#SHAREorg

SHARE
Technology · Connections · Results

z390, zCOBOL, and zCICS: What It Is, What's New, and What's Next

M. Ray Mullins
Cat Herder Software, LLC

Tuesday, 5 February 2013 1100-1200
Session 12252





SHARE in San Francisco 2013

Trademark Acknowledgements

- IBM Corporation
  - z/OS, z/VSE, z/VM, CICS, VSAM, DB2
- Microsoft Corporation
  - Windows
  - Visual C++
  - Visual Studio
- Oracle
  - Java
- Apple Computer
  - Mac OS X

Trademark Acknowledgements

- The FreeBSD Foundation
  - FreeBSD
- Fujitsu Technology
  - BS2000/OSD
- Linus Torvalds
  - Linux
- The Open Group
  - UNIX

Agenda

- z390 Portable Mainframe Assembler
  - Assemble, link, execute HLASM-compatible programs
- zCOBOL Portable Mainframe Compiler
  - Compile, link, execute COBOL programs
- zCICS
  - Support EXEC CICS COBOL and Assembler
  - Run local and remote TN3270 CICS transactions
- zPAR
  - Problem Analysis Reports

Agenda

- Recent administrative and developer changes
- Changes in support
- What's next
- Questions and Answers (but feel free to ask during the presentation!)

The z390 and zCOBOL Projects

z390 and zCOBOL – What are they?

- A Java-based tool to develop and test z/Architecture Assembler and IBM COBOL programs
- A training tool for those wishing to learn those languages
- CICS and VSAM are supported
- Prototype application system design and architecture
- “Rightsize” existing applications with minimal effort
- Runs in any environment where a JRE is supported– Windows, Mac OS X, Linux (and possibly more–we can test!)

And it’s free!!

The z390 Portable Mainframe Assembler

z390 Portable Mainframe Assembler

- z390 Open Source Java Project
  - Preprocess HLASM-compatible macro code
  - Assemble HLASM-compatible programs
  - Link object code into z390 load modules
  - Execute load modules on Java run-time platforms:
    - Windows (XP, Vista, 7, 8), Mac OS X, and Linux
    - 24/31-bit AMODE/RMODE
    - 32/64-bit GPR, HFP/BFP/DFP
    - QSAM and VSAM emulation
    - zCICS with SOA, TN3270

z390 Portable Mainframe Assembler

- Compatibility options
  - Some z/OS and z/VSE macros and assembler services
  - Link object code into z390 load modules
  - Creates basic object decks compatible with z/OS and z/VSE
  - BS2000/OSD assembler compatibility (system variables)

z390 Macro Processor

- Expands macro code into BAL source code
- Extensions to AREAD and PUNCH for file I/O
- Library with MVS-compatible common macros based on OS/VS2 MVS 3.8j
- Structured programming macros based on Concept 14 (very similar to the HLASM Toolkit SPMs)
- Built-in parser for EXEC CICS and EXEC SQL (zCICS)
- SOA client server application generator (zCICS)
- Macro processor and assembler parallel tasks
- Options for macro execution trace and statistics

z390 Assembler and Linker

- Assemble BAL source code to object code
- Support assembly of all z/Architecture opcodes and HLASM extended mnemonics
- Link multiple object files into load module
- Options for command input and auto-link
- Options to set AMODE and RMODE
- Options for listing, trace, and statistics
- Constantly working to improve HLASM compatibility– submit tickets for bugs found

z390 Emulator

- Execute z390 modules on any JRE-supported platform
- Execute all problem state instructions (except zEC12)
- z390 macro generated SVC support includes: GETMAIN, FREEMAIN, STORAGE, LINK, LOAD, DELETE, WTO, WTOR, TGET, TPUT, TIME, ESPIE, QSAM/BSAM DCB OPEN, CLOSE, GET, PUT, READ, WRITE, CHECK, VSAM ACB RPL OPEN, CLOSE, GET, PUT, POINT, GENCB, MODCB, TESTCB *(some only partially)*
- Options for interactive TEST and TRACE

z390 Emulator Additional Support

- CMDPROC macro and svc for multi-tasking
- CTD and CFD macros to convert HFP, BFP, and DFP floating point to decimal scientific form
- TN3270 screen I/O using TGET, TPUT
- TCPIO macro and svc to support TCP/IP sockets
- SOA application generator with demo
- 2 GHZ Intel dual processor executes 2.5+ MIPS (in 2007, on Windows XP)

z390 Hello World Demo

- To install and run z390 demo on Windows
  - Download and install a Java run-time environment (minimum release 6)
  - Download and install z390
  - Double click on z390 icon to start GUI
  - Enter the command: ASMLG DEMO\DEMO
  - Watch start, “Hello World”, and stop
  - Enter: notepad demo\demo.log to view log

Hello World Demo source file (MLC)

```
DEMO     SUBENTRY  (provided with z390)
         WTO 'HELLO WORLD'
         SUBEXIT
         END
```

Hello World Demo source BAL

```
DEMO      CSECT
……
          BRAS   1,*+(WTO#2_EOT-*+1)/2*2
          DC     AL2(WTO#2_EOT-*,0),C'HELLO WORLD'
WTO#2_EOT EQU     *
          SVC    35
……
          END
```

Hello World Assembler Listing (PRN)

```
……
00005A A715000A                  27          BRAS  1,…
00005E 000F0000C8C5D3D3          28          DC    C’H…
00006D 00006D                    29 WTO#2_EOT EQU *
00006E 0A23                      30          SVC 35
……
END
```

Hello World hex object file (OBJ)

```
.ESD ESD=0001 LOC=00000000 LEN=00000088 TYPE=CST NAME=DEMO
……
.TXT ESD=0001 LOC=00000050 LEN=10 50F0D00850D0F00418DFA715000A000F
.TXT ESD=0001 LOC=00000060 LEN=0D 0000C8C5D3D3D640E6D6D9D3C4
.TXT ESD=0001 LOC=0000006E LEN=10 0A2341F0000058D0D004980CD01458E0
……
```

- This format is az390 option OBJHEX, an enhancement to allow for easy debugging of generated object decks. It is fully supported by lz390. By default, az390 generates a standard OBJ deck.

