.00

Keywords: CODE GENERATOR

One-line description:
bad code for accessing parameters in nested procedures

Problem:
Compiler produces bad code when accessing parameters in nested procedures. Register are used twice and address are lost.

Temporary solution:
There is no known bug at this time.

Signed off 01/14/88 in release Z03.50

Number: 5000221994 Product: 8086/8 PASCAL 64814 03.03

Keywords: CODE GENERATOR

One-line description:
Using WITH statement and complex record structure causes bad code

Problem:
The 8086/8 PASCAL compiler generates code that does not execute correctly as documented below:

```
FILE TEST5:WORK

"80186"
$extensions on$
program tests;

type
rectype = record
  val1 : unsigned_32;
  val2 : unsigned_16;
  val3 : unsigned_16;
  pad2 : array[0..255] of unsigned_8;
end;

var
rec1 : array[unsigned_8(1)..unsigned_8(4)] of rectype;

$list_code on$
procedure test(aaa,bbb:unsigned_8);
begin
   with rec1[aaa] do
   begin
      val1 := val1 + rec1[bbb].val1;
     val2 := val2 + rec1[bbb].val2;
           mov      dx,ds:word ptr [bx+00004h] --- GETS [AAA].VAL2
           mov      ax,#+00108h
           mul      cx               --- STOMPS ON [AAA].VAL2
           mov      si,ax
           add      dx,ds:word ptr dtest5[si-00104h] --- ATTEMPTS TO
                                     ADD [BBB].VAL2 TO GARBAGE
           mov      ds:word ptr [bx+00004h],dx
     val3 := val3 + rec1[bbb].val3;
           mov      dx,ds:word ptr [bx+00006h] --- SAME PROB. AS ABOVE
           mov      ax,#+00108h
           mul      cx
           mov      si,ax
           add      dx,ds:word ptr dtest5[si-00102h]
           mov      ds:word ptr [bx+00006h],dx
   end;
end.
```

Temporary solution:
The bugs will disappear if either $amnesia$ is turned on or if the first operation is not a 32-bit operation.

Signed off 01/14/88 in release Z03.50

Number: D200019273 Product: 8086/8 PASCAL 64814 01.10

Keywords: RUN-TIME LIBRARY

One-line description:
Problem with Pascal I/O library (PIOLIB).

Problem:
When using the Pascal I/O library (PIOLIB), run-time error messages are not correctly returned to the 'Abort' routine. The routine 'Perror' places the error message into the DATA segment (DS) while 'Abort' assumes the message is in the CODE segment.

Please call your Hewlett-Packard Sales Representative in the event you experience this problem.

Temporary solution:
No known workaround at this time.

Signed off 01/14/88 in release Z03.50

Number: D200022137 Product: 8086/8 PASCAL 64814 01.10

Keywords: RUN-TIME LIBRARY

One-line description:
Real number comparisons may not 'evaluate' correctly.

Problem:
Real number comparisons, i.e. >, <, >=, <=, =, may not be evaluated correctly with numbers having six (6) or more significant places. For example, the following IF statement will NOT be evaluated correctly.

```
a := 12.3456;
b := 12.3455;
IF a > b THEN
   result := 1
ELSE
   result := 0;
```

In the above example, 'result' will equal '0'.

Temporary solution:
No known workaround at this time.

Signed off 01/14/88 in release Z03.50

Number: D200023283 Product: 8086/8 PASCAL 64814 01.10

Keywords: CODE GENERATOR

One-line description:
Illegal PUSH instruction generated.

Problem:
  Compiling the following program will cause the compiler to generate
an incorrect PUSH statement.

```
  $POINTER_SIZE 32$
  PROGRAM TEST;
  $EXTENSIONS$

     VAR
        VARIABLE : INTEGER;

     PROCEDURE PROC_ONE (OFFSET : UNSIGNED_16); EXTERNAL;

     BEGIN
        PROC_ONE (UNSIGNED_16(ADDR(VARIABLE)));
     END.
```

The PROC_ONE(...) statement will cause the compiler to generate a
PUSH BL instruction which is illegal.

Temporary solution:
  This problem appears to be related to the use of 32 bit pointers
in conjunction with the ADDR function.

Signed off 01/14/88 in release Z03.50

```
Number: D200053710  Product: 8086/8 PASCAL        64814             03.00
```

Keywords: CODE GENERATOR

One-line description:
Var. addresses incorrect inside nested WITH statements

Problem:

THE FOLLWOING CODE WAS TAKEN FROM THE PROGRAM LISTED IN THE SUBMITTER
TEXT. tHE FIRST LINE GENERATES A DIFFERENT VALUE FOR HIGH_BYTE THAN
THE SECOND TWO LINES.

```
DATA_BYTE [SOURCE_MOST_SIGN] := CRCS_10_ADDR [CRCS_10_NUM].HIGH_BYTE;

WITH CRCS_10_ADDR [CRCS_10_NUM] DO
DATA_BYTE[SOURCE_MOST_SIGN] := HIGH_BYTE
```

Another example of this problem can be found on !hplsdsb under
users/robin/awabug.s

Temporary solution:
Do not use nested WITH statements.

Signed off 01/14/88 in release Z03.50

```
Number: D200063750  Product: 8086/8 PASCAL        64814             03.01
```

One-line description:
functional type change of a constant into multi-byte structure gen's err

Problem:
Functional type casting of a constant into a multi-byte structure
generates bad data.

"processor"

```
PROGRAM BAD_DATA;

TYPE  EVENT = RECORD
        A   : BYTE;
        B   : BYTE;
        C   : INTEGER;
        D   : BYTE;
      END;

VAR   EVENT1   : EVENT;


PROCEDURE   GENERATOR();
   BEGIN
      EVENT1 := EVENT(0);  { THIS ASSIGNMENT RESULTS IN BAD DATA }
   END;

BEGIN
END.
```

Temporary solution:
No temporary solution at this time.

Signed off 01/14/88 in release Z03.50

```
Number: D200065060  Product: 8086/8 PASCAL        64814             03.01
```

Keywords: CODE GENERATOR

One-line description:
FOR loop counter gets destroyed when loop includes multiple WITH's

Problem:
The following program demonstrates the FOR loop counter
being destroyed. This problem only occurs if the WITH
contains at least 3 records and the record IO_UNITS has the
variable VOLT at least third in the variable list.

The code generated stores the FOR loop counter into CX, then
it later moves the counter to DI in preparation for the MOVSB
instruction for which uses CX. However, MOVSB uses DI as well.
The counter gets lost when the destination for the string move
is loaded into DI.

"8086" PREPROCESS

```
PROGRAM TEST;

   TYPE
      LOD_LIM = RECORD
                 MIN_HEAD : REAL;
                 END;
      IO_UNITS = RECORD
                 G_MW,G_MVAR,VOLT : REAL;
                 END;
      GEN_LOAD = RECORD
                  B_gener_mw : REAL;
                  B_volt : REAL;
                  END;

$GLOBVAR ON$
   VAR
      units : SIGNED_16;
      GEN_CON : ARRAY[1..17] OF GEN_LOAD;
      IO_GN : ARRAY [1..17] OF IO_UNITS;
      LL_GN : ARRAY [1..17] OF LOD_LIM;
$GLOBVAR OFF$

PROCEDURE BUFFER_DATA;
   BEGIN
     FOR units := 1 TO 17 DO
             {MOV CX,#+11H - loads cx with 17}
        BEGIN
        WITH GEN_CON[units],LL_GN[units],
             {MOV AX,CX - counter now in ax}
             IO_GN[units] DO
             {MOV CX,AX - counter moved back into CX}
           BEGIN
             B_volt := VOLT;
                  {MOV DI,CX - moves counter into DI}
                  {MOV CX, #4H - destroys contents of cx}
             B_gener_mw := G_MW;
                  {LEA DI,memory loc - destroys counter in DI}
           END {WITH};
        END {FOR};
           {MOV CX,DI -loads counter back into cx, but counter}
           {        isn't in DI}
           {LOOP        - decrements counter}
   END {BUFFER_DATA};

BEGIN
   BUFFER_DATA;
END.
```

Temporary solution:
The temporary solution requires that an IF statement be added to the BUFFERDATA procedure.

```
Change:    WITH GEN_CON[units],LL_GN[units],
                IO_GN[units] DO
              BEGIN
```

```
                B_volt := VOLT;

To:         WITH GEN_CON[units],LL_GN[units],
                 IO_GN[units] DO
              BEGIN
                IF (1=1) THEN x := TRUE;
                {x must be declared as a boolean variable}
                B_volt := VOLT;
```

Signed off 01/14/88 in release Z03.50

Number: D200068759 Product: 8086/8 PASCAL   64814   03.02

One-line description:
WITH construct causes wrong offset

Problem:
The examples for this problem have been given to the lab (file Problem3)

