| h | HEWLETT
|_p_| PACKARD

PORTABLE COMPUTER DIVISION 1000 N.E. Circle Blvd, Corvallis, OR 97330

HP-71B Known Bugs February 6, 1985

This document lists the known bugs in the HP-71B operating system, version 1BBBB. Included are all enhancement requests which, if implemented, would significantly improve certain areas of the software.

The bugs are listed in two parts:

Part 1 lists the bugs by severity. Included are a subjective rank of the seriousness of the bug, a description of each, and suggested workarounds.

Part 2 lists the bugs by STARS SR number -- or, equivalently, by reporting date. Included are a cross-reference to the severity classifications in Part 1, the priority of each bug, the date of reporting, appropriate comments, and a description.

Definitions:

UDF --
An abbreviation for User-Defined Function.

Severity --
"Severity" refers to the following classifications:

Major: A bug which is likely to be encountered by the user and which materially degrades the usefulness of the software, even for the user who has been alerted to the existence of the bug. It is not possible to fully recover from a major bug by:

1. using simple keystrokes; or
2. using other simple procedures (e.g., executing one program after another).

Minor: A bug which does not materially affect the usefulness of the software, provided the user has been alerted to

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

the existence of the bug and the ways to circumvent it. Minor bugs are corrected through:

1. simple keystrokes;

2. other simple procedures; or

3. explanations to clarify a procedure or outcome.

In other words, if there is a straightforward workaround for the bug, it is considered minor.

Trivial: A bug which does not materially affect the usefulness of the software, even for a user who has not been warned of the bug's existence.

Manual Change: Those code modifications which would require a change to the Owner's or Reference Manuals. The entries in this category will also be found in the "major", "minor" or "trivial" sections.

Rank --
"Rank" is our subjective assessment of the seriousness of a bug. It represents the expected number of users who would encounter the bug in one year's use of the HP-71, assuming that they had not been alerted of the bug's existence. For example, a rank of 30 would mean that out of 100 users of the HP-71, in a year's worth of normal programming about 30 of them would encounter the bug.

Priority --
"Priority" refers to our internal classification, on a scale of 1 to 5:
1= Critical -- Corrupts memory; cold starts.
2= Serious -- No workaround, gives undesirable results.
3= Guarded -- Workaround exists, but needs fix.
4= Cosmetic -- Nice feature, but may not fix.
5= Not fixed -- Usually a documentation change, not software.

STARS --
"STARS" stands for "Software Tracking and Reporting System", and is used to log and monitor the progress of all bug fixes. Each bug is identified with a STARS SR ("Service Request") number, such as SR#1082-7.

Date --
In Part 2, "Date" refers to the date the bug report was entered onto the STARS data base.

BY SEVERITY:

MAJOR (8)

	Number	Rank	Description & (Workaround)
1.	1155-1	60	Invalid leap year determination for even year in odd decade; DATE$ gives wrong date. (Don't use DATE$ after Feb 28; DATE$ virtually unusable.)
2.	1151-0	25	SUB statement chain field may look like a comment, tank machine when decompiled. (If a subprogram has line labels or UDFs, always include a comment after the SUB stmt. Otherwise, 1/256 chance that machine tanks.)
3.	1173-4	20	Assignment to substring such as A$[1234567]="xxx" cold starts if substring limit > 1048576. (In a substring assignment, screen out any substring limit > 65535, or NaN or Inf.)
4.	1112-2	10	Concatenating a string constant at low memory may cause cold start. (Check MEM at least > 2*length of string constant before concatenating.)
5.	1113-0	10	Concatenating a string variable at low memory may cause cold start. (Check MEM at least > 2*length of string variable before concatenating.)
6.	1162-7	7	Editing program clears SUB link before checking if EEPROM; bad keystroke can destroy EEPROM. (Make all EEPROMs SECURE or PRIVATE.)
7.	1122-1	2	Assignment to string array element with subscript out-of-range can tank. (If TRAP IVL is set to 2, don't assign values to string arrays. Don't assign a string value with length > 33740 bytes.)
8.	1123-9	2	Assignment to string array element in subprogram corrupts memory. (If TRAP IVL is set to 2, don't assign values to string arrays in subprograms.)