Hello World Linker Listing (LST)

```
LZ390I V1.3.00a Current Date 01/16/07 Time 19:16:00
LZ390I program = demo\demo.OBJ
LZ390I options = bal objhex nolistcall
LZ390I ESD=DEMO LOC=00000000 LEN=00000088
```

Hello World Execution Log

```
EZ390I V1.3.00a Current Date 01/16/07 Time 19:16:00
EZ390I program = demo
EZ390I options = bal objhex nolistcall
HELLO WORLD
EZ390I Stats total instructions = 13
EZ390I Stats current date 01/16/07 time 19:16:00
```

Hello World Execution Trace file (TRE)

```
……
800FFFD2 0 A715000A BRAS R1=00002300 S2(000FFFE6)=0A23 SVC
800FFFE6 0 0A23 SVC I1=23 WTO R1=ADDR(AL2(LEN),AL2(FLAGS),C'MSG')
EZ390I HELLO WORLD
800FFFE8 0 41F00000 LA RF=00000000 S2(00000000)
……
```

Hello World Interactive TEST log

```
20:33:44 demo      EZ390 START USING z390 V1.5.06 ON J2SE 1.7.0_11 01/22/13
test enter command or h for help
20:33:44 demo      EZ390 START USING z390 V1.5.06 ON J2SE 1.7.0_11 01/22/13
g
test cmd: g svc
test break on g svc
800FFFE6 0 0A23 SVC I1=23 WTO R1=ADDR(AL2(LEN),AL2(FLAGS),C'MSG')
g
test cmd: g
HELLO WORLD
20:34:16 demo      EZ390 ENDED   RC= 0 SEC=31 MEM(MB)= 14 IO=23 INS=13
```

z390 Portable Mainframe Assembler What's new since SHARE Summer 2011

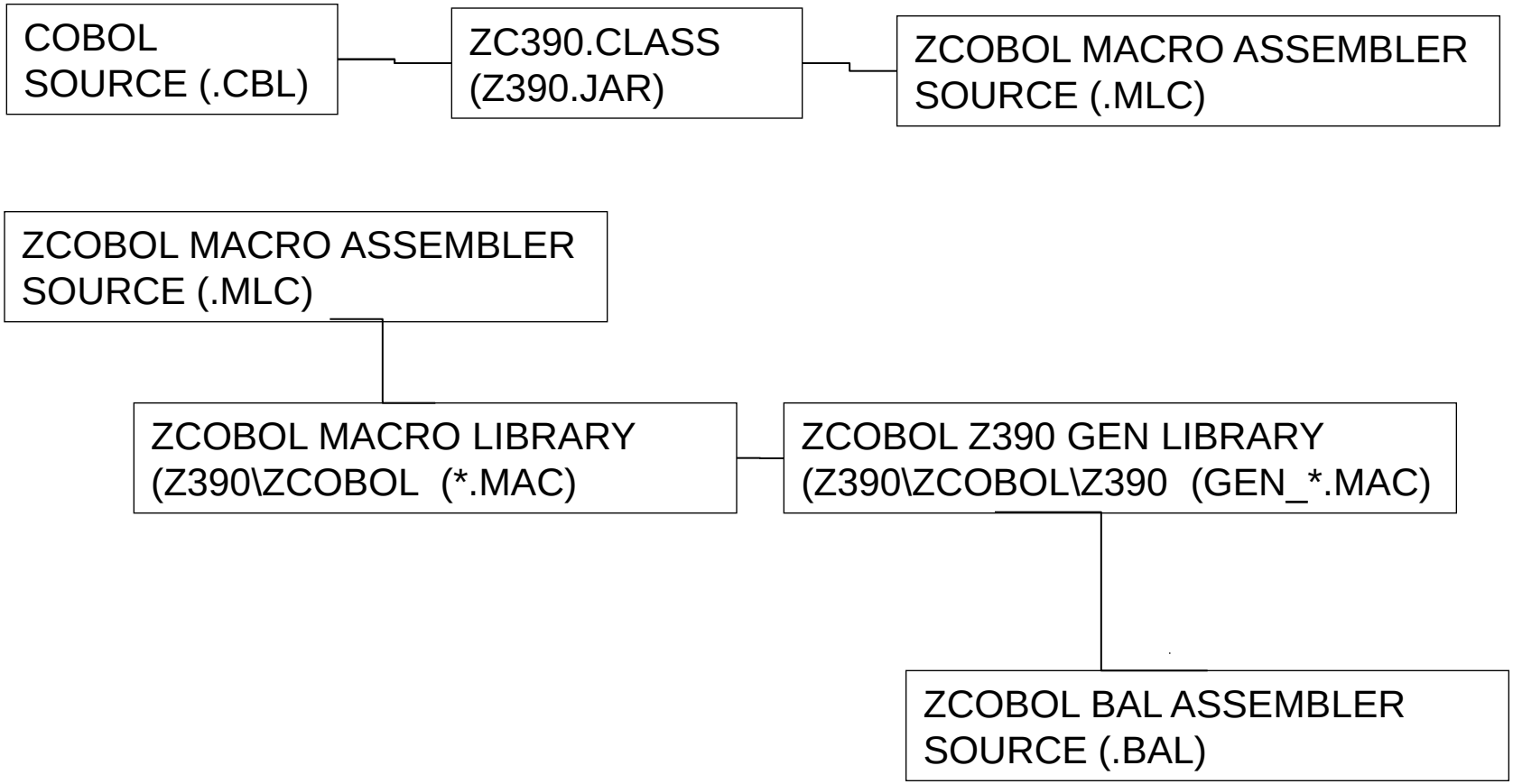
- Version 1.5.0.6 released May 2012
- Bug fixes