When using the WITH construct, the compiler calculates the correct offset for the variable in the FOR statement, but it fails to save the segment part of the pointer. Later it loads in a random number as this segment address. This only occurs when the IF statement is inside the WITH statement.

```
"80186" PREPROCESS
$EXTENSIONS ON$
$RECURSIVE ON$
$POINTER_SIZE 32$
$FAR_LIBRARIES ON$
$FAR_PROC ON$
$SEPARATE_CONST OFF$

PROGRAM CC_ANALYSIS_job;

CONST
   MAXINT = 32767;
   Addr_min = 0
   Addr_max = 8;
   Analys_b_no = UNSIGNED16(196);
   Analys_res = UNSIGNED _16(197);

TYPE
   INTEGER = SIGNED_16;
   Time_type = UNSIGNED _16;
#DEFINE ORD INTEGER;

TYPE
 que_type = ( timer_que, event_que);
 Ptr = ^INTEGER;
 Pid = RECORD
       dma : BYTE;
       process : INTEGER;
       END;
 Signal_p = ^Signal;
 Event_type = UNSIGNED_16;
 Addr_type = RECORD
             Len : BYTE;
             Adrtype : BYTE;
             Digits : ARRAY [Addr_min .. Addr_max] OF CHAR;
             END;
 User_cat = ( No_bar, Bar_1, Bar_2);
 e_Analys_b_no = RECORD
                 Dest_addr_type;
                 Cat : User_cat;
                 END;
 Ana_res_type = (Next_digit,Local_call,No_convert,
                 Outgoing_call, Invalid_digit);
 Signal = RECORD
          Link, Backlink : Signal_p;
          Que : que_type ;
          Duration : Timer_type;
         Address, Sender : Pid;
        CASE Event : Event_type OF
           Analys_b_no : (Analys_b_no_e : e_Analys_b_no);
          END;
 Analysis_Result = RECORD
                  Result : Ana_res_type;
          CASE Ana_res_type OF
                Next_digit : (Ana_index,dummy1 : BYTE);
                Local_call : (Mult_no :UNSIGNED_16);
                No_convert : (Conv_index,dummy2 : BYTE );
                Outgoing_call : (Rout_index,Bar_c:BYTE);
                Invalid_digit : (dummy3 : INTEGER);
          END;
No_sub_table = ARRAY [0..15] OF Analysis_result;
Table_p = ^No_sub_table;
Conv_type = (Replace_no,Delete_digits,Add_digits,Del_and_add);
Conv_elements = RECORD
             Conversion : Conv_type;
             Del_position : BYTE;
             Number_of_del : BYTE;
             Add_position : BYTE;
             Number_of_add : BYTE;
             Newe_digits : ARRAY [1..15] OF CHAR;
            END;

VAR
   sig : Signal_p;
i : BYTE;
   Analys_li : BYTE;
   Analys_no : ARRAY [1..15] OF BYTE;

$RECURSIVE OFF$

FUNCTION IA5_converted : BOOLEAN;
VAR
  i : BYTE;

BEGIN
   IA5_convertede := TRUE;
   WITH sig^,Analys_b_no_e,Dest_adr DO
```

```
     IF Len = 0
             :
          MOV DS:WORD PTR DIA5_converted+00006H,SI
           {but does not save segment}
          :
     THEN IA5_converted := FALSE;
     ELSE FOR i := 1 TO Len DO
        CASE Digits[i] OF
             :
          LES BX,DS:WORD PTR DIA5_converted+0006H
               {this loads garbage into ES}
          '0','1','2','3','4','5','6','7','8','9'
                    : Analys_no[i] := BYTE(Digits[i] - '0');
          '*' : Analys_no[i] := 10;
          '#' : Analys_no[i] := 11;
          'A' : Analys_no[i] := 12;
          'B' : Analys_no[i] := 13;
          'C' : Analys_no[i] := 14;
          'D' : Analys_no[i] := 15;
        END;
        Analys_li := sig^.Analys_b_no_e.Dest_adr.Len;
     END;

BEGIN
END.
     END;
 END;

BEGIN
END.
```

Temporary solution:
No known temporary solution.

Signed off 01/14/88 in release Z03.50

Number: D200068767 Product: 8086/8 PASCAL 64814 03.02

One-line description:
With construct causes wrong offset

Problem:
The examples for this problem have been given to the lab (file Problem4)

An globally ORGed variable is accessed inside a Procedure using a WITH construct. The array offset is calculated and stored in the AX register. It is then moved to the BX register. When the offset is changed, the code access the AL register instead of the BX register. Hence, the wrong offset into the array is calculated.

This problem does not occur when the variable is loacl to the procedure.

```
"80186" PREPROCESS
$EXTENSIONS ON$
$RECURSIVE ON$
```

```
$POINTER_SIZE 32$
$FAR_LIBRARIES ON$
$FAR_PROC ON$
$SEPARATE_CONST OFF$

PROGRAM WITHTEST;

CONST
   MAXINT = 32767;

TYPE
   INTEGER = SIGNED_16;
#DEFINE ORD INTEGER;

TYPE
   Tilst_txt_lintyp = RECORD
                      Pos_L : BYTE;
                      Pos_H : BYTE;
                      Li : BYTE;
                      Txtstring : ARRAY [1..42] OF BYTE;
                      END;

   Tilst_txt_element = RECORD
                         Tilst_txt_lin : ARRAY [1..2] OF Tilst_txt_linty
p;
                       END;

   Tilst_txt_type = ARRAY [0..16] OF Tilst_txt_element;

VAR



$ORG 20000000H$
    X :BYTE;
$END_ORG$

$EXTVAR ON$
Tilst_txt : Tilst_txt_type;
$EXTVAR OFF$

PROCEDURE FAST_TXT_TILST(Tilst_txtnr,Linie_ofset : INTEGER);

VAR
   N :INTEGER;
   {   X : BYTE;   makes the problem go away   }
 BEGIN
   WITH Tilst_txt[Tilst_txtnr] DO
     BEGIN
       FOR N := 1 TO 2 DO
          BEGIN
            X := Tilst_txt_lin[N].Pos_1;
                 :
              SUB AL,#+002DH   (shold be SUB BX,#+002DH)
                 :
          END;
     END;
```

```
 END;

BEGIN
END.
```

Temporary solution:
No known temporary solution.

Signed off 01/14/88 in release Z03.50

Number: D200075952 Product: 8086/8 PASCAL   64814   03.02

One-line description:
Unsigned_8 treated as signed value in FOR loop test.

Problem:
Assigning a constant to an unsigned_8 variable whose upper bit is set causes problems. Specifically, when the unsigned_8 var is used later it is treated as a signed value. In the example below, an unsigned_8 is assigned 247 decimal at the top of a FOR loop. When the compiler compares it is does a byte compare and therefore interprets the unsigned_8 as a signed quantity.

```
"processor"

$EXTENSIONS ON$

PROGRAM DOLOOP;

VAR   SECTORNUM,STOPSECTOR    : UNSIGNED_8;
      A                       : INTEGER;

BEGIN
    STOPSECTOR := UNSIGNED_8(247);

    FOR  SECTORNUM := UNSIGNED_8(0) TO STOPSECTOR DO BEGIN

         A := 5;

    END;

END.
```

Temporary solution:
USE AN UNSIGNED_16 FOR THE CONTROLLING VAR.

```
"PROCESSOR"

$EXTENSIONS ON$

PROGRAM DOLOOP;

VAR    SECTORNUM,STOPSECTOR   :  UNSIGNED_16;
       A                      :  INTEGER;
```

```
BEGIN

     STOPSECTOR := UNSIGNED_16(247);

     FOR SECTORNUM := UNSIGNED_16(0) TO STOPSECTOR DO BEGIN

     A := 5;
     END;
END.
```

This works for values up to 8000H.

Signed off 01/14/88 in release Z03.50

Number: D200077875 Product: 8086/8 PASCAL 64814 03.02

One-line description:
$RECURSIVE $ option defaults to incorrect mode (OFF)

Problem:

```
"80188"
PROGRAM ERROR;
PROCEDURE ERR_3;

VAR A,B : SINGED_16;
BEGIN
 A:= B;
    MOV AX,DS:WORD PTR DERR_3+0002H
    MOV DS:WORD PTR DERR_#, AX
 ERR_3;
   CALL FAR PTR ERR_3;
END;
```

The $RECUSRSIVE$ option should default to ON, but instead defaults to OFF.

Temporary solution:
Use explicit Directive in program.

Signed off 01/14/88 in release Z03.50

Number: D200078642 Product: 8086/8 PASCAL 64814 03.00

Keywords: CODE GENERATOR

One-line description:
pointers passed as procedure parameters not fully dereferenced.

Problem:
When passing a pointer to a pointer to a record, the pointer is not fully dereferenced. See verifier text for example.

```
$EXTENSIONS ON$
```

```
$SEPERATE_CONST OFF$
$SEPARATE ON$
$FAR_PROC ON$
$GLOBPROC ON$
$FAR_LIBRARIES $
$POINTER_SIZE 32$
$FAR_EXTVARS$
$RECURSIVE ON$
$OPTOMIZE ON$
$DEBUG OFF$
$IOCHECK OFF$
$FULL_LIST OFF$
$LIST_CODE OFF$
$LIST_OBJ  OFF$

PROGRAM Error_15;

TYPE
  ARTIKEL = RECORD
            ELE1 : SIGNED_16;
            ELE2 : SIGNED_16;
            END;
  ARTIKEL_PTR = ^ARTIKEL;

PROCEDURE ART_DEFAULTS(VAR ART : ARTIKEL);
BEGIN
END;

{this is the problem routine}

PROCEDURE ERR_15(VAR ART : ARTIKEL_PTR);
BEGIN
ART_DEFAULTS(ART^); {<---------this  generates the following error code}
             PUSH    SS:[BP+0000AH]
             PUSH    SS:[BP+00008H]
             CALL    FAR PTR ART_DEFAULTS
{The variable art^ was never fully dereferenced.  It now points at
a pointer to ARTIKEL, not at ARTIKEL}

END;
```

Temporary solution:

```
PROCEDURE WORK_15(VAR ART : ARTIKEL_PTR);
VAR
  A : ARTIKEL_PTR;  {this solution will fully dereferene the pointer}
BEGIN
```

```
A := ART;
ART_DEFAULTS(A^);
END;
```

Signed off 01/14/88 in release Z03.50

Number: D200079194 Product: 8086/8 PASCAL 64814 03.02

One-line description:
Pascal does not report error for assignment of constant to structure

Problem:
The Pascal/64000 compiler fails to report an error when using the functional type change operator to attempt to assign an immediate constant to a multi-byte structure.

Since the Pascal/64000 compiler does not support structured constants, there is no meaningful way to assign a constant to a structure. Each element must be assigned individually.

The Pascal/64000 compiler does report an error 505 (Warning: type changes physical size), when it should generate a fatal error. It tries to generate code for the illegal statement which will not produce the results expected by the user.
The compiler should produce fatal Error #451: Structured constants not implemented.

