HP 71

Bugs (1986)

SR REPORT FOR: TEST-TITAN,TEST [illegible] AS OF 02/02/86 14:22 PAGE 10

SR # 0039011791 TITAN [illegible] HP-44 [illegible] 03.28 Sev: N Pri: 2

Classification: Known problem [illegible] Status: Open known problem

When current file is BINary, UDF in CALC mode cold starts.

I M188BB

Company : HP LAB/039
Submitter: MARK BANWARTH
User # :
Verifier : MARK BANWARTH
SO ref # :
Proj mgr : ERIC EVETT
Mkt engr : BUGCZAR
Lab engr : MARK BANWARTH
QA engr :
Func area: MAINFRAME DESIGN
Prod line: SATURN
Sub line : SATURN
Sys model:
Release : TI%R6B
Prod type: UNRL

Submitted : 02/08/85
Entered : 02/08/85
SR updated : 02/08/85
Text updated: 02/08/85
Reclassified: / /
Purge date : / /
To lab : 02/08/85
From lab : 02/08/85
AR date : / /
To QA : / /
From QA : / /
New v.u.f : . .
Fix number :
Prob cause :

SUBMITTER TEXT:

REPEATABLE? : Yes.
OPER. MODE : CALC mode.
DESCRIPTION : When current file is BINary, UDF in CALC mode cold starts.
Example: Find a BINary file somewhere (use the Curve Fit ROM, for instance, and RUN FITLIB to make it the current file).
With the BINary file as the current file, toggle into CALC mode and enter FNX [ENDLINE]. The message "ERR:Insufficient Memory" is displayed and CALC mode enters the command stack. Press [ENDLINE] and the memory error is displayed again; as long as you are in the command stack, [ENDLINE] will produce the Insufficient Memory error. This will cause a cold start if you clear out the command stack line with [f][C-line], the press [ENDLINE]. The computer will sit there for a few minutes as it decompiles the entire machine; press [ENDLINE] after it comes back, and you get Memory Lost.

This same situation occurs if the current file is BASIC, but in a non-RAM device and not yet chained. For example, with a RAM box, [illegible] PORT(1) (or whatever port the RAM box is plugged into). Then EDIT BUG:PORT(1) and enter a couple of lines. Then turn the machine off (don't RUN or take any action which will chain the file). Now change the switches on the RAM box from 8000A to 8010A (from RAM to ROM). Turn on the machine and EDIT BUG:PORT(1). (If you RUN now, you will get the error ERR:Illegal Access, because the program cannot be chained unless it is in RAM; the same thing will happen if you enter FNX [ENDLINE] in BASIC). Now toggle into CALC mode and enter FNX [ENDLINE]. ERR:Insufficient Memory will be reported, and the same situation describe above for BINary files will occur.

The problem is that the CALC mode parser works with memory inverted, and when the UDF is searched, program scope is checked first. The [illegible] routine can take 2 hardwired error exits: (1) if the file is non-BASIC, it exits to ERR:Illegal File Type, and (2) if the program

is not chained and in a non-RAM device, it exits to ERR:Illegal Access.
Now when the error driver is called, there are only 4 nibs in available
memory (CALC mode inversion), so the message driver aborts the message
and reports Insufficient Memory, trying to recover available memory
in the process. This leaves the CALC mode input buffer scrambled.

****

SR REPORT FOR: TEST.TITAN,TEST AS OF 02/02/86 14:22 PAGE 12

SR # 0039011809 TITAN HP-44 03.28 Sev: N Pri: 2

Classification: Known problem Status: Open known problem

MERGE at low memory: with ON ERROR in effect, or current file in IRAM.