MINOR (29)

	Number	Rank	Description & (Workaround)
1.	1110-6	60	ON...GOTO performs ON...RESTORE if OPTION ROUND NEG or POS, or if index is a function. (Set OPTION ROUND NEAR or ZERO before ON...GOTO; make intermediate assignment for index.)
2.	1165-0	45	Sequential READ# from DATA file skips over EOF. (Use record size of 8 for numeric data; use TEXT files for string data. Or use random READ#.)
3.	1121-3	35	POS(<str1>,<str2>,0) always returns 0. (Ensure 3rd param in POS is nonzero.)
4.	1119-7	20	VAL, TRANSFORM and MERGE all trash GOSUB stack. (Only a problem if they error out; don't use those keywords with ON ERROR and RETURN.)
5.	1160-1	20	ON TIMER GOSUB within ON TIMER GOSUB reschedules alarm at wrong intervals. (Don't reschedule an ON TIMER within a GOSUB.)
6.	1078-5	18	DATE$ sometimes causes Data Type error when it is concatenated. (Instead of A$&DATE$, use D$=DATE$ @ A$&D$.)
7.	1103-1	16	Creating or copying in a file from a subprogram may corrupt memory. (To avoid this, every program must have an END statement.)
8.	1161-9	16	A=FNB, then redimension A within FNB corrupts the variable, causes cold start sometimes. (Never re-dim a var while it's being assigned.)
9.	1131-2	14	END SUB trashes first variable in calling environment's chain if 1st var < 8 bytes. (Ensure first variable in chain is > 8 bytes.)
10.	1115-5	12	CREATE allows creating a null-named file, which ends the file chain. (Verify a non-null name before CREATE.)
11.	1148-6	12	10 A=1.23456789E-10 decompiles as 10 A=.0000000 (Use 10 A=123456789*1E-18 .)
12.	1084-3	10	Small probability exists for bad timer readings, causing random clock error. (Risk of this bug occurring increases if user

runs 2 or 3 ON TIMERs with short intervals.)

13. 1082-7 8 READ# with complex destination doesn't work right sometimes; corrupts variable.
(Use SDATA file for storing complex numbers.)

14. 1175-9 CALL a subprogram in an EEPROM always gives "ERR:Sub Not Found".
(CALL SUBXX errors, but CALL SUBXX IN FILEXX will work.)

15. 1072-8 4 Outer FOR...NEXT loop is corrupted when inner FOR statement errors at execution.
(Only a problem if you do keyboard GOTO after the FOR statement error.)

16. 1105-6 4 Can't use UDF string parameter to specify a file name in CALL.
(Make intermediate assignment for UDF param.)

17. 1138-7 3 In CALC mode, UDF into program with ON ERROR cold starts.
(Never use UDF in CALC mode if SUSP is lit.)

18. 1178-3 3 RUN<file>:<HPIL device> copies the file, but sometimes won't run, may give spurious error.
(COPY <file>:TAPE first, then RUN <file>.)

19. 1124-7 2 LIST and PLIST should exit through DPART3; presently HPIL does not clean up loop.
(A problem only in rare cases where loop is used heavily; SEND UNT UNL will prevent it.)

20. 1133-8 2 Programmatic RUN<filename>,<line#> cold starts, if <line#> too large.
(Verify line#'s before using RUN in program.)

21. 1140-3 2 RETURN to binary program will not work if TRACE FLOW is in effect.
(Never use TRACE FLOW for BINary programs.)

22. 1142-9 2 TRANSFORM at low memory sometimes leaves an empty file.
(Check for Insufficient Memory error if TRANSFORM errors out, purge file.)

23. 1177-5 2 RUN<file>,<label> leaves file pointers messed up if "ERR:Stmt Not Found", or error in <label> evaluation.
(If this happens, do an END ALL. To guard against worse problems, don't use UDF for

			label; e.g., RUN <file>,FNX$.)
24.	985-2	1	Assigning UDF, which changes OPTION BASE, to nonexistent array, corrupts memory. (Never change OPTION BASE in a UDF.)
25.	1095-9	1	Typing (1,2,3,...,n) [ENDLINE] in CALC mode tanks machine if 7<n<16. (Don't ever enter an n-tuple.)
26.	1120-5	1	With TRAP(IVL,2) , CALL with subscript out-of-range gives infinite loop with warnings. (Restore from infinite loop with INIT 1.)
27.	1169-2	.05	When waking up with password in CALC mode, [g][ERRM] key sequence can cold start. (Requires an obscure sequence of 5 steps, and usually doesn't corrupt memory.)
28.	1137-9	.03	In CALC mode, 2-dimensional variable corrupts input buffer when buffer full. (Even if it happens, tiny chance memory corrupt.)
29.	1154-4	.00	Configuration fails to set NoCont flag if configuration changes. (We haven't found a bug from this yet, but...)