Here is a simple example and the workaround by explicit individual assignment statements.

```
"PASCAL" PREPROCESS
"6809"
{ Test program to demonstrate Pascal language defect }
{ Functional type change of constant to multi-byte variable }
PROGRAM PTEST101;
$EXTENSIONS ON$
 TYPE event = RECORD
                type     : BYTE;
                qualifier: BYTE;
                msg      : INTEGER;
                send_task: BYTE;
              END;
 VAR event1: event;
      i: INTEGER;
      R: REAL;

 BEGIN

{The following code is attempting to initialize}
{ the multibyte record event to zeros. }
{It should be interpreted as a Pass 1 error }
{ Error #451: Structured constants not implemented}
{ The code produced will be processor dependent }

     event1 := event(0);  {This code is incorrect Pascal}
```

```
 {Correct Pascal using individual assignments}
     event1.type:=0;
     event1.qualifier:=0;
     event1.msg:=0;
     event1.send_task:=0;
 END.
```

Signed off 01/14/88 in release Z03.50

Number: D200079251 Product: 8086/8 PASCAL 64814 03.02

Keywords: PASS 1

One-line description:
Functional type changes not always evaluated correctly

Problem:
Some functional type changes are not correctly evaluated. For example, the following code illustrates the problem.

```
$EXTENSIONS ON$
PROGRAM PTEST;

VAR
    S8  : SIGNED_8 ;
    U8  : UNSIGNED_8 ;
    S16 : SIGNED_16 ;
    U16 : UNSIGNED_16 ;

BEGIN
    U16 := UNSIGNED_16(S8);  (* signed extension of S8 - correct *)
    U16 := UNSIGNED_8(S8);   (* signed extension of S8 - incorrect *)

    S16 := SIGNED_16(U8);    (* unsigned extension of U8 - correct *)
    S16 := SIGNED_8(U8);     (* unsigned extention of U8 - incorrect *)
END.
```

Signed off 01/14/88 in release Z03.50

Number: 5000121830 Product: 9900/0 ASSEMB 64847 00.46

Keywords: CODE GENERATOR

One-line description:
Autoincrement with indirect addressing does not assemble correctly.

Problem:
THE HOSTED 9989 ASSEMBLER/LINKER ON THE 9000 DOES NOT ASSEMBLE CORRECTLY WHEN THE CUSTOMER USES THE INDIRECT ADDRESSING WITH AUTOINCREMENT. THE CODE DOES ASSEMBLE CORRECTLY ON THE 64000.
EXAMPLE:

```
                 MOV     R11,*R7+
AND
                 INC     *R6+
```

GENERATES THE ERROR MISSING OPERATOR, AN ARITHMETIC OPERATOR WAS EXPECTED AND WAS NOT FOUND ON THE HOSTED SOFTWARE.

Temporary solution:
There is no know work around at this time.

Signed off 01/14/88 in release Z01.80

Number: 5000232959 Product: 9900/0 ASSEMB 64847 00.00

Keywords: CODE GENERATOR

One-line description:
Problem with negative displacementwith SBO SBZ instructions.

Temporary solution:
All constants have a 32 bit internal representation. When negative numbers need to be reprsented, the following syntax should be used:

```
 SBO  0FFFFFFFFH
```

When the used does not put 8 F's, the system sign extends the high order bytes, and interprets the number as positive 255, outside the legal range for this instruction.

Signed off 01/14/88 in release Z01.80

Number: D200035220 Product: 9900/0 ASSEMB 64847 00.46

One-line description:
In macros, "" and '' are not equivalent.

Problem:
In macros, "" and '' are not equivalent, but should be as defined in the manual.

Signed off 01/14/88 in release Z01.80

Number: 5000224022 Product: F9450 EMULATION 64286 01.04

One-line description:
Answer to "Emulate SCR?" is automatically changed to "yes"

Problem:
If the emulator is configured for 17 bits of address or greater, and the user modifies the memory configuration, then the configuration setting of "emulate the system configuration register" is set to "yes" even if the user had previously set the response to "no".

Temporary solution:
If you don't want to emulate the SCR, and you are using 17 bits of address or greater, then you must always check the answer to "emulate system configuration register?" to be sure that the answer is "no".

Signed off 01/14/88 in release Z01.05

Number: 5000164004 Product: F9450 EMULATION 64286 01.03

Keywords: ENHANCEMENT

One-line description:
Cannot single step through a Software Breakpoint

Problem:
It is not possible to single step through a Software Breakpoint and have the breakpoint actually function properly. If you step the BEX code, you end up single stepping the monitor code itself at the Software Break Entry location. Since the monitor is not re-entrant, this causes some minor problems. The biggest problem is that the user's registers and IC get lost since the monitor itself uses some of the registers.

This problem occurs most often when the user has multiple software breakpoints set. The user encounters a software breakpoint while running, then decides to single step the code at the breakpoint location. The user then accidentally single steps the BEX code itself and finds himself stepping the emulation monitor code.

Temporary solution:
A good work-around for this problem is to "modify software breakpoints clear" after breaking to the emulation monitor. Then perform the single stepping that is desired. And to restore all of the breakpoints, just enter "modify software_breakpoints set" before using the "run" command.

Signed off 01/14/88 in release Z01.05

Number: D200066357 Product: F9450 EMULATION 64286 01.03

Keywords: ENHANCEMENT

One-line description:
MASK, STATUS, and IC are not always cleared when running from reset.

Problem:
When you enter the monitor from a hardware reset, the MASK, STATUS word, and Instruction Counter contain the values that existed at the previous break entry into the monitor. These three registers should be cleared when running in the monitor from a reset condition since the F9450 cpu will clear the same registers after a reset. Note that the values are initialized to zero in the assembly code for the monitor, but there is no code that actively resets the values to zero upon entering the monitor at the RESET_IN location.

Signed off 01/14/88 in release Z01.05

Number: D200068957 Product: F9450 EMULATION 64286 01.04

Keywords: ENHANCEMENT

One-line description:
Enhance the register display to show the Pending Interrupt Register

Problem:
The current register display does not show the Pending Interrupt Register.

Temporary solution:
A good temporary implementation of this enhancement can be made by the user of the F9450 emulator. The emulation monitor can be modified such that the PIR is displayed in place of the Fault Register. All that is necessary is a small modification to the F9450 monitor program so that the PIR is stored in the memory location where the Fault Register is now stored.

To implement this idea, replace the instruction XIO R14,RCFR with the instruction XIO R14,RPIR in the "UNLOAD" section of the monitor, MON_9450:HP:source.

Signed off 01/14/88 in release Z01.05

Number: 1650019257 Product: HOST SOFTWARE / VAX 64882 01.20

One-line description:
VAX help on MAPBUS command causes system error

Problem:
Digital VMS help command gives an access violation when part of a keyword is entered and the help file has a blank line after the keyword. The MAPBUS help entry has a spurious blank line after the MAPBUS keyword.

Temporary solution:
Delete spurious blank line after MAPBUS keyword in help file.

Signed off 01/14/88 in release Z02.30

Number: D200069104 Product: HOST SOFTWARE / VAX 64882 01.70

One-line description:
64100 cluster disk free list is corrupted so there is not enough space

Problem:
The HSL seems to destroy the free page list during operatino.
Transfer reports no space available to transfer the file.
Directory reports no space available. Directory of all user ids shows that the disk has a significant portion remaining. (+20%)
This appears to be happenning at 4 sites. One site has sent me an image backup made after the problem occurred. This is probably just a VAX HSL link problem.

Temporary solution:
SOFT FIX will generally repair the problem, or at least indicate any exceptionally large files on the disk. These files can then be purged.

Signed off 01/14/88 in release Z02.30

Number: D200082669 Product: HOST SOFTWARE / VAX 64882 02.00

Keywords: TRANSFER

One-line description:
CSIB process does not come up on system bootup

Problem:
CSIB process does not come up on system boot-up. The following error message is seen on the system console:

```
      CSIBx: unable to connect to HSL driver, check installation.
```

This is an intermittent problem, and rebooting the system can work.

Temporary solution:
Try rebooting the system until the CSIB process comes up.

Signed off 01/14/88 in release Z02.30

Number: D200082636 Product: HOST SOFTWARE / 300 64883 01.00

One-line description:
Transfer does not handle imbedded linefeeds in binary files.

Problem:
Transfer software not capable of handling imbedded linefeeds in binary files.

Signed off 01/14/88 in release Z01.10

Number: 1650018721 Product: HOST SOFTWARE / 500 64880 01.60

One-line description:
Debug file transfers may not function with 14 character file names.

Problem:
Debug file transfer may not function correctly when 14 character file names are used during program development. As an example, consider the following:

1) create a source file with a 14-character filename (i.e., "source7890ab.C") and compile

2) link the relocatable file and create an absolute filename of the form "source7890ab.X")

3) transfer the data to a 64000 system using the command

```
$ transfer -thda source789ab.L :TMP
```

Transfer will report problems with one of the temporary files which transfer created.

Signed off 01/14/88 in release Z01.80

Number: 5000219204 Product: HOST SOFTWARE / 500 64880 01.06

Keywords: TRANSFER

One-line description:
Bad file format does not cause error in transfer.

Problem:
The transfer software will successfully transfer files with formats that are not supported. For example, an absolute file with records greater than 256 bytes will successfully transfer. However, this file will cause unpredictable problems on the destination system.

The example given is a file transfered from 500 to 64000. This transfer completes successfully. When the file is transfered back to the 500, an error is generated and the transfer is not complete. The error gives no indication that the file format is wrong.

Temporary solution:
To insure the successful transfer and usability of files, the users should use the utility read64 on files to be transfered. This utility will indicate the size of the records in the files. For a description of the format of absolute files, see manual 64880-90903.

Signed off 01/14/88 in release Z01.80

Number: D200019265 Product: HOST SOFTWARE / 500 64880 01.10

One-line description:
Invalid file names are not detected by the transfer utility.

Problem:
When constructing the transfer command as a one line command, the transfer software does not verify the syntax of the 64000 file name before sending the command to the 64000. The result is a syntax error on the 64000 for invalid file names.

Signed off 01/14/88 in release Z01.80

Number: 1650006908 Product: OPERATING SYSTEM 64100 00.01

Keywords: DC600

One-line description:
64000 backup from 7946 to 9144 with 150' tape produces wrong message.