I M18888

Company : HP
Submitter: MARK BANWARTH LAB/039
User # :
Verifier : MARK BANWARTH
SO ref # :
Proj mgr : ERIC EVETT
Mkt engr : BUGCZAR
Lab engr : MARK BANWARTH
QA engr :
Func area: MAINFRAME DESIGN
Prod line: SATURN
Sub line : SATURN
Sys model:
Release : TIXRM6
Prod type: UNRL

Submitted : 02/27/85
Entered : 02/27/85
SR updated : 02/28/85
Text updated: 02/28/85
Reclassified: 02/28/85
Purge date : / /
To lab : 02/28/85
From lab : 02/28/85
AR date : / /
To QA : / /
From QA : / /
New v.u.f : . .
Fix number :
Prob cause :

SUBMITTER TEXT:

REPEATABLE? : Yes.
OPER. MODE : Running, Keyboard.
DESCRIPTION : MERGE at low memory can cause a cold start: with ON ERROR in effect, or with current file in IRAM. Two aspects of this bug:
1) With ON ERROR in effect, a programmatic MERGE can cause a cold start. For example,

```
EDIT AA
10 ON ERROR GOTO 40
20 MERGE QQ
30 STOP
40 BEEP
```

Now CREATE DATA XX,MEM-80,1 to fill up most of memory.
Now EDIT QQ

```
200 ! A long, long, long, long comment. (as long as memory allows)
```

Now MEM should be 35 or so.
Now RUN AA, and line 20 will generate an Insufficient Memory error, execute the ON ERROR GOTO, and beep. Now replace line 20:

```
20 X=1/2
```

which will generate another error at runtime.
Now RUN AA again, and the machine will cold start.
The problem: MERGE calls PEDIT, which clears the SUB and label links; but the Memerr does the ON ERROR GOTO, which compiles the GOTO! Now we have a file with zeroed links, but a compiled GOTO (or several, if the ON ERROR routine does several GOTOs). Entering the new line 20 would normally clear all compiled GOTOs, BUT -- since the SUB and label links are cleared, the code 'assumes' the GOTOs are cleared, too, which isn't true in this case. Now running the program again fills in the links, of course, but the GOTOs are not compiled until actually executed, and then only if nonzero. But the ON ERROR GOTO is nonzero, and now points into unknown tokens!

2) MERGE at low memory into a file in a non-MAIN device tanks. For example, (clear memory from above example, if necessary)

```
     FREEPORT(0)
     EDIT QQ (in MAIN)
     100aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
(Line 100 is lots of implied DISP statements, more than 250 bytes long.)
Now CREATE DATA X,MEM-500,1
and DIM A$[MEM-20]     to get into low memory (MEM should be about 30).
     EDIT AA:PORT(0)
     MERGE QQ       this will corrupt variable A$ .
Now PURGE X         to get back some memory.
     LEN(A$)        will show some unexpected value like 20350 !
```

The problem is that MERGE uses the output buffer (OUTBS) to store each merged line temporarily (while it calls PEDIT). But it calls MOVEU2 to transfer the line into the output buffer, without checking for sufficient memory! When memory is low and the line is long, the line will overwrite Available Memory End. If there are stacks there (such as a CALL stack), it will be corrupted. Otherwise, the variable storage is corrupted.

FIX INFORMATION TEXT:

MERGE now computes the length of the longest line, and makes sure there is sufficient memory for an output buffer of this length. If the current file is in main, it also adds this length to the length of the merge block, to allow for reduced memory for the output buffer.

****

SR REPORT FOR: TEST_TITAN,TEST AS OF 02/02/86 14:22 PAGE 14

SR # 0039011817 TITAN HP-44 03.28 Sev: N Pri: 1

Classification: Known problem Status: Open known problem

MERGE must be followed by RENUMBER! Lines with GOTO/GOSUB not zeroed.