The zCOBOL Portable Mainframe Compiler

zCOBOL Portable Mainframe Compiler

- Compiler architecture
- Compiler examples of source code generation
- Compiler code generation
- Compiler commands
- Demo and regression test programs

zCOBOL Portable Mainframe Compiler



zCOBOL Compiler Architecture

- zc390.java parser CBL to MLC macro assembler
- zcobol library for all COBOL verb macros (139)
- zcobol\z390 library for all HLASM gen macros (102)
- Base-free code
  - Temporary regs used for WS and Linkage, some fixed regs used for zCICS
- zcobol\java, vce, & i586 libraries for Java, C++, & HLA/MASM code gen macros (11 each)
  - Note: Once the z390 code gen macros are stabilized, they will be copied to other target language libraries and modified to support other source code generation options.

zCOBOL to z390 code generation Example 1

COBOL SOURCE:

77 CTR-1 COMP PIC S9(9).

01 SYSTEM-DATE.
    02 SYSTEM-DD PIC 99.
    02 SYSTEM-MM PIC 99.

HLASM > MACROS > BAL:

WS 77,CTR_1,COMP,PIC,S9(9)

- GEN_WS
  CTR_1 DS FL4

WS 01,SYSTEM_DATE
WS 02,SYSTEM_DD,PIC,99
WS 02,SYSTEM_MM,PIC,99

- GEN_WS
  - SYSTEM_DATE DS 0CL4
  - SYSTEM_DD DS ZL2
  - SYSTEM_MM DS ZL2

zCOBOL to z390 code generation Example 2

IF CTR_1 = 2 GO TO OPT_2 .

IF CTR_1,=,2

- GEN_COMP

  L R0,CTR_1

  CHI R0,2

- GEN_BC 7,PG_IF_1

  JNE PG_IF_1

GO TO,OPT_2

- GEN_B PG_OPT_2

  J PG_OPT_2

PERIOD

- GEN_LABEL PG_IF_1,ENDIF

  PG_IF_1 DS 0H

ENDIF

zCOBOL Compile Commands

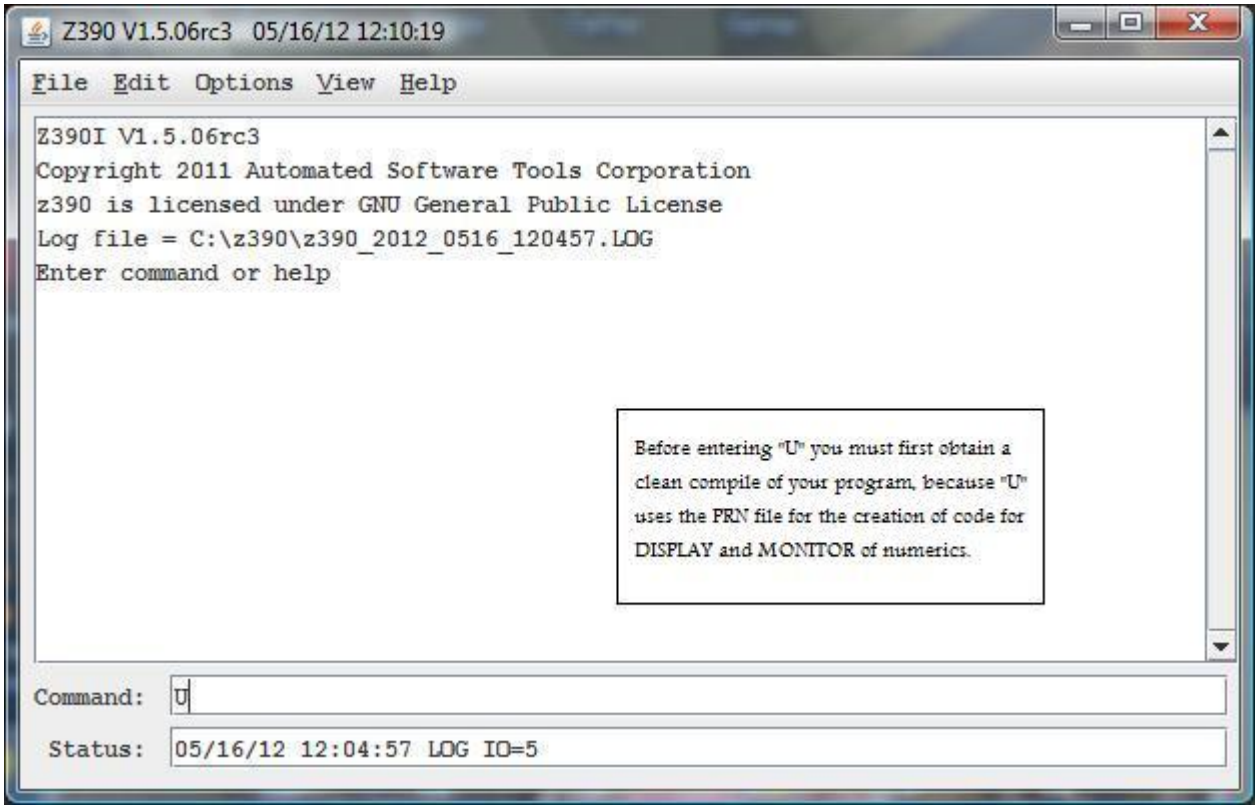
- ZC390C – compile to z390 object code
- ZC390CL – compile and link z390 load module
- ZC390CLG – compile, link, and execute z390 pgm
- ZCJAVCLG – compile and execute Java pgm
- ZCVCECLG – compile, link, and execute C++ pgm
- ZC586CLG – compile, link, and execute MASM pgm
- Note other system software requirements (all free):
  - All require Java run-time and z390 installs
  - ZCVCECLG requires Microsoft Visual Express C++ install
  - ZC586CLG requires HLA and MASM installs