Problem:
Backup to a 9144 tape drive and using a tape that is too small for the disc, i.e. a 150' tape, will produce an incorrect message:
"Disc and DC600 units must be at the same controller address"

Temporary solution:
Multi-tape backups to a 9144 are not currently supported on the 64000 system.

Signed off 01/14/88 in release Z02.10

Number: 1650033209 Product: OPERATING SYSTEM 64100 02.07

One-line description:
Unique label is flagged as undefined in macro expansion.

Problem:
Using labels in macro causes incorrect error message
Error-ML - Macro Label, Label not found within macro body.

```
          "68000"
          REGMEMS   MACRO  &INST,&FPM,&EA,&IX
                    .IF "&IX"  .EQ. "''"  LLLL1_&&&&
                    MOVE.L   D0,D1
          LLLL1_&&&& MOVE.L  D1,D0
          NUL_REL   TST.B    D7
                    MEND
```

The errors occur in the maocro calls within the main program:

```
 *Main Program

         REGMENS  FMOVE,FP0,D1
         .IF '' .EQ. "" LLLL1_0001
                                    ^
ERROR-ML
```

Temporary solution:
No temporary solution known at this time.

Signed off 01/14/88 in release Z02.10

Number: 5000111666 Product: OPERATING SYSTEM 64100 02.01

One-line description:
Formatting a floppy from a command file sometimes is unsuccessful.

Problem:
When the 64941 floppydrive is used to initialise a floppy from a

command file, it first shows a number of retries, than it seems to format but at the end 'System area on disc 0 bad: format failed' is displayed.
Formatting by giving the command manually, always goes well.
Formatting from a command file sometimes goes well but mostly not.
The floppy used was a 'XIDEX' precision flexible disk.

Temporary solution:
A temporary solution is to put something in the command file to delay execution of the next command until the drive has finished initial-ization.

Signed off 01/14/88 in release Z02.10

Number: 5000181552 Product: OPERATING SYSTEM 64100 02.04

Keywords: DC600

One-line description:
No multi-tape backup strategy for discs > 64MB on the 64000.

Signed off 01/14/88 in release Z02.10

Number: 5000189985 Product: OPERATING SYSTEM 64100 02.07

One-line description:
Instructions assembling differently than previous assembler.

Problem:
TWO INSTRUCTIONS ASSEMBLE DIFFERENTLY THAN THEY DID ON A PREVIOUS RELEAS
E. THIS WOULD REQUIRE CHANGING A GREAT DEAL OF CODE.

```
FOR MACROS           IF  LABLE.EQ.LABLE2

                     NOW GIVES ERRORS UNLESS BLANKS ARE ADDED ON EACH SIDE
                     OF .

                     IF LABLE . EQ . LABLE2


FOR REFERENCES OF THE TYPE  [A1,D3.L]  THERE IS NOW AN ERROR POINTING TO
                     THE .L WHICH SAYS MISSING OPERAND.
```

Temporary solution:
No temporary solution known at this time.

Signed off 01/14/88 in release Z02.10

Number: 5000202770 Product: OPERATING SYSTEM 64100 02.06

One-line description:
Logical operators generate MO error.

Problem:
All assemblers generate Missing Operand error at logical operator.
For example the .NE. operator will generate an error.

Temporary solution:
Reload operating system 2.05.

Signed off 01/14/88 in release Z02.10

Number: D200027953 Product: OPERATING SYSTEM 64100 02.01

Keywords: COPY

One-line description:
"copy f:link_com to display" doesn't display all attributes of "f".

Problem:
Link a file with list, xref, overlap_check, and comp_db on. Then, when one does "copy file:link_com to display", only the values for map and xref appear.

Temporary solution:
None at this time.

Signed off 01/14/88 in release Z02.10

Number: D200062604 Product: OPERATING SYSTEM 64100 02.04

One-line description:
Comment is taken as a parameter when a null parameter is passed.

Problem:
If you invoke a macro with a null parameter and a comment is included on the line, the comment will be taken as the parameter (even with a semi-colon).

```
"68000"

X           MACRO           &PARM
            .IF             "&PARM".EQ."''"  DONE
            MOVE            #3,D0
DONE        .NOP
            MEND

            X         HI
            X                   ;THIS COMMENT WILL BE A PARAM.
            END
```

Temporary solution:
No work around at this time other than not placing comments on the macro invokation line.

Signed off 01/14/88 in release Z02.10

Number: D200085050 Product: OPERATING SYSTEM 64100 02.07

One-line description:
Phase error incorrectly reported on 64000 and hosted assemblers.

Problem:
The 1750 assembler reports a spurious error message PH_ERR (Phase error) due to the usage of COUNTER_UPDATE vs. GEN_CODE in passes 1 and 2 of the assembler.

Cause:
This defect has been fixed and this report is being submitted for QA release purposes.

Temporary solution:
None At this Time...

Signed off 01/14/88 in release Z02.10

Number: 5000223792 Product: TMS 32010 MODULES 64285 01.00

One-line description:
Inverse assembly for 1E91H is incorrectly shown as SUB *-, E, 0

Problem:
The instruction code 1E91H should be inverse assembled to "SUB *-, E,1" but instead is displayed as "SUB *-, E,0"

Temporary solution:
Until this problem is fixed, you can verify whether the inverse assembly is correct by examining the code itself. 1E91H is the code for SUB *-, E, 0.

Signed off 01/14/88 in release Z01.02

Number: D200081620 Product: USER DEF ASSEMB 500 64851S001 01.60

One-line description:
expressions of the form 123456.78 cause errors

Problem:
There is a problem with the expression handler on the hosted software when parsing expressions of the form 12345.67, which the assembler thinks is a real number. This problemis shown in sample code supplied to Dave Ritchie by JLO in conjunction with the 64180 assembler.

Signed off 01/14/88 in release Z02.10

Number: D200081638 Product: USER DEF ASSEMB VAX 64851S003 01.60

One-line description:
expressions of form 123456.78 cause errors

Problem:
There is a problem with the expression handler on the hosted software when parsing expressions of the form 12345.67, which the assembler thinks is a real number. This problemis shown in sample code supplied to Dave Ritchie by JLO in conjunction with the 64180 assembler.

Signed off 01/14/88 in release Z02.10

Number: 5000241562 Product: USER DEF EMULATION 64274 01.05

One-line description:
run until <addr> fails from reset when reset points to user code.

Problem:
If the reset vector points to user code, the command "run until <addr>" from a reset state may not work correctly. The expected result is that the emulator will start executing user code and when the address is found by the analyzer, a break into the monitor will occur.

What may happen is that the emulator will break into the monitor as a result of the analysis break from the "run until <addr>" command. Following this, an "exit monitor" command is given and the emulator starts running user code again without an analyzer break pending. The condition that will cause this command to work incorrectly is that the "until" address must not be accessed within approximately 30 mS of the release from reset.

The generic algorithm that the UDE software uses for a "run until <addr>" command is as follows:

1) The analyzer is set up with the break condition for the until address and it is then started.

2) The processor is released from reset.

3) An ARE_YOU_THERE monitor command is executed by the host to determine if the emulator is running in the monitor or running user code.

4) If in the monitor, an EXIT monitor command is given. No test is made to determine if the break condition occurred to get into the monitor.

   If running user code, nothing is done.

The incorrect behaviour is caused by step 4. Since there is no test to determine if the analyzer break caused entry into the monitor, the emulation software assumes that it must exit to user code so it issues the EXIT command when it should not.

Temporary solution:
A very reliable and easy to use workaround can be set up with a command file. In a command file called RU (for "run until"), put the following instructions.

```
&PARMS ADDRESS
trace before &ADDRESS break_on trigger
run
```

This command file performs exactly the same function as "run until <address>". It is invoked by typing

```
RU xxxx
```

where xxxx is the "until" address.

Signed off 01/14/88 in release Z01.06

Number: D200043828 Product: USER DEF EMULATION 64274 01.04

One-line description:
Bad display of trace data with 8-bit UDE

Problem:
With 8-bit UDE emulators, the trace, when displayed, gives the wrong data at odd addresses. What is seen in the trace is the same data in the odd address as was captured for the preceeding even address. Note that this problem is only with version 1.04 of the UDE software, the previous version 2420 does not exhibit the problem.

Temporary solution:
Changing the UDE Configuration file parameter "OPCODE_SIZE" to WORD will partially solve this problem. Changing the parameter to WORD, should allow the internal analyzer to display each data field, however, the data will be visually offset on the HP64000 display. Changing the "OPCODE_SIZE" to equal WORD may adversely affect the inverse assembly of opcodes during single-stepping.

Signed off 01/14/88 in release Z01.06

Number: D200046623 Product: USER DEF EMULATION 64274 01.04

One-line description:
modify memory starting at odd addresses does not always work

Signed off 01/14/88 in release 401.06

Number: 1650036525 Product: USER INTERFACE 300 64808S004 01.20

One-line description:
Pmon flags a syntax error when attempting to append files

Problem:
pmon flags a syntax error for

```
cat file >> file2
          ^
```

But this a valid command and necessary to append i. e. a reloc file to a library.

Temporary solution:
Workaround:

Use

```
!cat file >> file2
```

In this case however the pmon softkeys can't be used.

Signed off 01/14/88 in release Z02.10

Number: 1650038521 Product: USER INTERFACE 300 64808S004 02.00

Keywords: MENUS

One-line description:
Pmon command completion via shell variables not working correctly

Problem:
Pmon generates collision errors in command completion between the external shell variables (whether set externally or by internal default) and the "default" pmon commands when they are the same.

The problem appears with all commands controlled by external shell variables.

Example: PMON_COMPILE not set, default pmon compile is "comp". If the user types co<tab>, pmon should complete to "comp". However, the error ERROR: possible tokens: comp, comp is generated.

Signed off 01/14/88 in release Z02.10

Number: D200077495 Product: USER INTERFACE 300 64808S004 01.20

One-line description:
User softkey display should be erased after shell escape.