I M1BBBB

Company : HP
Submitter: MARK BANWARTH LAB/039
User # :
Verifier : MARK BANWARTH
SO ref # :
Proj mgr : ERIC EVETT
Mkt engr : BUGCZAR
Lab engr : MARK BANWARTH
QA engr :
Func area: MAINFRAME DESIGN
Prod line: SATURN
Sub line : SATURN
Sys model:
Release : TI%IM6
Prod type: UNRL

Submitted : 02/28/85
Entered : 02/28/85
SR updated : 02/28/85
Text updated: 02/28/85
Reclassified: / /
Purge date : / /
To lab : 02/28/85
From lab : 02/28/85
AR date : / /
To QA : / /
From QA : / /
New v.u.f : . .
Fix number :
Prob cause :

SUBMITTER TEXT:

REPEATABLE? : Yes.
OPER. MODE : Keyboard.
DESCRIPTION : MERGE of a line with GOTO or GOSUB can cause cold start.
For example:

```
     EDIT AA
     10 DISP "10..."
     20 GOTO 10
```

Now RUN to chain the file and compile the GOTO; interrupt with [ATTN].
Now EDIT BB

```
     15 ! XXXXXXXXXXXXXXX
```

Now MERGE AA. List will show (file BB):

```
     10 DISP "10..."
     15 ! XXXXXXXXXXXXXXX
     20 GOTO 10
```

Now RUN BB, and the machine will lock up, eventually cold start.
The problem is that MERGE clears links and compiled GOTOs only when the first line is merged (merge calls EDIT, then CLLINK). On subsequent merged lines, CLLINK finds the links zeroed, so any compiled GOTOs are not cleared! MERGE 30 copies a line with a compiled GOTO into the file BB, and running BB takes this offset and jumps -- into oblivion. This bug will not show up if the merged block is undisturbed in the current file; that is, if all relative offsets are still valid, with no interspersed or missing lines.

FIX INFORMATION TEXT:

MERGE now calls RENSUB after the merge is completed. This zeroes all line references.

SR REPORT FOR: TEST_TITAN,TEST AS OF 02/02/86 14:22 PAGE 15

SR # 0039011825 TITAN HP-44 03:28 Sev: N Pri: 3

Classification: Known problem Status: Open known problem

MERGE & PURGE don't clear GOSUB stack; causes "ERR:System Error".

I M1BBBB

Company : HP
Submitter: MARK BANWARTH LAB/039
User # :
Verifier : MARK BANWARTH
SO ref # :
Proj mgr : ERIC EVETT
Mkt engr : BUGCZAR
Lab engr : MARK BANWARTH
QA engr :
Func area: MAINFRAME DESIGN
Prod line: SATURN
Sub line : SATURN
Sys model:
Release : TI%RM6
Prod type: UNRL

Submitted : 03/03/85
Entered : 03/03/85
SR updated : 03/03/85
Text updated: 03/03/85
Reclassified: / /
Purge date : / /
To lab : 03/03/85
From lab : 03/03/85
AR date : / /
To QA : / /
From QA : / /
New v.u.f : . .
Fix number :
Prob cause :

SUBMITTER TEXT:

REPEATABLE? : Yes.
OPER. MODE : Keyboard.
DESCRIPTION : MERGE and PURGE should clear program stacks. Doing either within a GOSUB can generate an "ERR:System Error".
For example,

```
EDIT X
10 !
EDIT Y
10 GOSUB 20
20 WAIT 2
30 RETURN
```

Now RUN Y, interrupt with [ATTN] during the WAIT. Enter MERGE X; this will turn off the PRGM annunciator. But type RETURN from the keyboard, and the error "ERR:System Error" results. The problem is that MERGE calls ZERPGM instead of CLRSTK (CLRSTK clears the FORSTK, GSBSTK and MTHSTK before falling into ZERPGM).
Similar to the above example, enter PURGE when the program is suspended, then RETURN; "ERR:System Error" is reported. A System Error is a most frightening thing to a user, for it usually portends a Memory Lost. However, in these cases nothing bad has actually happened to memory: the RETURN simply found an address which is no longer valid in memory, since the GSBSTK was left intact during the MERGE (or PURGE). There is no good reason why this should happen -- both the statements should clear the GSBSTK; there might be a use for keeping the FORSTK around, but since this might only be handy for keyboard MERGE/PURGE, there is no great loss from clearing these, too.