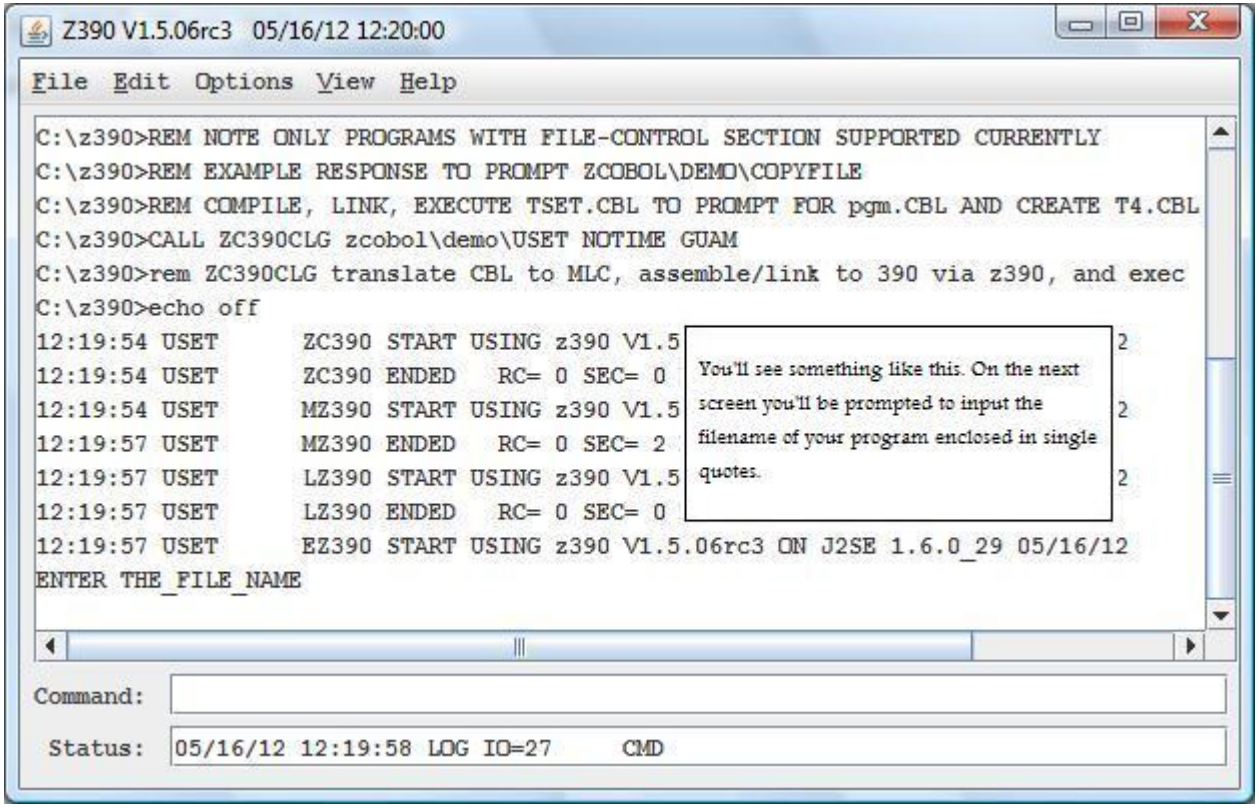
zCOBOL Demo and Regression Tests

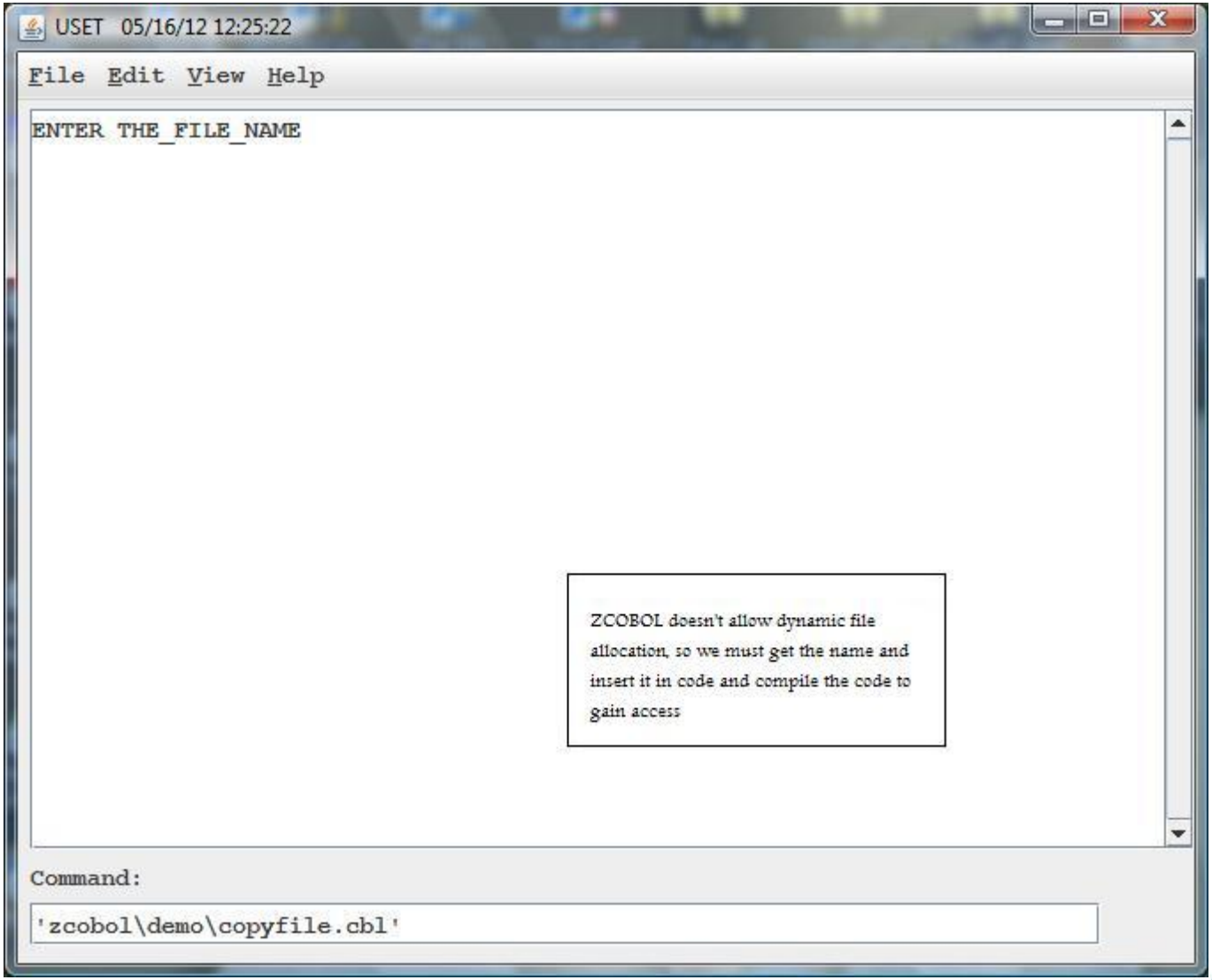
- Demos in zcobol\demo include:
  - HELLO.CBL - display ”Hello World”
  - DATETIME.CBL - display current time and date