Part 1: HP-71 Bugs by Severity

TRIVIAL (21)

	Number	Rank	Description & (Comments)
1.	1145-2	30	Write to CARD sometimes doesn't work if card pulled too fast. (User would just think it was a bad write.)
2.	1168-4	18	Programmatic BYE leaves display off at wakeup. (Several statements will turn it on; simplest is to execute DISP.)
3.	1166-8	8	Programmatic BYE leaves 3 key rows de-energized at wakeup. (Several statements will re-energize; simplest is to execute KEYDOWN.)
4.	1035-5	7	Modify POKE to prevent write of "00" byte to first byte of file header. (Enhancement to file security.)
5.	1134-6	6	Trace with implied END SUB displays garbage line number. (Minor nuisance; only if a null subprogram.)
6.	1146-0	6	RENUMBER does an END SUB. (Minor nuisance; user re-starts program.)
7.	1106-4	4	LIST should be able to list a BASIC line of more than 95 characters. (Enhancement to LIST.)
8.	1094-2	3	DISP USING does not handle non-existing string array elements properly. (Errors out; minor nuisance.)
9.	1135-3	3	Cursor is turned off by ON ERROR branch. (Minor nuisance; fix enhances display routines.)
10.	1172-6	3	TRACE VARS shows only lower 4 digits of a subscript or substring limit. (Assignment is still done correctly, so array not harmed. When using large subscripts, keep in mind that TRACE VARS truncates subscript.)
11.	1065-2	2	CAT CARD on DATA files <4 bytes long shows file length to be huge. (Disturbing to user, but minor nuisance.)
12.	1126-2	2	At low memory, subprogram string parameter passed by value is null.

			(Doesn't affect memory or program.)
13.	1136-1	2	DEF KEY does not accept undeclared string array element. (Errors out; minor nuisance.)
14.	1144-5	2	ON TIMER...GOSUB behaves funny with BYE and OFF TIMER. (Interesting effects, but no harm.)
15.	1164-3	1	LOGP1(-0) returns 0 instead of -0. (If this is a problem for a user, he can trap out -0 before the LOGP1 function.)
16.	1080-1	.2	Colon key definition execution doesn't do a cursor-far-right before CR/LF. (User won't even notice unless using CRT.)
17.	1171-8	.2	External alarm setting is more complicated than presently documented in the IDS. (Assembly language programmers should consult with Technical Support for proper interface with clock system.)
18.	1100-7	.1	IMAGE syntax does not behave as documented in certain cases. (If user follows manual, he'll never know.)
19.	1086-8	.01	If FORTH ROM absent, configuration does not mark page F as off-limits. (Only a problem if user plugs in 2 64K chips.)
20.	1170-0	.01	CALL a null program, SUSPend quickly with [ATTN] key makes FETCH show a garbage line#. (Amusing feature; no harm.)
21.	1130-4	.00	READ# takes wrong error exit, shows blank error message. (Most obscure bug on the list.)
22.	1150-2	.00	Routine TDPARS in clock system does an ASRB without clearing A(5). (We haven't found a bug from this yet, but...)
23.	268-3	na	SHOW PORT should give information on all plug-in devices. (Enhances SHOW PORT; not programmable anyway.)

Manual Changes (4)

	Number	Comments
1.	268-3	(See entry #23 in "Trivial" category.) SHOW PORT statement now lists all plug-ins.
2.	1110-6	(See entry #1 in "Minor" category.) ON...GOTO bug; remove current addendum card.
3.	1165-0	(See entry #2 in "Minor" category.) READ# has been determined to operate the same as Series 80, so changing it would be undesirable. The Reference Manual conflicts with the way READ# actually works, so the manual will be updated to properly describe the action of sequential READ#.
4.	1171-8	(See entry #17 in "Trivial" category.) The IDS Vol I will be updated to describe the proper interface to the alarm system.

Part 2: HP-71 Bugs by SR Number

BY SR NUMBER:

Number	Severity	Priority	Date	Comment
268-3	trivial	4	4/08/83	Enhancement request; manual will have to be revised. SHOW PORT should give information on all plug-in devices.
985-2	Minor	2	8/03/83	Assigning UDF, which changes OPTION BASE, to nonexistent array, corrupts memory.
1035-5	trivial	4	8/25/83	Enhancement request. Modify POKE to prevent write of "00" to first byte of file header.
1065-2	trivial	4	9/12/83	CAT CARD on DATA files <4 bytes long shows file length to be huge.
1072-8	Minor	4	9/15/83	Outer FOR/NEXT loop is corrupted when inner FOR statement errors at execution.
1078-5	Minor	3	9/16/83	DATE$ sometimes causes Data Type error when it is concatenated.
1080-1	trivial	3	9/20/83	Colon key definition execution doesn't do a cursor-far-right before CR/LF.
1082-7	Minor	3	9/23/83	Reported by QA engineer. READ# with complex destination doesn't work right sometimes, corrupts variable.
1084-3	Minor	3	9/28/83	Reported by customer. Small probability exists for bad timer readings, causing random clock error.
1086-8	trivial	4	10/06/83	If FORTH ROM absent, configuration does not mark page F as off-limits.
1094-2	trivial	4	10/17/83	Reported by Applications Eng. DISP USING does not handle non-existing string array elements properly.
1095-9	Minor	2	10/17/83	Reported by Applications Eng. Typing (1,2,3,...,n) [ENDLINE] in CALC mode tanks machine if 7<n<16.
1100-7	trivial	4	10/21/83	Reported by Applications Eng.