A more sinister problem:

```
EDIT A
1 ! PPPPPPPPPPPPPPPPPPPP
EDIT B
1 ... CALL 7 ...
```

```
    20 END
    30 SUB Z
    40 WAIT 2
    50 END SUB
```

Now RUN A, and interrupt with [ATTN] during the WAIT. Now MERGE A; the PRGM annunciator will turn off. Now [SST] a few times:

```
1st [SST]:  10 ! PPPPPPPPPPPPP      (program starts)
2nd [SST]:  20 END                  (program ends)
3rd [SST]:  <nothing>               (machine locks up!!!)
```

The problem is that MERGE hasn't cleared the CALSTK, so subprogram information is kept. When the END is reached, it performs an END SUB, and recovers the CNTADR (CONTinue address) from the CALL stack. But the CONTinue address points into the new merged line, and single-stepping there tries to decompile garbage. Eventually, if the bogus line is long enough, the machine will cold start. So along with clearing the GSBSTK and FORSTK, MERGE should also clear the CALSTK -- that is, MERGE should call CLPSTK instead of ZERPGM. (PURGE probably should not clear CALSTK, since the worst that can happen is a "ERR:File Not Found" error when the END is executed. Purging a current file while CALLed from another file may be useful.)

FIX INFORMATION TEXT:

MERGE now calls CLPSTK (insteac of ZERPGM), and PURGE now calls CLRSTK (instead of ZERPGM).

****

SR # 0039011833 TITAN HP-44 03.28 Sev: N Pri: 4

Classification: Known problem Status: Open known problem

CHAIN doesn't check exceptions: infinite CHAIN loop, no [SST] decompile.

I M18888

Company : HP LAB/039
Submitter: MAINFRAME DESIGN
User # :
Verifier : MAINFRAME DESIGN
SO ref # :
Proj mgr : ERIC EVETT
Mkt engr : BUGCZAR
Lab engr : MARK BANWARTH
QA engr :
Func area: MAINFRAME DESIGN
Prod line: SATURN
Sub line : SATURN
Sys model:
Release : TI%PM6
Prod type: UNRL

Submitted : 03/03/85
Entered : 03/03/85
SR updated : 03/03/85
Text updated: 03/03/85
Reclassified: / /
Purge date : / /
To lab : 03/03/85
From lab : 03/03/85
AR date : / /
To QA : / /
From QA : / /
New v.u.f : . .
Fix number :
Prob cause :

SUBMITTER TEXT:

REPEATABLE? : Yes.
OPER. MODE : Running.
DESCRIPTION : %I CHAIN XX causes infinite BASIC loop.
For example, FREEPORT(1), and EDIT XX:PORT. Enter 10 CHAIN XX
Then COPY XX into MAIN.
Now RUN XX , and the machine can only be interrupted by an INIT 1,
which has a very good chance of corrupting memory since files are
being purged, copied and moved continuously.
Also, CHAIN does not single-step. Enter
EDIT XX
10 "in XX..."
20 CHAIN YY
EDIT YY
10 "in YYY..."
20 "ending YYY..."
Now EDIT XX, and [SST]:
1st [SST]: 10 "in YY..." (and the string is displayed)
2nd [SST]: (no line is decompiled, but "in YY..." is displayed)
3rd [SST]: 20 "ending YY..." (and the string is displayed)

Both problems happen because CHAIN does not check exceptions.

FIX INFORMATION TEXT:

CHAIN now runs through RUNPT1 (instead of BSCEX2), which checks
exceptions.

SR # 0039011841 HPIL HP-44P 00.00 Sev: N Pri: 2

Classification: Known problem Status: Open known problem

PACK on 9114 corrupts disk.