  - COPYFILE.CBL - copy line sequential file
- Regression tests in zcobol\test:
  - New tests added for improved verification

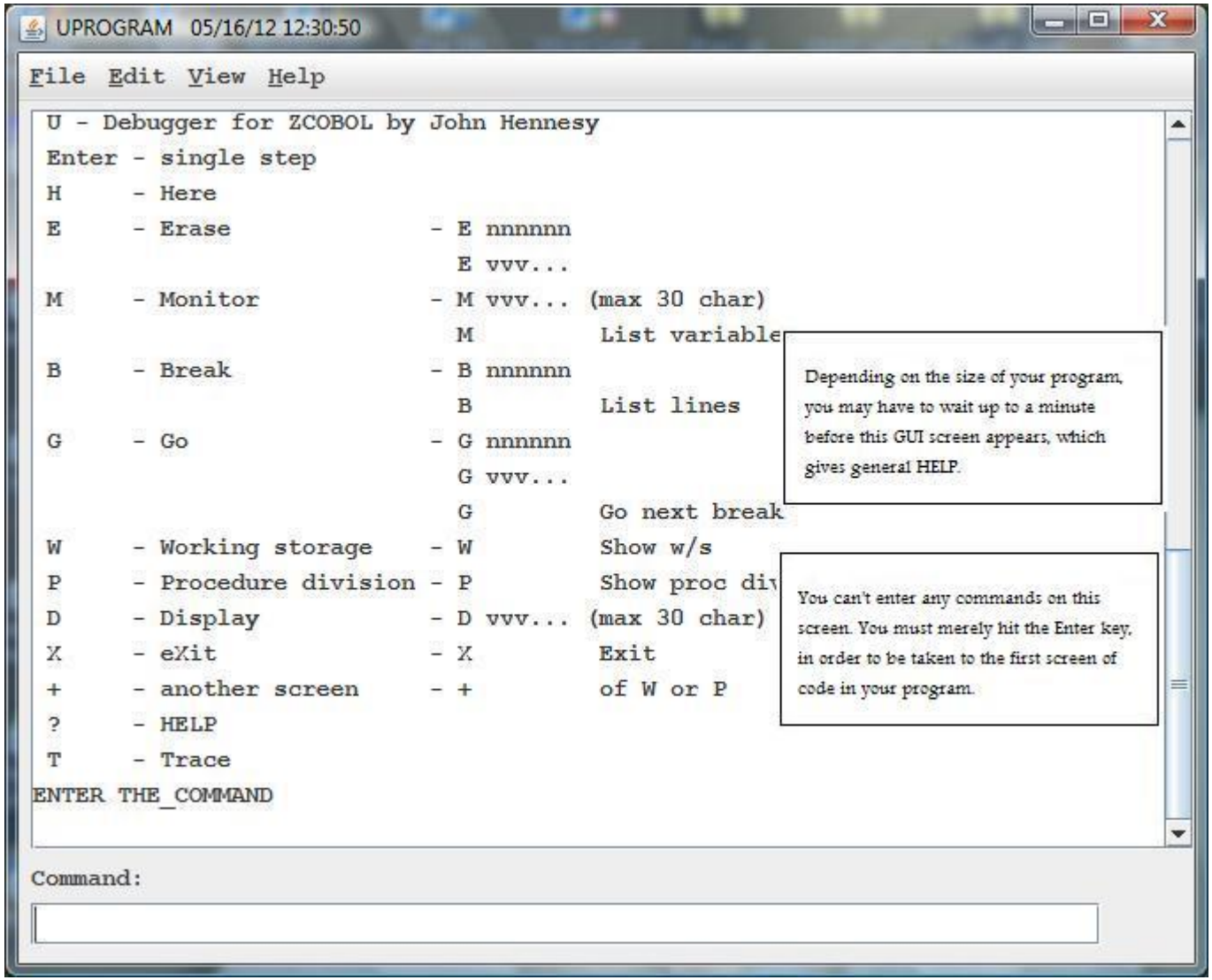
zCOBOL Portable Mainframe Compiler What's new since SHARE Summer 2011