IMAGE syntax does not behave as documented in certain cases.

1103-1 Minor 2 11/02/83
Creating or copying in a file from a subprogram may corrupt memory.

1105-6 Minor 3 11/07/83
Can't use UDF string parameter to specify a file name in CALL.

1106-4 trivial 4 11/11/83
LIST should be able to list a BASIC line of more than 95 characters.

1110-6 Minor 2 12/08/83 Reported by Applications Eng.
..... Currently listed on addendum card.
ON...GOTO performs ON...RESTORE if OPTION ROUND NEG or POS, or if function used for index.

1112-2 MAJOR 1 12/16/83
Concatenating a string constant at low memory may cause cold start.

1113-0 MAJOR 1 12/16/83
Concatenating string variables at low memory may cause cold start.

1115-5 Minor 2 12/29/83
CREATE allows creating a null-named file, which ends the file chain.

1119-7 Minor 3 1/10/84 Reported by customer.
VAL, TRANSFORM and MERGE all trash GOSUB stack.

1120-5 Minor 3 1/10/84 Reported by Applications Eng.
With TRAP(IVL,2) , CALL with subscript out of range gives infinite loop with warnings.

1121-3 Minor 3 1/18/84
POS(<str1>,<str2>,0) always returns 0, contrary to manual.

1122-1 MAJOR 2 1/23/84
Assignment to string array element with subscript out-of-range can tank.

1123-9 MAJOR 3 1/24/84
Assignments to string array element in subprogram corrupts memory.

1124-7 Minor 3 1/27/84
LIST and PLIST should exit through DPART3; presently HPIL does not clean up loop.

1126-2 trivial 3 1/31/84
At low memory, subprogram string parameter passed by value is null.

Part 2: HP-71 Bugs by SR Number

1130-4 trivial 3 2/03/84
READ# takes wrong error exit, shows blank error message.

1131-2 Minor 2 2/08/84 Reported by customer.
END SUB trashes first variable in calling environment's chain if first var is < 8 bytes.

1133-8 Minor 1 2/13/84
Programmatic RUN<filename>,<line#> tanks, if <line#> too large.

1134-6 trivial 3 2/13/84
Trace with implied ENDSUB displays garbage line number.

1135-3 trivial 4 2/14/84
Cursor is turned off by ON ERROR branch.

1136-1 trivial 4 2/14/84
DEF KEY does not accept undeclared string array element.

1137-9 Minor 2 2/17/84
In CALC mode, 2-dimensional variable corrupts input buffer when buffer full.

1138-7 Minor 1 2/17/84
In CALC mode, UDF into program with ON ERROR cold starts.

1140-3 Minor 3 2/28/84
RETURN to binary program will not work if TRACE FLOW is in effect.

1142-9 Minor 3 2/29/84
TRANSFORM at low memory sometimes leaves an empty file.

1144-5 trivial 3 3/15/84
ON TIMER...GOSUB behaves funny with BYE and OFF TIMER.

1145-2 trivial 3 3/29/84 Needed production card reader.
Write to CARD sometimes doesn't work if card pulled too fast.

1146-0 trivial 4 4/06/84
RENUMBER does an END SUB.

1148-6 Minor 3 4/23/84
10 A=1.23456789E-10 decompiles as 10 A=.00000000

1150-2 trivial 3 5/01/84
Routine TDPARS in clock system does an ASRB without clearing A(5).

1151-0 MAJOR 1 5/04/84
SUB statement chain field may look like a comment, tank machine when decompiled.

1154-4 Minor 2 5/17/84
Configuration fails to set NoCont flag if configuration changes.

1155-1 MAJOR 2 7/13/84 Reported by European field.
..... Should be on current addendum card.
Invalid leap year determination for even year in odd decade makes DATE$ unusable.

1160-1 Minor 3 8/20/84 Reported by Applications Eng.
ON TIMER GOSUB within ON TIMER GOSUB reschedules alarm at wrong intervals.