Problem:
Display needs to be cleaned up when executing certain commands.
In one case the softkeys need to be erased before going on.

Signed off 01/14/88 in release Z02.10

Number: D200080135 Product: USER INTERFACE 300 64808S004 01.20

One-line description:
pwd truncates the /net/system portion of the path when RFA'ed to system.

Problem:
When using the HP 64000-UX products and netunaming across the LAN to another system, such as a compile server, the HP-UX command "pwd" which is used by the HP64000-UX product to tell what the local directory is, truncates the "/net/system" part of the path.

This is a HP-UX operating system defect. It is not a defect in the HP 64000-UX application software. As soon as this defect is fixed in HP-UX, it will work correctly when using the HP 64000-UX applications.

Signed off 01/14/88 in release Z02.10

Number: D200080721 Product: USER INTERFACE 300 64808S004 01.20

One-line description:
Pmon command completion intermittantly fails after completion error.

Problem:
Pmon evidently sets a flag on a command completion error which inhibits further errors on the same completion. Once the user types more characters and/or starts a new command, the flag should be reset such that new completion requests will be processed.

Unfortunately, the flag is not consistently reset, so command completions after a completion error will intermittantly have no effect.

Temporary solution:
Use softkeys or, when command completion fails with no message of any kind, add more characters of command and try again. Most errors occur only at the one or two character level.

Signed off 01/14/88 in release Z02.10

Number: D200081141 Product: USER INTERFACE 300 64808S004 01.20

One-line description:
Command search algorithm should match the softkey package.

Problem:
Pmon will not properly execute a command file which is not in the current directory. If the command file is in the search path specified in the PATH variable, a shell is forked to execute the command file. The shell will refuse to execute the command file, presumably since pmon command files are not executable.

Temporary solution:
Use only command files which are in the current directory.

Signed off 01/14/88 in release Z02.10

Number: 5000170654 Product: Z80/NSC800 C 64824 01.03

One-line description:
Error using switch (*x++).

Problem:
The following program hangs during complation on the 64100 and flags an error on the 9000/500 hosted compiler. The 64100 displays "STATUS: comp iling C/Z80 in pass 2 Line #8, Errors=0". The error message from the 90 00/500 is "line #8 -- pass 2 error #1006, 1006: Compiler Error. Contact Hewlett Packard. The program is as follows:

```
"C"
"Z80"
/* program with switch problem */
main()
{
    char *chrptr,*chrptr2;
    *chrptr='a';
    switch (*chrptr++)  {
    case 'a':
         *chrptr='z';
    case 'b':
         *chrptr='y';
    }
}
```

Temporary solution:
The work around is to break the switch(*chrptr++) into two statements. The first is switch(*chrptr) and then after the switch statements do a *chrptr++.

Signed off 01/14/88 in release Z02.10

Number: 5000172825 Product: Z80/NSC800 C 64824 01.03

One-line description:
RETI is not generated when exiting an interrupt procedure.

Problem:
When using $INTERRUPT ON$ the compiler does not generate the RETI instruction of the Z80 microprocessor. This is crucial when designing with such Z80 peripheral chips as the Toshiba Z84C30 Counter/Timer because the chip expects to see this instruction inorder to terminate its interrupt cycle. If the compiler is going to push all of the registers when using $INTERRUPT ON$ then why not use the correct instruction when returning from the interrupt. One other point to be made is that the Z80 emulator 64253A takes into consideration this peripheral requirement when emulating out of emulation memory by enabling output buffers during emulation memory read cycles.

Signed off 01/14/88 in release Z02.10

Number: 5000173278 Product: Z80/NSC800 C   64824   01.03

One-line description:
Array is being placed in the PROG section rather than data.

Problem:
Compiler puts array that should be in DATA section in PROG section
Example:

```
"C"
"Z80"
char  array[12];
```

The above code when compiled creates an array of twelve bytes that will reside in the PROG section. This should be placed in the DATA section.

Temporary solution:
Generate an ASM_FILE and edit the ASMProcessor file to place the array under the DATA counter.

Signed off 01/14/88 in release Z02.10

Number: 5000231605 Product: Z80/NSC800 C   64824   01.04

One-line description:
+= operator does not work for pointers to structures.

Problem:
Bad code generated when plus equals, times equals or divide equals (+=, *=, /=) with pointers to structure elements are used. Example:

```
"C"
"Z80"
$SEPARATE ON$
struct   fb  {   int   i;  int   size; } x, y;
main(){
   struct  fb *a,*b;
   a = &x;
   b = &y;
   b -> size =3;
   a -> size =4;
   if( b != a )  /* removing this line eliminates the problem */
       a -> size += b -> size;  /* wrong value stored in a-> size */
}
```

The result of a += is actually a->size = &a->size + b->size

Temporary solution:
Expand the expression as follows:

```
a->size = a->size + b->size;
```

Signed off 01/14/88 in release Z02.10

Number: 5000233866 Product: Z80/NSC800 C   64824   01.04

One-line description:
ZDconvert library module has two errors in the ZDdwordtoword routine.

Problem:
The ZDconvert module contains two errors in the ZDsdwordtoword conversion utility.
These errors cause two problems:

First, if the data to be converted is located at an adress above or equal to 8000H the routine generates an overflow error trap, although there actually is no overflow.

Second, if the data to be converted is negative, the stack gets misalligned, which will cause unpredictable results.

The problem is restricted to the "debug" conversion routine, i.e. it will only occur if $DEBUG ON$ is set in the source program.

See the corrected Library Routine below for details:
ZDsdwordtoword:

```
         CALL  Zsave_address
         PUSH  AF
         PUSH  DE
         LD    E,[HL]
         INC   HL
         LD    D,[HL]
* check for overflow
         LD    A,H        ! ERROR  1 !   should be  LD A,[HL]
         OR    A
         JP    M,NEG_DWORD
         INC   HL
         .
         .
         .
         RET
NEG_DWORD  INC HL
           LD  A,[HL]
           INC HL
           AND [HL]
           EX  DE,HL
           CP  -1
           CALL NZ,Zoverflow
           RET                  ! ERROR 2 !
                                 POP DE
                                 POP AF
                                 must be included before the return
                                 to be consistent with the POP's on
                                 entry of ZDsdwordtoword, because this
                                 is the second return point of this
                                 routine !
```

Temporary solution:
No temporary solution, however you can turn $DEBUG OFF$.

Signed off 01/14/88 in release Z02.10

```
Number: D200038778  Product: Z80/NSC800 C          64824              01.01
```

One-line description:
Incorrect transfer address when linking 9 or more files.

Problem:
I have files submitted by a customer which have the following problem. If you compile and assemble these files (most are C programs, a couple are assembly code) on the vax they will not link properly. What happens is the linker reports transfer address at loc XXXX defined by Zlibrary The address it reports is different than the address of the ENTRY label and in fact is outside of the module Zlibrary altogether. The customer said he had this problem when he linked eight or more files of any size. I cannot duplicate this with files I create, but, I can duplicate the problem with the files he sent me. Incidently, if the files are compiled on the 64000 and uploaded (relocs uploaded) they will link successfully.

Temporary solution:
Compile the files on the 64000 and upload the relocatables to the VAX. The 64000 generated reloc's will link successfully.

Signed off 01/14/88 in release Z02.10

```
Number: D200071373  Product: Z80/NSC800 C          64824              01.03
```

One-line description:
INT Multiplication of short by negative constant with SHORT_ARITH.

Problem:
When the compiler directive $SHORT_ARITH ON$ is in effect multiplication of byte sized quantities by negative constants will produce code that extends the byte to int size and then preforms an int size multiplication operation. The following code illustrates:

```
$SHORT_ARITH ON$

short s;
unsigned short us;

main ()
{
   if (s*-4)     /* s is extended to int and a word mul performed */
      error();
   if (us*0xfc)  /* us is extended to int and a word mul performed */
      error();
}
```

Signed off 01/14/88 in release Z02.10

```
Number: D200071431  Product: Z80/NSC800 C          64824              01.03
```

Keywords: PASS 1

One-line description:
DIV, MOD and COMParisons may do unsigned estend of signed values

Problem:
Conditionals that employ div, mod, or comparison operations may not correctly extend signed short values to int size if the other operand is an unsigned short or char. For example, in the following code s is extended as if it were declared an unsigned short.

```
$SHORT_ARITH OFF$

short s;
unsigned short us;

main()
{
   if ((s/us)^0xffff)     /* both s and us get unsigned extend */
      error();
   if ((us%s)^0x007f)     /* both s and us get unsigned extend */
      error();
   if (us==s)             /* both s and us get unsigned extend */
      error();
   if (s!=us)             /* both s and us get unsigned extend */
      error();
   if (s<us)              /* both s and us get unsigned extend */
      error();
   if (s>us)              /* both s and us get unsigned extend */
      error();
}
```

Signed off 01/14/88 in release Z02.10

```
Number: D200075044  Product: Z80/NSC800 C          64824              01.04
```

One-line description:
.

Problem:
Z80 COMPILER GENERATING INCORRECT CODE FOR THE FOLLOWING PROGRAM.

```
"C"
"Z80"

INT   (*BGETVC()) ();

GETC(PARM)
INT PARM;

{  INT (*TEST)();
   TEST = BGETVC(10);
   RETURN((TEST)(11,PARM));
}
```

TEST CONTAINS AN ADDRESS WHICH POINTS TO A FUNCTION. WHEN IT IS ASSIGNED TO, THE CODE ASSIGNS THE CORRECT ADDR OF THE FUNCTION. HOWEVER, WHEN THE COMPILER GENERATES CODE TO CALL THE FUNCTION VIA TEST IT DOES NOT ACCESS TEST. INSTEAD IT ACCESSES A LOCATION TWO BYTES LOWER IN MEMORY.

Signed off 01/14/88 in release Z02.10

```
Number: D200077222  Product: Z80/NSC800 C          64824              01.04
```

Keywords: CODE GENERATOR

One-line description:
Floating point division of 2 constants generates incorrect result

Problem:
Compiler generates incorrect code for evaluation of double division:

```
"C"
"8088"
main()
{
      double   xx;
      xx = 2.0/3.0;
      xx = 2.0;
}
```

xx is assigned the value 2.0 by both statements.