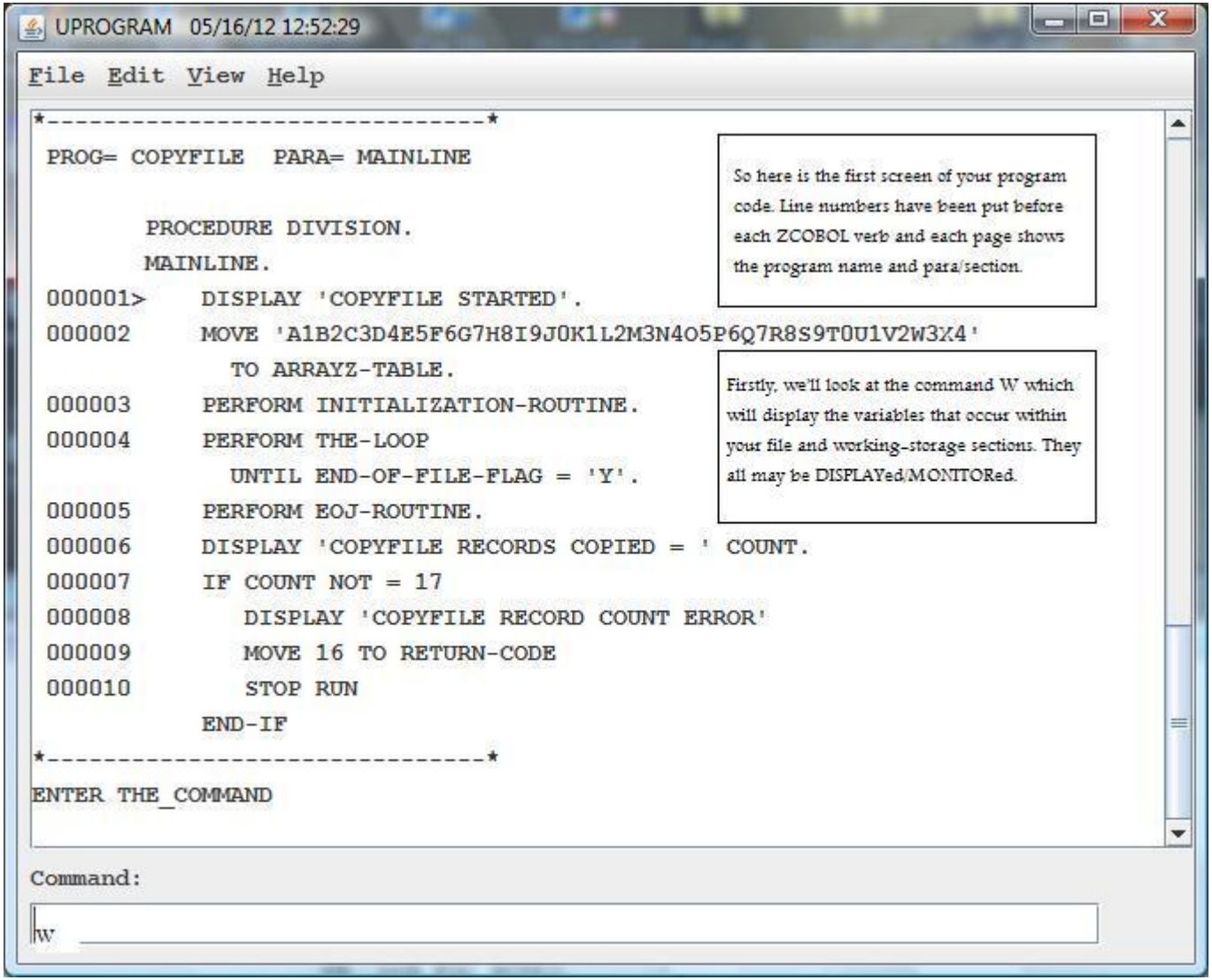
- Interactive Debugger “U” by John Hennesy
- Written in zCOBOL
- Distributed starting with current release
- Demonstration slides courtesy of Mr. Hennesy


Z390 V1.5.06rc3 05/16/12 12:10:19
File Edit Options View Help
Z390I V1.5.06rc3
Copyright 2011 Automated Software Tools Corporation
z390 is licensed under GNU General Public License
Log file = C:\z390\z390_2012_0516_120457.LOG
Enter command or help
Before entering "U" you must first obtain a clean compile of your program, because "U" uses the PRN file for the creation of code for DISPLAY and MONITOR of numerics.
Command: U
Status: 05/16/12 12:04:57 LOG IO=5