1161-9 Minor 1 10/01/84
A=FNB, then redimension A within FNB corrupts variable, causes cold start.

1162-7 MAJOR 3 10/01/84 Should be on current addendum card.
Editing program clears SUB link before checking if EEPROM; destroys EEPROM.

1164-3 trivial 4 11/28/84 Reported in Bell Labs paper on
..... August 21, 1984.
LOGP1(-0) returns 0 instead of -0.

1165-0 Minor 5 11/28/84 READ# has been determined to act
..... the same for Series 80; no change.
Sequential READ# from DATA files skips over EOF mark.

1166-8 trivial 4 11/30/84 Reported by customer.
Programmatic BYE leaves 3 key rows de-energized at wakeup.

1168-4 trivial 4 11/30/84
Programmatic BYE leaves display off at wakeup.

1169-2 Minor 2 11/30/84
When waking up with password in CALC mode, pressing [g][ERRM] can cause cold start.

1170-0 trivial 4 12/06/84
CALL a null program and SUSPend quickly with [ATTN] key makes FETCH show a garbage line number.

1171-8 trivial 5 12/11/84 Requires IDS change.
External alarm setting is more complicated than presently documented in the IDS.

1172-6 trivial 3 12/27/84
TRACE VARS shows only lower 4 digits of a subscript or substring limit.

1173-4 MAJOR 1 12/28/84

Assignment to substring such as A$[1234567]="xxxx" cold starts if substring limit >= 1048575.

1175-9 Minor 2 01/09/85
CALL a subprogram in an EEPROM always gives "ERR:Sub Not Found".

1177-5 Minor 2 01/21/85
RUN<file>,<label> leaves file pointers messed up if "ERR:Stmt Not Found", or if error in <label> evaluation.

1178-3 Minor 2 01/24/85
RUN<file>:<HPIL device> will copy the file, but sometimes won't run, may give spurious error.

HP-71B Serious Bugs
Version 1BBBB

February 6, 1985

This is a list of known bugs on the HP-71B (version 1BBBB) which can cause 'Memory Lost'. All bugs which are known to corrupt memory are included, even though some have little likelihood of being encountered by the user. Each description includes the STARS SR number, and the date it was reported.

------ SR#985-2 ---------- 08/83/83 -----

Changing the OPTION BASE setting in a User-Defined Function.

Example:

```
10 DESTROY ALL
20 OPTION BASE 0
30 A(10)=FNX
40 DEF FNX
50 OPTION BASE 1
60 FNX=123
70 END DEF
```

Comments:
This doesn't always cause a cold start. But it will at least corrupt memory, which may be a worse problem because it may not be detected.

Workaround:
Never change OPTION BASE setting in a UDF.

------ SR#1095-9 ---------- 10/17/83 -----

Typing (1,2,3,...,n) [ENDLINE] in CALC mode cold starts, if 7<n<16.

Example: in CALC mode,
(1,2,3,4,5,6,7,8) [ENDLINE]

Comments:
Don't even fool around with this one. It cold starts every time.

Workaround:
Never type n-tuples in CALC mode

------ SR#1103-1 ---------- 11/02/83 -----

Creating or Copying a file from a subprogram.

Example:

```
EDIT FILEA
10 SUB XX
20 CREATE DATA JUNK !    or COPY JUNK:TAPE
21 !                     or COPY JUNK:PORT   , etc.
30 END SUB
EDIT FILEB
10 CALL X
```

Comments:

In order for this bug to show up:

1) The CALLing program must be the last program in memory; note above that FILEB is the last file in memory. (Files are located according to their chronological order -- that is, the last file created or copied is always the last one in memory.)
2) The CALLing program doesn't end with an END statement.

Usually this bug will show up by freezing the display with the PRGM annunciator on, and just sit there for minutes on end. Eventually the computer cold starts.

Workaround:

Every program should have an END statement, to be safe. (E.g., in FILEB above, add the line: 20 END .)

------ SR#1112-2 ---------- 12/16/83 -----

Concatenating a string constant at low memory.

Example:

```
DIM A$[MEM-100]     ! To get into a low memory condition
A$="00000000000000000000000000000"
DISP A$&"1111111111111111111111111111111"
    --- This will report "ERR:Insufficient Memory"
```

At this point, anything you do will cause a cold start.

If you are trying to reproduce the bug by this description, you may have to adjust the number of 0's and 1's in the strings above.

Comments:

This is a slippery one. It isn't obvious how much memory is "low memory", in a given situation. Probably 1 in 10,000 string concatenations will hit this bug, maybe less. "Low memory" could be defined as "if MEM is less than the length of the string constant, plus the length of the input line." (The length of the input line must be added in, since it is put in the command stack, which uses some memory.)