Company : HP
Submitter: NATHAN ZELLE LAB/039
User # :
Verifier : NATHAN ZELLE
SO ref # :
Proj mgr : ERIC EVETT
Mkt engr : BUGCZAR
Lab engr : NATHAN ZELLE
QA engr :
Func area: HPIL DESIGN
Prod line: SATURN
Sub line : SATURN
Sys model:
Release : TI%HP6
Prod type: UNRL

Submitted : 04/11/85
Entered : 04/11/85
SR updated : 02/02/86
Text updated: 04/11/85
Reclassified: / /
Purge date : / /
To lab : 02/02/86
From lab : 02/02/86
AR date : / /
To QA : / /
From QA : / /
New v.u.f : . .
Fix number :
Prob cause :

SUBMITTER TEXT:

REPEATABLE? : Y
OPER. MODE : Any
DESCRIPTION : PACK on an HP9114 disc drive can corrupt any or all files on the disc.

To demonstrate this, do the following:

```
>INITIALIZE :TAPE,8  ! Assume the 9114 is the only mass storage device
>CREATE TEXT FIRSTFILE:TAPE,500
>CREATE TEXT SECONDFILE:TAPE,500
>CREATE TEXT THIRDFILE:TAPE,500
>PURGE FIRSTFILE:TAPE @ PURGE SECONDFILE:TAPE
>CREATE TEXT FOURTHFILE:TAPE,1000
! Now the tape directory looks like this:
!   Name        Type     Size in Sectors     Starting Sector
! FOURTHFILE   TEXT          4                    3
! SECONDFILE PURGED          2                    5
! THIRDFILE    TEXT          2                    7    <<Not valid, PURGED
>PACK  :TAPE  ! This is the killer.  The directory after the PACK:
! FOURTHFILE   TEXT          4                    3 ! Still OK here
! SECONDFILE PURGED          2                    7 ! THIS SHOULD BE GONE
! THIRDFILE    TEXT          2                    9 ! Contains garbage
```

LAB TEXT:

The problem:
This is not a bug in the HP-71, but rather a bug in the first release of the 9114 ROM. It has been fixed in current release drives.

The bug is that the 9114 does not properly emulate the HP82161 correctly for the "CLOSE RECORD" DSK command. If the last mode was not "write to buffer " the disc does NOT write the data to the medium when the close record is received, while the HP82161 does. The HP-71 (and the 75!) depend on close record always writing the data to the medium to pack it, so this bug causes destruction of data.

The effect of this bug is that the PACKDIR command on the HP-71 becomes a time-consuming loop. PACK calls PACKDIR initially, then assumes that there are no PURGED files left in the directory. As this assumption is not correct in the (buggy version of the) 9114, data is lost as files may be moved *farther out* on the medium (see the example above).

****

SR # 0039011858 TITAN HP-44 03.28 Sev: N Pri: 3

Classification: Known problem Status: Open,known problem

PRINT#, with string item, to external file can lose remaining data.

I M1AAAA I M1BEBB

Company : HP LAB/039
Submitter: MARK BANWARTH
User # :
Verifier : MARK BANWARTH
SO ref # :
Proj mgr : ERIC EVETT
Mkt engr : BUGCZAR
Lab engr : NATHAN ZELLE
QA engr :
Func area: MAINFRAME DESIGN
Prod line: SATURN
Sub line : SATURN
Sys model:
Release : TIXRM6
Prod type: UNRL

Submitted : 02/02/86
Entered : 02/02/86
SR updated : 02/02/86
Text updated: 02/02/86
Reclassified: / /
Purge date : / /
To lab : 02/02/86
From lab : 02/02/86
AR date : / /
To QA : / /
From QA : / /
New v.u.f : . .
Fix number :
Prob cause :

SUBMITTER TEXT:

REPEATABLE? : Yes.
OPER. MODE : Running, keyboard.
DESCRIPTION : PRINT# with string item to external file can lose data.