This problem also occurs with other variable types such as float, long. Any constant divided by a constant will generate this error.

Temporary solution:

```
   xx = 2.0/y;   where y = 3.0;
```

Signed off 01/14/88 in release Z02.10

```
Number: D200079079  Product: Z80/NSC800 C          64824              01.04
```

One-line description:
+=, -=, *=, & /= may fail to auto vars with $RECURSIVE ON$

Problem:
Text:
  +=, -=, *=, & /= may fail to auto vars with $RECURSIVE ON$

SUBMITTER TEXT:

Composite assignment operators may fail to automatic variables when $RECURSIVE ON$ is in effect. This problem results from the compiler failing to keep track of how many bytes of parameters have been pushed onto the stack (and then popped off) for runtime library routines. The effect of this failure is an incorrect stack location being updated. The following program segment illustrates this problem.

```
"C"
"8085"
$RECURSIVE ON$

func(i1,i2,doub)
int i1,i2;
double doub;
{
   int answer;

   answer = 1;

   answer += i2*x;  /* after this statement answer still is 1 */
                    /* however i1 = i2 * x                    */
}
```

Signed off 01/14/88 in release Z02.10

```
Number: D200080374  Product: Z80/NSC800 C          64824              01.04
```

One-line description:
Warning message text is incorrect.

Problem:
68000 C compiler, Just updated to 2.07.

Warning 521: Unsigned integer to real conversion treated as signed.
Is incorrect.
The wording should imply that the conversion should be going the other w ay, from real to unsigned integer.

To get the error:

```
"C"
"68000"
unsigned int a;
main()
{
a=0.0;
}
```

NOTE: this error message is not in the manuals.

Temporary solution:
If you do not want to see this message you may specify $WARN OFF$. This will turn off all warning messages.

Signed off 01/14/88 in release Z02.10

```
Number: D200081471  Product: Z80/NSC800 C          64824              01.04
```

One-line description:
MOD operation returning the wrong value.

Problem:
The MOD operator is returning the wrong value.

"BZ80"

```
PROGRAM TEST;
$EXTENSIONS ON$
$GLOBVAR ON$

VAR  I1,I2,I3:  INTEGER;

BEGIN

   I1 := 5280;
   I2 := 1000;
   I3 := I1 MOD I2;

END.
```

The result of the mod operation is 4280 decimal.

Temporary solution:
No temporary solution at this time.

Signed off 01/14/88 in release 402.10

Number: 5000226605 Product: Z80/NSC800 C   300 64824S004   01.20

One-line description:
Double word divide library returning incorrect result.

Problem:
Zsdworddiv calculates incorrectly. The result calculated becomes 1,
if it was incorrect.
Zsdworddiv is correct on 64000. This is the problem on 9000/300.
The example is as followes,

```
   "C"
   "Z80"
          long a,b,ldiv;
          main()
          {
              a=70000;
              b=150;
              ldiv = a/b; <-------- ldiv must be 01D2H but it is 1 !
          }
```

Temporary solution:
No temporary solution at this time.

Signed off 01/14/88 in release Z02.10

Number: 1650011585 Product: Z80/NSC800PASCAL 64823 01.01

Keywords: PASS 2

One-line description:
Incorrect code generated when set elements are passed as parameters.

Problem:
Incorrect code is generated when sets are passed as parameters.
The stack pointer is manipulated so that the program "goes in the
weeds" after the call to the procedure. The following code is
an example:

```
"processor name"
$SEPARATE ON$
$EXTENSIONS ON$
TYPE
  Letters = (a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,r);
  Set_of_Letters = SET OF Letters;
$GLOBPROC ON$
PROCEDURE Letters_Pas(Received:Set_of_Letters):EXTERNAL;
PROCEDURE Init_Set;
BEGIN
  Letters_Pas([]);     (*Code generates an extra INC SP after the
END;                      call to Letters_Pas*)
$GLOBPROC OFF$
.
```

Temporary solution:
Any set size which is not equal to 3 bytes will work correctly.

Signed off 01/14/88 in release Z01.90

Number: 5000161000 Product: Z80/NSC800PASCAL 64823 01.03

One-line description:
Unbelieveable amount of library code linked for no-line program.

Problem:
The following no-line program yields a 4000 byte absolute file
when linked with Zlibrary and Zreallib.

```
"BZ80"
PROGRAM LOTS_OF_CODE;
PROCEDURE REAL_ADD; EXTERNAL;
BEGIN
END.
```

Several modules that seem to be unnecessary (e.g. REAL_ADD,REAL_ATAN,
REAL_MUL) are loaded. Since the address space of the Z80 is only
64K, the libraries should be written in such a way to minimize the
code loaded.

Signed off 01/14/88 in release Z01.90

Number: 5000161034 Product: Z80/NSC800PASCAL 64823 01.03

One-line description:
Libraries reference procedures not actually needed.

Problem:
The following code requires the lib, reallib, piolib, and simlib
to be linked so that no linker errors are generated.

```
"BZ80"
PROGRAM SIM;
VAR
    REAL_S : STRING;
    X : REAL;
    P,Q : INTEGER;
BEGIN
    P := 1;
    STRWRITE(REAL_S,P,Q,X);
    STRREAD(REAL_S,P,Q,X);
END.
```

Linking all of the libraries causes the absolute file to be
17000 bytes long.

Signed off 01/14/88 in release Z01.90

Number: 5000163287 Product: Z80/NSC800PASCAL 64823 01.04

One-line description:
Code generated by compiler increased 12% with latest version.

Problem:
Latest revision of the compiler generates approximately 12% more
code then the previous revision did, according to this customer.
The following example was submitted showing a piece of code that
generated 2 lines of code with the previous version, and now
generates 7 lines of code with the most recent version.

```
"BZ80"
PROGRAM DUMMY;
CONST
  C_S = 32;
TYPE
  P1 = RECORD
       C : SIGNED_16;
       END;
VAR
  LED : P1;

PROCEDURE A;
BEGIN
  LED.C := C_S;  (*Generates 7 lines of code, used to generate 2*)
END;

BEGIN
  LED.C := C_S;
END.
```

Signed off 01/14/88 in release Z01.90

Number: 5000182014 Product: Z80/NSC800PASCAL 64823 01.03

One-line description:
FOR statement with SIGNED_BYTE produces incorrect code.

Problem:
Bad code generated when FOR statement requires a signed byte
mod library call. Example:

```
"BZ80"
$EXTENSIONS ON$
VAR
    A,B:BYTE;
BEGIN
    FOR A:=(B MOD 64) TO 100 DO
            ;
END.
```

The problem is that a temporary storage location is set up for the
FOR loop counter, but after the initial value of the mod is
calculated, register A is loaded from this location temporary storage
location instead of being stored there.

Temporary solution:
Use UNSIGNED_8 instead of BYTE, or declare a temporary variable
to hold the result of the mod operation.

Signed off 01/14/88 in release Z01.90

Number: 5000186742 Product: Z80/NSC800PASCAL 64823 01.03

Keywords: ADDR

One-line description:
ADDR(x) generates incorrect code is x is of type BYTE.

Problem:
Incorrect code generated when using ADDR function if taking the
address of a local variable of type "BYTE". EXAMPLE:

```
"BZ80"
PROGRAM TEST;
$EXTENSIONS ON$
$RECURSIVE ON$

PROCEDURE RUN;
VAR
    ONE:BYTE;
    ONE_POINTER:^BYTE;
BEGIN
    ONE_POINTER:=ADDR(ONE);
END;
.
```

An incorrect value is written to ONE_POINTER

Temporary solution:
Use $RECURSIVE OFF$, or declare ONE_POINTER outside the procedure,

or use type INTEGER instead of BYTE.

Signed off 01/14/88 in release Z01.90

Number: 5000190629 Product: Z80/NSC800PASCAL 64823 01.04

One-line description:
Bad code generated when CASE expression involves addition of two bytes.

Problem:
Bad code generated when CASE statement expression is the addition
of two BYTEs:

```
"BZ80"
$EXTENSIONS+$
PROGRAM BUG_PROG;
PROCEDURE BUG;
VAR
       A,B,C:BYTE;
BEGIN
       A:=1; B:=2;
       CASE A+B OF
            0 : C:=0;
            1 : C:=1;
            2 : C:=2;
            3 : C:=3; OTHERWISE C:=100; END END;
```

Temporary solution:
1. Store the result of A+B in a temporary value, then CASE on that
temporary value.
2. Use INTEGER(A+B)
3. Declare A and B as type INTEGER

Signed off 01/14/88 in release Z01.90

Number: 5000217927 Product: Z80/NSC800PASCAL 64823 01.04

One-line description:
Signed_32 divide returns wrong result.

Problem:
The library Zsdworddiv improperly handles some signed 32 bit
divisions. Example:

```
"BZ80"
PROGRAM DIVIDE;
VAR
    B1,B2,B3:SIGNED_32
BEGIN
    B1:=362700;
    B2:=2000;
    B3:=B1/B2;   {the result returned is one}
END
.
```

Temporary solution:
No temporary solution at this time.

Signed off 01/14/88 in release Z01.90

Number: D200063875 Product: Z80/NSC800PASCAL 64823 01.03

One-line description:
functional type change of a constant into multi-byte structure gen's err

Problem:
Functional type casting of a constant into a multi-byte structure generates bad data.

```
"processor"

PROGRAM BAD_DATA;

TYPE  EVENT = RECORD
          A   : BYTE;
          B   : BYTE;
          C   : INTEGER;
          D   : BYTE;
      END;

VAR   EVENT1   : EVENT;


PROCEDURE   GENERATOR();
   BEGIN
      EVENT1 := EVENT(0);  { THIS ASSIGNMENT RESULTS IN BAD DATA }
   END;

BEGIN
END.
```

Temporary solution:
No temporary solution at this time.