Workaround:

In a program, check to insure MEM > LEN(<string constant>) before executing the concatenation. A simple procedure is to make a global check beforehand for sufficient memory for all concatenations.

From the keyboard, make sure MEM > 2.5*LEN(<string constant>), as this will insure enough room for the command stack entry.

------ SR#1113-0 ---------- 12/16/83 -----

Concatenating a string variable at low memory.

Example:

```
DIM A$(10)[MEM/12]   ! Dimension A$ large enough.
A$(7)="abcdefghijklmnopqrstuvwxyz"

Then do A$(7)=A$(7)&A$(7) two or three times until
  LEN(A$(7)) is at least 200.

DIM B$[MEM-100]    ! To get into low memory condition.
                   ! Now MEM should be about 94.
DISP A$(7) ! Will cause cold start.
```

Comments:

See comments for above bug, SR#1112-2.

Workaround:

See workaround for above bug, SR#1112-2.

------ SR#1115-5 ---------- 12/29/83 -----

Creating a null-named file.

Example:
CREATE DATA :MAIN
or CREATE TEXT :PORT(2) , etc.

Comments:
This bug doesn't actually cause a cold start, but an INIT 3 is the only way to recover from it (unless the user is familiar enough with the computer to do some creative POKEs).

Creating a null-named file ends the file chain, PERMANENTLY. Any other files created or copied after it can never be accessed.

Workaround:
Never create a null-named file. In a program which prompts the user for a file name to create, the program must trap out a null name.

------ SR#1122-1 ---------- 01/23/84 -----

Assignment to a string array element with out-of-range subscript.

Example:

```
TRAP(IVL,2)    ! IVL must be set=2 for this bug to occur.
DIM S$(2)
S$(33)=PEEK$("0",MEM+90) ! This will take a long time
                  to execute, since MEM+90 is large.
```

Comments:

This rarely corrupts memory, and even more rarely causes a cold start. It depends on how long the string is, and how much memory is in the machine (the PEEK$ string above, with MEM+90, is as long as you can get).

For a sure way to see the bug, first execute FREEPORT(0) before you do the S$(33) assignment. Then, do the assignment and execute CAT :PORT(0), and you will see that the memory in PORT(0) has been corrupted.

Workaround:

First of all, the user must have at some point executed TRAP(IVL,2) in order to set IVL=2. Without this, there is no possibility of hitting this bug. Otherwise, if IVL is set to 2, make sure the subscript is not out of range before making an assignment.

(The easiest workaround may be to insure IVL is set to 1 before making assignments to string arrays. See the next bug, SR#1123-9, also.)

------ SR#1123-9 ---------- 01/24/83 -----

Assignment to string array element in a subprogram, with an out-of-range subscript.

Example:

```
TRAP(IVL,2) ! This is necessary for bug to occur.
DIM A$(5),B$
10 SUB XX(J$,K$)
20 K$=PEEK("0",500)
30 J$=K$
40 END SUB
```

Now CALL XX(B$,A$(33)) causes a cold start.

Comments:

See comments for SR#1122-1 above.

Workaround:

See workaround for SR#1122-1 above.

------ SR#1133-8 ---------- 02/13/83 -----

Programmatic RUN<filename>,<line#> causes a cold start if the line# is too large.

Example:

```
EDIT FILEA
10 RUN FILEB,50

EDIT FILEB
10 BEEP
```

Now RUN FILEA . Since FILEB only goes up to line 10, the RUN statement in FILEA will cause a cold start.

Comments:

FILEA in the above example must reside in memory before FILEB in order for the bug to occur. (If FILEB resides before FILEA, the bug won't show up.)

Workaround:

Don't use a RUN statement in a program without first verifying the line number.

------ SR#1137-9 -------------------------- 02/17/84 -----

In CALC mode, a 2-dimensional variable corrupts the input buffer when the buffer is full.

Example:

```
DIM X(5,5)  , then toggle into CALC mode.  Press:
X(X(X(X(X(X(X(X(X(X(X(X(X(X(X(X(X(X(X(X(X(X(X(X
```

That's 24 times. At the 24th entry, you will get
WRN:Line Too Long
If you press [g][>] to display the last characters in the line, you will see some garbage in the last 10 or so characters. Now if you press [ENDLINE], there is a tiny chance the machine will cold start; otherwise, you will see "WRN:Operator Expected", or "WRN:Illegal Context", with no harm.

Comments:
The chances of getting a cold start are extremely remote. In fact, the chances of finding this bug are extremely remote.

Workaround:
If you ever get "WRN:Line Too Long" in CALC mode, press the [ATTN] key to clear the display. It's just a safe thing to do anyway.