If PRINT# has a string item followed by another item, and the byte which follows the string item in the file is the last byte in the external device's physical record, then the first byte of the following item will be replaced with an End-of-Record mark (hex FF). All data following the string item in the PRINT# statement will therefore be inaccessible. For example,

```
10 CREATE DATA TEST:TAPE,2,248
20 ASSIGN #1 to TEST:TAPE
30 PRINT #1,1; "ABCD",6          ! second item could also be string
40 READ#1,1+ A$,B @ DISP A$,B
50 END
```

The READ# statement on line 40 causes "ERR L40:Record Ovfl".

Workarounds:

1) Break the PRINT# statement into several statements such that no item follows a string item. To fix the example above, change line 30 to read: 30 PRINT#1,1; "ABCD" @ PRINT #1; 6

2) Choose a logical record length which divides evely into 256. These lenghts are 4,8,16,32,64,128 and 256. However, this method won't help for records > 256, and may waste a large amount of spac space on the medium.

3) Use only numeric data items, or only one string item which is the last item in the PRINT# list.

4) Create and PRINT to the file in BAS, then copy it to the external device.

SR #:0039011866 TITAN HP-44 03.28 Sev: N Pri: 3

Classification: Known problem Status: Open known problem

Funny function which creates variables in a UDF corrupts variable chain.

I M1AAAA I M1BBBB

Company : HP
Submitter: PAUL SWADENER LAB/039
User # :
Verifier : PAUL SWADENER
SO ref # :
Proj mgr : ERIC EVETT
Mkt engr : BUGCZAR
Lab engr : MARK BANWARTH
QA engr :
Func area: MAINFRAME DESIGN
Prod line: SATURN
Sub line : SATURN
Sys model:
Release : TI%RM6
Prod type: UNRL

Submitted : 02/02/86
Entered : 02/02/86
SR updated : 02/02/86
Text updated: 02/02/86
Reclassified: / /
Purge date : / /
To lab : 02/02/86
From lab : 02/02/86
AR date : / /
To QA : / /
From QA : / /
New v.u.f : . .
Fix number :
Prob cause :

SUBMITTER TEXT:

REPEATABLE? : Yes.
OPER. MODE : Running, keyboard.
DESCRIPTION : Creation of variable during funny function execution can corrupt memory.

Example using FNROOT in MATH pac:

```
10 DESTROY ALL
20 DIM F(9), A$(9)
30 J=6 @ V=0 @ DISP J;V                    ! Compare the value of V ...
40 F(6)=FNROOT(10,12,FNY)
50 DISP J;V @ STOP                         !    ... with this value.
60 DEF FNY @ BEEP 500,.2                   ! Just to count the FNY calls.
70 V=V+1 @ F(6)=FVAR @ Z6=1/V-.9           ! Stuff to make FNROOT work.
80 DIM Z$[34]                              ! This is the killer!
90 FNY=Z6 @ END DEF
```

Running the program will call FNY about 12 time; you can count the beeps. Line 40 displays: 6 0
but line 50 displays: 6 3.000001e01

Replace DIM Z$[34] with DIM Z$[38], and line 50 shows: 6 1.
Replace DIM Z$[34] with DIM Z$[70], and line 50 shows: 6 14

Creating a variable within a UDF called by a funny function causes the address of the pending assignment (F(6) in the example above) to become invalid. The assignment is made, and, in this case, it happens to be when V resides. Which variables get corrupted, or where, depends on the length of the new variable -- that's why the length of Z$ affects the new "value" of V.

Workaround:
Dimension all the variables before you execute a funny function which calls a UDF.

For example, instead of creating the variables Z6 and Z$ within the UDF, create it on line 20:

```
  20 DIM F(9), A$(9), Z6, Z$[34]
```

Now the program will run correctly, with no corruption of variables.

****