Z390 V1.5.06rc3 05/16/12 12:20:00
File Edit Options View Help
C:\z390>REM NOTE ONLY PROGRAMS WITH FILE-CONTROL SECTION SUPPORTED CURRENTLY
C:\z390>REM EXAMPLE RESPONSE TO PROMPT ZCOBOL\DEMO\COPYFILE
C:\z390>REM COMPILE, LINK, EXECUTE TSET.CBL TO PROMPT FOR pgm.CBL AND CREATE T4.CBL
C:\z390>CALL ZC390CLG zcobol\demo\USET NOTIME GUAM
C:\z390>rem ZC390CLG translate CBL to MLC, assemble/link to 390 via z390, and exec
C:\z390>echo off
12:19:54 USET ZC390 START USING z390 V1.5
12:19:54 USET ZC390 ENDED RC= 0 SEC= 0
12:19:54 USET MZ390 START USING z390 V1.5
12:19:57 USET MZ390 ENDED RC= 0 SEC= 2
12:19:57 USET LZ390 START USING z390 V1.5
12:19:57 USET LZ390 ENDED RC= 0 SEC= 0
12:19:57 USET EZ390 START USING z390 V1.5.06rc3 ON J2SE 1.6.0_29 05/16/12
ENTER THE_FILE_NAME
You'll see something like this. On the next screen you'll be prompted to input the filename of your program enclosed in single quotes.
Command:
Status: 05/16/12 12:19:58 LOG IO=27 CMD

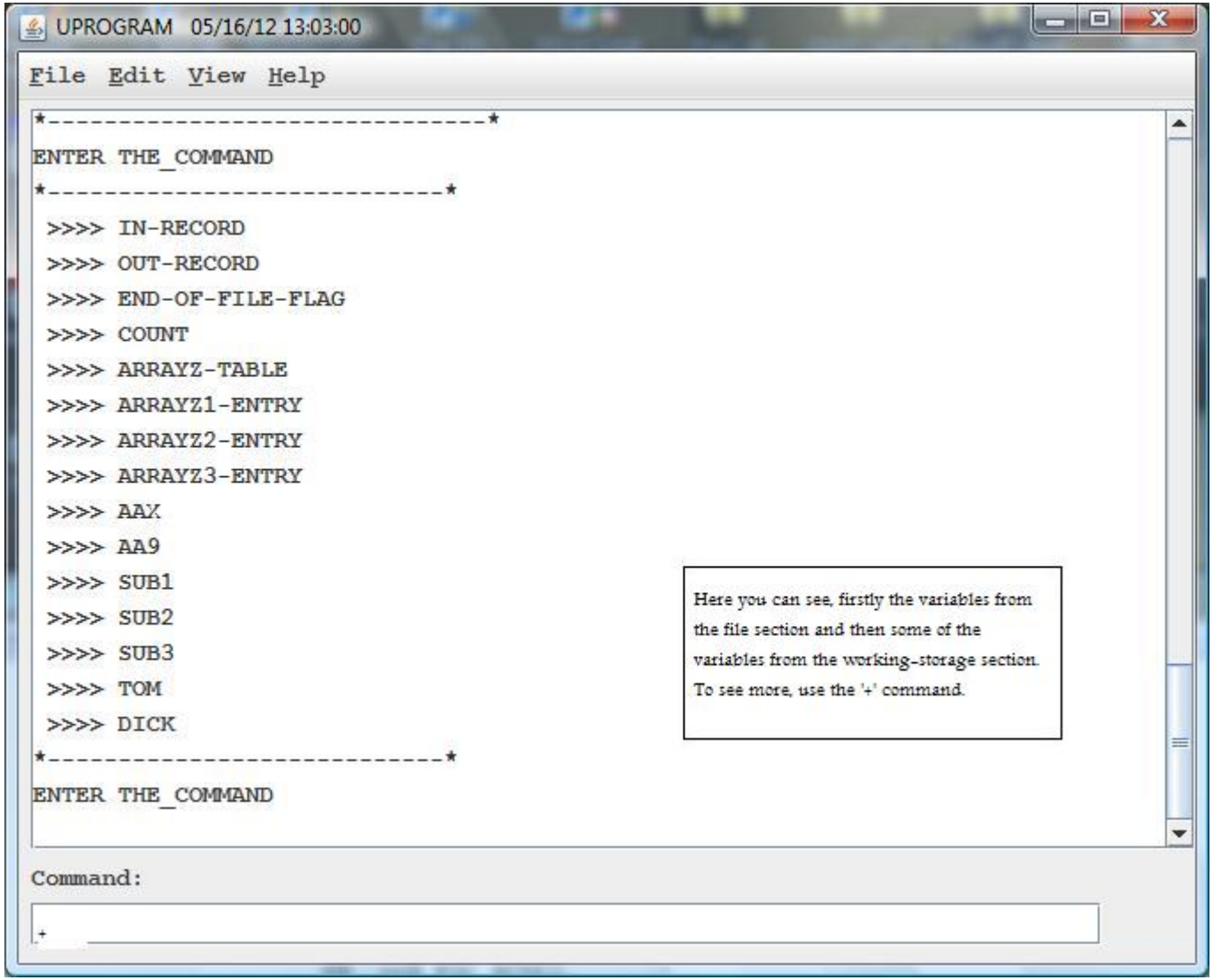

USET 05/16/12 12:25:22
File Edit View Help
ENTER THE_FILE_NAME
ZCOBOL doesn't allow dynamic file allocation, so we must get the name and insert it in code and compile the code to gain access
Command:
'zcobol\demo\copyfile.cbl'