------ SR#1138-7 ---------- 02/17/84 -----

Executing a User-Defined Function in CALC mode, when ON ERROR is in effect.

Example:

```
10 ON ERROR GOTO 40
20 DEF FNA=0/0  ! This generates an error at execution.
30 STOP
40 DISP "NO! NO! NO! NO!"
50 INPUT X
60 STOP
```

Now single-step once, to put the ON ERROR condition in effect. The SUSP annunciator will come on. Now toggle into CALC mode, and enter FNA [ENDLINE]. This will execute the UDF and, of course, generate the error ("0/0"); but the ON ERROR branch will take place, and the program will try to execute. You'll see the "NO! NO! NO! NO!" string displayed, then the prompt for input on line 50. Enter 55 or whatever, and press [ENDLINE]; this will cold start the machine.

Comments:

Executing a BASIC program in CALC mode shouldn't be possible, and doing so usually causes a cold start. However, the ON ERROR branch may not be destructive, depending on the statements executed.

Workaround:

Before getting into CALC mode, always execute OFF ERROR, or turn off the SUSP annunciator (with STOP, END, END ALL, etc.). Or both.

------ SR#1140-3 ---------- 02/28/84 -----

TRACE FLOW or TRACE VARS with a RETURN to a BINary program.

Example:
Executing a BINary program while TRACE is in effect may cause a cold start, if the BINary program calls a BASIC program which does a RETURN.

Comments:
No one has actually come across this bug yet, since no one has written any BINary programs which call a BASIC program which does a RETURN.

Workaround:
Never have TRACE FLOW or TRACE VARS in effect when running a BINary program.

Note for assembly language programmers: you can help prevent this bug (on version 1BBBB) by clearing S15 before calling the BASIC program, or by clearing the trace flags in RAM. In addition, the user should be cautioned not to set TRACE mode in the BASIC program.

------ SR#1151-0 ---------- 05/05/84 -----

SUB statement with line labels or user-defined functions may decompile into garbage.

Example:

```
10 SUB XX
20 DISP MEM ! ABCDEFGHIJKL
30 ! ABCDEFGHIJKLMNOPQRSTUVWXYZ
40 ! ABCDEFGHIJKLMNOPQRSTUVWXYZ
50 ! ABCDEFGHIJKLMNOPQRSTUVWXYZ
60 LABEL: BEEP
50 END SUB
```

Now press [RUN] (causing the program pointers to be set up). FETCH 10 will decompile ('list') line 10 with all kinds of garbage.

Comments:

This has never caused a cold start for us, but it's conceivable that it might.

This bug needs two conditions to occur:

1) the subprogram has line labels and/or user-defined functions in it (both are linked in the same chain)
2) the SUB statement does not have a "!" comment.

Even if these two conditions are met, the bug will show up only about 1 in 256 times.

Workaround:

If a subprogram has line labels or user-defined functions in it, always include a "!" comment after the SUB statement. (E.g., change line 10 to read: 10 SUB XX !)

------ SR#1161-9 ---------- 10/01/84 -----

Re-dimensioning a variable within a User-Defined Function.

Example:

```
10 DESTROY ALL              or      10 DESTROY ALL
20 SHORT G                          20 INTEGER G(FNX)
30 G=FNX                            30 STOP
40 DEF FNX                          40 DEF FNX
50 INTEGER G(11,22)                 50 DIM G(23,88)
60 FNX=123                          60 FNX=3
70 END DEF                          70 END DEF
```

Running either of these programs will corrupt memory. Executing DESTROY G will then probably cause a cold start.

Comments:

This doesn't always cause a cold start. But it will at least corrupt memory, which may be a worse problem because it may not be detected.

Workaround:

Never re-dimension a variable in a UDF when that variable is pending an assignment or creation. (I.e., in the above programs, G is being assigned/created by the UDF, which in turn re-dimensions it.) Always allocate it beforehand with a DIM statement (or REAL, INTEGER, etc.).

------ SR#1169-2 ---------- 11/24/84 -----

Waking up with a password in CALC mode and pressing [g][ERRM].

Example:

1) First, generate some error so that the [g][ERRM] keystroke will display some non-null string.
2) Then enter:
   LOCK "xxx"
3) Then toggle into CALC mode, and turn off the machine.
4) Now turn on the machine, and it will prompt:
   password?
5) Before entering the password, press [g][ERRM]; this will lock up the machine for a few seconds. It will then report "ERR:Insufficient Memory", and turn off.
6) Now turn on the machine again, and enter the correct password. The computer will get into the command stack; if you press [ENDLINE], you will see "ERR:Insufficient Memory" -- or worse, 'Memory Lost'.