Signed off 01/14/88 in release Z01.90

Number: D200066761 Product: Z80/NSC800PASCAL 64823 01.03

One-line description:
Code generated by compiler increased 12% with latest version.

Problem:
Latest revision of the compiler generates approximately 12% more code then the previous revision did, according to this customer. The following example was submitted showing a piece of code that generated 2 lines of code with the previous version, and now generates 7 lines of code with the most recent version.

```
"BZ80"
PROGRAM DUMMY;
CONST
  C_S = 32;
TYPE
  P1 = RECORD
       C : SIGNED_16;
```

```
       END;
VAR
  LED : P1;

PROCEDURE A;
BEGIN
  LED.C := C_S;  (*Generates 7 lines of code, used to generate 2*)
END;

BEGIN
  LED.C := C_S;
END.
```

Signed off 01/14/88 in release Z01.90

Number: D200071332 Product: Z80/NSC800PASCAL 64823 01.03

One-line description:
Links not correctly established during calls of nested procedures.

Problem:
Calls between procedures at different nesting levels may not correctly establish links needed for referencing higher level variables. The following code illustrates the problem.

```
"PASCAL"
"BZ80"
$EXTENSIONS ON$
$RECURSIVE ON$

PROGRAM BUG1018B;

PROCEDURE TEST1018B;
VAR
   LOCAL1018 : INTEGER ;

   PROCEDURE INC_LOCAL ;
   BEGIN
      LOCAL1018 := LOCAL1018 + 1;
   END ;

   PROCEDURE NEST1 ;
   VAR
      DUMMY1 : INTEGER ;

      PROCEDURE NEST2 ;
      VAR
         DUMMY2 : INTEGER ;
      BEGIN  { NEST2 }
         INC_LOCAL ; (* variable local is NOT correctly incremented *)
      END ;  { NEST2 }

   BEGIN  { NEST1 }
      INC_LOCAL ;  (* variable local is correctly incremented *)
      NEST2 ;
   END ;  { NEST1 }
```

```
BEGIN  { TEST1018B }
   LOCAL1018 := 0 ;
   NEST1 ;
END ;  { TEST1018B }
.
```

Signed off 01/14/88 in release Z01.90

Number: D200071340 Product: Z80/NSC800PASCAL   64823   01.03

One-line description:
Certain variable accesses by nested procedures may not work

Problem:
In certain situations where a local variable of one procedure is referenced by another procedure nested inside the first procedure, the register loaded with the link may be walked on. The following code illustrates the problem.

```
"PASCAL"
"BZ80"
$EXTENSIONS ON$
$RECURSIVE ON$

PROGRAM BUG1018A;

PROCEDURE LEVEL1 ;
VAR
   LEV1_1, LEV1_2 : INTEGER ;

   PROCEDURE LEVEL2 ;
   VAR
      ARR1 : STRING ;
      INDEX : INTEGER ;

   BEGIN { LEVEL2 }
      ARR1[INDEX] := CHR (LEV1_2 + 1) ;  { access to LEV1_2 incorrect }
   END ; { LEVEL2 }

BEGIN { LEVEL1 }
   LEVEL2 ;
END ; { LEVEL1 }
.
```

Signed off 01/14/88 in release Z01.90

Number: D200071365 Product: Z80/NSC800PASCAL   64823   01.03

Keywords: PASS 1

One-line description:
Functional type changes not always evaluated correctly

Problem:
Some functional type changes are not correctly evaluated. For example,

the following code illustrates the problem.

```
$EXTENSIONS ON$
PROGRAM PTEST;

VAR
    S8  : SIGNED_8 ;
    U8  : UNSIGNED_8 ;
    S16 : SIGNED_16 ;
    U16 : UNSIGNED_16 ;

BEGIN
    U16 := UNSIGNED_16(S8);  (* signed extension of S8 - correct *)
    U16 := UNSIGNED_8(S8);   (* signed extension of S8 - incorrect *)

    S16 := SIGNED_16(U8);    (* unsigned extension of U8 - correct *)
    S16 := SIGNED_8(U8);     (* unsigned extention of U8 - incorrect *)
END.
```

Signed off 01/14/88 in release Z01.90

Number: D200071423 Product: Z80/NSC800PASCAL   64823   01.03

One-line description:
Function Calls via pointer may fail

Problem:
Function calls via pointers may fail. The following code sample illustrates the problem:

```
typedef int (*PFI)();

extern int code_array[100];  /* code_array is actually a function */

main()
{
   (*((PFI)code_array))();    /* this call fails to transfer
                                 control to code array */
           LXI H,main01_0
           PUSH H
           LHLD code_array    /* the instruction should be
                                 LXI H,code_array */
           PUSH H
           RET
        main01_0
              .
              .
              .
}
```

Temporary solution:

```
typedef int (*PFI)();
PFI func_ptr;
extern int code_array[100];  /* code_array is actually a function */

main()
```

```
{
   func_ptr = code_array;
   (*func_ptr)();            /* call to code_array is correct */
}
```

Signed off 01/14/88 in release Z01.90

Number: D200073031 Product: Z80/NSC800PASCAL   64823   01.03

One-line description:
Pascal does not report error for assignment of constant to structure

Problem:
The Pascal/64000 compiler fails to report an error when using the functional type change operator to attempt to assign an immediate constant to a multi-byte structure.

Since the Pascal/64000 compiler does not support structured constants, there is no meaningful way to assign a constant to a structure. Each element must be assigned individually.

The Pascal/64000 compiler does report an error 505 (Warning: type changes physical size), when it should generate a fatal error. It tries to generate code for the illegal statement which will not produce the results expected by the user.
The compiler should produce fatal Error #451: Structured constants not implemented.

Here is a simple example and the workaround by explicit individual assignment statements.

```
"PASCAL" PREPROCESS
"6809"
{ Test program to demonstrate Pascal language defect }
{ Functional type change of constant to multi-byte variable }
PROGRAM PTEST101;
$EXTENSIONS ON$
 TYPE event = RECORD
               type     : BYTE;
               qualifier: BYTE;
               msg      : INTEGER;
               send_task: BYTE;
             END;
 VAR event1: event;
      i: INTEGER;
      R: REAL;

 BEGIN

{The following code is attempting to initialize}
{ the multibyte record event to zeros. }
{It should be interpreted as a Pass 1 error }
{ Error #451: Structured constants not implemented}
{ The code produced will be processor dependent }
```

```
    event1 := event(0);  {This code is incorrect Pascal}


 {Correct Pascal using individual assignments}
    event1.type:=0;
    event1.qualifier:=0;
    event1.msg:=0;
    event1.send_task:=0;
 END.
```

Signed off 01/14/88 in release Z01.90

Number: D200076067 Product: Z80/NSC800PASCAL   64823   01.04

One-line description:
Unsigned_8 treated as signed value in FOR loop test.

Problem:
Assigning a constant to an unsigned_8 variable whose upper bit is set causes problems. Specifically, when the unsigned_8 var is used later it is treated as a signed value. In the example below, an unsigned_8 is assigned 247 decimal at the top of a FOR loop. When the compiler compares it is does a byte compare and therefore interprets the unsigned_8 as a signed quantity.

```
"processor"

$EXTENSIONS ON$

PROGRAM DOLOOP;

VAR   SECTORNUM,STOPSECTOR    : UNSIGNED_8;
      A                       : INTEGER;

BEGIN
    STOPSECTOR := UNSIGNED_8(247);

    FOR  SECTORNUM := UNSIGNED_8(0) TO STOPSECTOR DO BEGIN

         A := 5;

    END;

END.
```

Temporary solution:
USE AN UNSIGNED_16 FOR THE CONTROLLING VAR.

```
"PROCESSOR"

$EXTENSIONS ON$

PROGRAM DOLOOP;

VAR    SECTORNUM,STOPSECTOR   :  UNSIGNED_16;
```

```
       A                         :  INTEGER;


BEGIN

     STOPSECTOR := UNSIGNED_16(247);

     FOR SECTORNUM := UNSIGNED_16(0) TO STOPSECTOR DO BEGIN

     A := 5;
     END;
END.
```

This works for values up to 8000H.

Signed off 01/14/88 in release Z01.90

Number: D200079301 Product: Z80/NSC800PASCAL   64823   01.04

One-line description:
Compiler may confuse similar parameters in different subroutines

Problem:
The Z80 & 8085 B Pascal compilers may confuse similar parameters in different (but nested) subroutines. The following program illustrates this problem. The problem stems from the compiler searching for a parameter in a register. It finds one with the correct attributes (position, size, indirects, data type, etc), but it is unfortunately not from the correct subroutine.

```
"PASCAL" PREPROCESS
"BZ80"
$EXTENSIONS ON$
$SEPARATE ON$
$RECURSIVE ON$

PROGRAM PTEST104;

PROCEDURE TEST1018; { Test problems in Pascal Scoped_ACCESSES;}

 PROCEDURE  Level1(l1p1,l1p2:BYTE);
    VAR
          l1v1,l1v2:  BYTE;

  PROCEDURE  Level2(l2p1,l2p2:BYTE);
      VAR
            l2v1,l2v2:  BYTE;

   BEGIN  {Level2}
    l2p2:=l1v2;
    l1v1:=l1p2; { ERROR l1v1 gets the value of l2p2 rather than l1p1. }
   END;         { The value of l2p2 was in a register from the        }
                { previous assignment.

 BEGIN  {Level1}
```

```
 END;

BEGIN  {TEST1018}
END;
.
```

Temporary solution:
No temporary solution.

Signed off 01/14/88 in release 301.90

Number: 5000224204 Product: Z80/NSC800PASCAL VAX 64823S003   01.70

One-line description:
Nested external procedure call causes bad code to be generated.

Problem:
Bad code is generated when a procedure which is declared external
within a second procedure that passes parameter(s) to a third.
Example:

```
"BZ80" PREPROCESS
PROGRAM HAROLD;
$EXTENSIONS+$
VAR   B:BYTE;
PROCEDURE ERROR ;
   PROCEDURE CLR_PRTB( MASK : BYTE ) ; EXTERNAL :
BEGIN
    CLR_PORTB( B ) ;
END;
.
```

Temporary solution:
WORKAROUND: declare PROCEDURE CLR_PORTB(MASK:BYTE);EXTERNAL;
in the outer scope of the module, not inside the procedure.

Signed off 01/14/88 in release Z01.90

Number: D200076760 Product: Z8000 C   64820   01.06

One-line description:
Array is being placed in the PROG section rather than data.

Problem:
Compiler puts array that should be in DATA section in PROG section
Example:

```
"C"
"Z80"
char  array[12];
```

The above code when compiled creates an array of twelve bytes that will
reside in the PROG section. This should be placed in the DATA section.

Temporary solution:
Generate an ASM_FILE and edit the ASMProcessor file to place
the array under the DATA counter.

Signed off 01/14/88 in release Z02.10

Number: D200077107 Product: Z8000 C   64820   01.06

Keywords: CODE GENERATOR

One-line description:
Floating point division of 2 constants generates incorrect result

Problem:
Compiler generates incorrect code for evaluation of double division:

```
"C"
"8088"
main()
{
      double   xx;
      xx = 2.0/3.0;
      xx = 2.0;
}
```

xx is assigned the value 2.0 by both statements.

This problem also occurs with other variable types such
as float, long. Any constant divided by a constant will
generate this error.

Temporary solution:
```
   xx = 2.0/y;   where y = 3.0;
```

Signed off 01/14/88 in release Z02.10

Number: D200079137 Product: Z8000 C   64820   01.03

Keywords: PASS 1

One-line description:
DIV, MOD and COMParisons may do unsigned estend of signed values

Problem:
Conditionals that employ div, mod, or comparison operations may not correctly extend signed short values to int size if the other operand is an unsigned short or char. For example, in the following code s is extended as if it were declared an unsigned short.

```
$SHORT_ARITH OFF$

short s;
unsigned short us;

main()
{
    if ((s/us)^0xffff)      /* both s and us get unsigned extend */
        error();
    if ((us%s)^0x007f)      /* both s and us get unsigned extend */
        error();
    if (us==s)              /* both s and us get unsigned extend */
        error();
    if (s!=us)              /* both s and us get unsigned extend */
        error();
    if (s<us)               /* both s and us get unsigned extend */
        error();
    if (s>us)               /* both s and us get unsigned extend */
        error();
}
```

Signed off 01/14/88 in release Z02.10

Number: D200080341 Product: Z8000 C   64820   01.06

One-line description:
Warning message text is incorrect.

Problem:
68000 C compiler, Just updated to 2.07.

Warning 521: Unsigned integer to real conversion treated as signed.
Is incorrect.
The wording should imply that the conversion should be going the other w
ay, from real to unsigned integer.

To get the error:

```
"C"
"68000"
unsigned int a;
main()
{
a=0.0;
}
```

NOTE: this error message is not in the manuals.

Temporary solution:
If you do not want to see this message you may specify
$WARN OFF$. This will turn off all warning messages.

Signed off 01/14/88 in release Z02.10

Number: D200063834 Product: Z8000 PASCAL 64816 01.11

One-line description:
functional type change of a constant into multi-byte structure gen's err

Problem:
Functional type casting of a constant into a multi-byte structure generates bad data.

"processor"

```
PROGRAM BAD_DATA;

TYPE  EVENT = RECORD
          A   : BYTE;
          B   : BYTE;
          C   : INTEGER;
          D   : BYTE;
      END;

VAR   EVENT1   : EVENT;


PROCEDURE   GENERATOR();
   BEGIN
      EVENT1 := EVENT(0);  { THIS ASSIGNMENT RESULTS IN BAD DATA }
   END;

BEGIN
END.
```

Temporary solution:
No temporary solution at this time.

Signed off 01/14/88 in release Z01.90

Number: D200076026 Product: Z8000 PASCAL 64816 01.12

One-line description:
Unsigned_8 treated as signed value in FOR loop test.

Problem:
Assigning a constant to an unsigned_8 variable whose upper bit is set causes problems. Specifically, when the unsigned_8 var is used later it is treated as a signed value. In the example below, an unsigned_8 is assigned 247 decimal at the top of a FOR loop. When the compiler compares it is does a byte compare and therefore interprets the unsigned_8 as a signed quantity.

"processor"

```
$EXTENSIONS ON$

PROGRAM DOLOOP;

VAR   SECTORNUM,STOPSECTOR    : UNSIGNED_8;
      A                       : INTEGER;
```

```
BEGIN
   STOPSECTOR := UNSIGNED_8(247);

   FOR  SECTORNUM := UNSIGNED_8(0) TO STOPSECTOR DO BEGIN

        A := 5;

   END;

END.
```

Temporary solution:
USE AN UNSIGNED_16 FOR THE CONTROLLING VAR.

"PROCESSOR"

```
$EXTENSIONS ON$

PROGRAM DOLOOP;

VAR    SECTORNUM,STOPSECTOR    :  UNSIGNED_16;
       A                       :  INTEGER;


BEGIN

     STOPSECTOR := UNSIGNED_16(247);

     FOR SECTORNUM := UNSIGNED_16(0) TO STOPSECTOR DO BEGIN

     A := 5;
     END;
END.
```

This works for values up to 8000H.

Signed off 01/14/88 in release Z01.90

Number: D200079210 Product: Z8000 PASCAL 64816 01.12

One-line description:
Pascal does not report error for assignment of constant to structure

Problem:
The Pascal/64000 compiler fails to report an error when using the functional type change operator to attempt to assign an immediate constant to a multi-byte structure.

Since the Pascal/64000 compiler does not support structured constants, there is no meaningful way to assign a constant to a structure. Each element must be assigned individually.

The Pascal/64000 compiler does report an error 505 (Warning: type changes physical size), when it should generate a fatal error. It

tries to generate code for the illegal statement which will not produce the results expected by the user.
The compiler should produce fatal Error #451: Structured constants not implemented.

Here is a simple example and the workaround by explicit individual assignment statements.

```
"PASCAL" PREPROCESS
"6809"
{ Test program to demonstrate Pascal language defect }
{ Functional type change of constant to multi-byte variable }
PROGRAM PTEST101;
$EXTENSIONS ON$
 TYPE event = RECORD
                type     : BYTE;
                qualifier: BYTE;
                msg      : INTEGER;
                send_task: BYTE;
              END;
 VAR event1: event;
      i: INTEGER;
      R: REAL;

 BEGIN

{The following code is attempting to initialize}
{ the multibyte record event to zeros. }
{It should be interpreted as a Pass 1 error }
{ Error #451: Structured constants not implemented}
{ The code produced will be processor dependent }

     event1 := event(0);  {This code is incorrect Pascal}


 {Correct Pascal using individual assignments}
     event1.type:=0;
     event1.qualifier:=0;
     event1.msg:=0;
     event1.send_task:=0;
 END.
```

Signed off 01/14/88 in release Z01.90

Number: D200079277 Product: Z8000 PASCAL 64816 01.12

Keywords: PASS 1

One-line description:
Functional type changes not always evaluated correctly

Problem:
Some functional type changes are not correctly evaluated. For example, the following code illustrates the problem.

```
$EXTENSIONS ON$
PROGRAM PTEST;

VAR
    S8  : SIGNED_8 ;
    U8  : UNSIGNED_8 ;
    S16 : SIGNED_16 ;
    U16 : UNSIGNED_16 ;

BEGIN
    U16 := UNSIGNED_16(S8);  (* signed extension of S8 - correct *)
    U16 := UNSIGNED_8(S8);   (* signed extension of S8 - incorrect *)

    S16 := SIGNED_16(U8);    (* unsigned extension of U8 - correct *)
    S16 := SIGNED_8(U8);     (* unsigned extention of U8 - incorrect *)
END.
```

Signed off 01/14/88 in release Z01.90

Number: 5000137869 Product: Z8002 EMULATION 64233 01.07

One-line description:
Monitor displays wrong value for R15 and SSTK

Problem:
When displaying register contents, the value shown for R15 and SSTK is 6 (six) less than it should be. This is a result of the way the monitor is entered.

Temporary solution:
Modify the emulation monitor as follows:

```
 440  LD     MONR14L,R14
NEW   INC    R15,#6
 441  LD     MONR15L,R15
NEW   INC    MONSSTKL,#6
 .
 .
 709  *************************
NEW   DEC    MONSSTKL,#6
```

Signed off 01/14/88 in release Z02.01

Number: 5000151431 Product: Z8002 EMULATION 64233 02.00

One-line description:
User interrupts are not serviced for 17ms after analysis generated break

Problem:
Starting with revision 1.07 of the emulation operating software, the STOP line is asserted over a 17ms period with a duty cylce of 520us asserted, followed by 40us of time with the STOP line disasserted. Version 1.06 of the emulation software similarly asserted the STOP line but for only approximately 2.3ms with a 50% duty cyle of 40us on, followed by 40us off. The problem comes with the fact that the cpu will suspend all operations while the STOP line is asserted. So with version 1.07 of the software, there is a period of 17ms during which the cpu will only be active for 1.2ms If user interrupts occur more often than every 17ms and the interrupt service routine takes more than 1.2ms to complete, then user interrupts will be missed.
NOTE: This problem occurs only after an analysis generated break. Therefore you may only see the problem after "trace break_on trigger/measurment_complete", or "run until <state>".

Version 1.07 is essentially equivalent to version 2.00 with regard to this problem.

Temporary solution:
Contact your HP representative if you encounter this problem. He should be able to get you a copy of the software that solves this problem until the new revision of software is available.

Signed off 01/14/88 in release Z02.01

Number: D200071415 Product: Z8002 EMULATION 64233 02.00

One-line description:
Can't find symbols loaded with more address bits than specified.

Problem:
If the user links his code with more significant address bits than the number specified in the emulation configuration, the emulator will be unable to access the symbols, and the monitor may not function properly.

Signed off 01/14/88 in release Z02.01

HEWLETT
PACKARD

5958-6019, Dec. 1987
E1287

Printed in U.S.A.