Comments:

The chances of getting a cold start are pretty slim with the above sequence. But when a STARTUP string is in effect, the results are more predictable. For instance, enter

```
STARTUP "INPUT X"
```

before you do the above steps. After step 6, the computer will prompt for input; enter 55 or whatever. Now try to toggle out of CALC mode -- the computer is lost.

Workaround:

Never do the above sequence of steps.

------ SR#1173-4 ---------- 12/28/84 -----

Assignment to substring with a very large substring limit.

Examples:

```
A$[1234567]="xxx"
F$[1E22]=G$
U$(4)[NaN]=TIME$
P$[Inf]="zzzz"
```

Comments:

The above statements will cause a cold start (or, rarely, corrupt memory without losing it) whether typed in from the keyboard or executed in a program.

The conditions which cause a cold start are these:

-- the string value being assigned is at least two characters long, and
-- the substring limit (s) PLUS the length (l) of the assigned string must be:
   -- in the range 524289 <= s+l <= 524321
   -- or, s+l >= 1048577 .

(Note that 524287 = 7FFFF hex, and 1048575 = FFFFF hex; this may give an idea why these strange ranges came about.)

Note that NaN and Inf are represented internally as greater than FFFFF hex, so that using either as a substring limit in an assignment will cause a cold start. Using an expression as a substring limit requires caution, since the expression may evaluate to NaN or Inf. For example,

-- F$(N(14))="XXX" may cause a cold start if the array N is dimensioned less than 14 -- i.e., a subscript out of range returns NaN (if TRAP IVL is set to 2).

-- F$(A/B)="ZZZ" will cause a cold start if B=0, since A/0 will return Inf (or MAXREAL, depending on the DEFAULT setting).

These two cases are probably the most serious with this bug, since rarely will anyone specify a substring with a large value.

Workaround:

Since a string can be at most 65535 bytes, don't use any values larger than that for substring limits.
When using an expression for a substring limit in a string assignment, screen out large values, NaN and Inf.

------ SR#1177-5 ---------- 01/21/85 -----

Using a RUN statement in a program, and specifying the starting label with a UDF.

Example:

```
EDIT TEST
10 BEEP
20 RUN OTHER,FNL$

EDIT XXX
10 DISP "WWWWWWWW"

EDIT OTHER
33 !
44 DEF FNL$
55 PURGE TEST    ! or TRANSFORM TEST INTO TEXT
66 FNL$="LBL1"
77 END DEF
```

Notice that file TEST specifies the starting label in the RUN statement using a UDF. The UDF is found in the file which is to be run. But FNL$ evaluates to the string 'LBL1', and there is no such label in file OTHER. This causes an error; but the error reported may be unrelated to the label problem:

```
ERR~~Data Type
```

Since the original file has been altered (purged in this case), the computer cannot recover the location to resume executing. (In fact, in the above example the file XXX has been moved to where TEST used to be, so that resuming execution will pick up somewhere inside file XXX -- and the chances of finding a valid BASIC statement at the same location are pretty slim.)

Now single-stepping (or CONTinuing) will cause problems; some wild line numbers might appear, and further single-steps will cause a 'Memory Lost'.

Comments:

These problems will only happen when the starting line in a RUN statement is specified with a UDF, and when the original file is altered (PURGE or TRANSFORM) within the UDF. The fact that the UDF is found in the NEW file would indicate that using a UDF for the label is not very practical.

Workaround:

Don't use a UDF to specify the starting line in a RUN statement. (A UDF to specify the file name is OK, such as RUN FNL$,LBL1 .) If the user gets into such a situation, and an error occurs, execute END ALL before trying to resume execution.

------ SR#1178-3 ---------- 01/24/85 -----

Running a file from an HPIL device.

Example:
RUN NEWFILE:TAPE

If memory is set up just right (plug-in modules and files in all the 'right' places), there is a very small chance that this will cause a 'Memory Lost'. Any HPIL device may be specified, not just :TAPE .

Comments:
More than 99% of the time, this statement will run correctly. Occassionally (maybe .1% of the time), the file will be copied in properly, but the computer will stop after copying and not run the file. (A subsequent RUN command will work OK.) It is feasible that it may cause a 'Memory Lost' (maybe .0001%), but the required half-dozen conditions are extremely rare.

No one has reported a 'Memory Lost' from this, but a "COPY but no RUN" has been noticed once.

Workaround:
Instead of RUN <file>:<device>, first copy the file in with the command COPY <file>:<device>, then RUN <file>. However, the chances of 'Memory Lost' are so remote that this workaround may be deemed unnecessary.