UPROGRAM 05/16/12 12:30:50
File Edit View Help
U - Debugger for ZCOBOL by John Hennesy
Enter - single step
H - Here
E - Erase - E nnnnnn
E vvv...
M - Monitor - M vvv... (max 30 char)
M List variable
B - Break - B nnnnnn
B List lines
G - Go - G nnnnnn
G vvv...
G Go next break
W - Working storage - W Show w/s
P - Procedure division - P Show proc div
D - Display - D vvv... (max 30 char)
X - eXit - X Exit
+ - another screen - + of W or P
? - HELP
T - Trace
ENTER THE_COMMAND
Command:
Depending on the size of your program, you may have to wait up to a minute before this GUI screen appears, which gives general HELP.
You can't enter any commands on this screen. You must merely hit the Enter key, in order to be taken to the first screen of code in your program.


UPROGRAM 05/16/12 12:52:29
File Edit View Help

PROG= COPYFILE PARA= MAINLINE
PROCEDURE DIVISION.
MAINLINE.
000001> DISPLAY 'COPYFILE STARTED'.
000002 MOVE 'A1B2C3D4E5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X4'
TO ARRAYZ-TABLE.
000003 PERFORM INITIALIZATION-ROUTINE.
000004 PERFORM THE-LOOP
UNTIL END-OF-FILE-FLAG = 'Y'.
000005 PERFORM EOJ-ROUTINE.
000006 DISPLAY 'COPYFILE RECORDS COPIED = ' COUNT.
000007 IF COUNT NOT = 17
000008 DISPLAY 'COPYFILE RECORD COUNT ERROR'
000009 MOVE 16 TO RETURN-CODE
000010 STOP RUN
END-IF

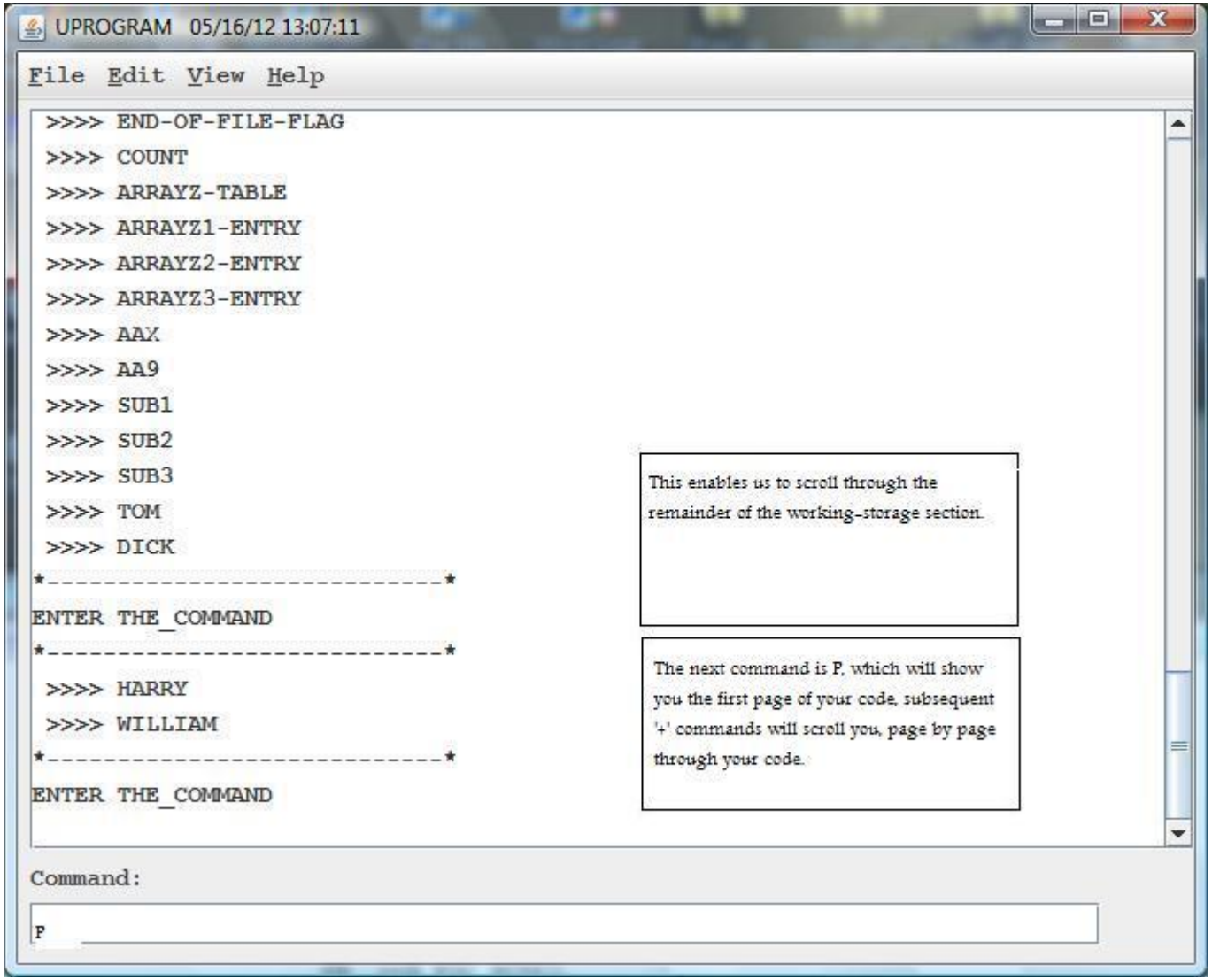
ENTER THE_COMMAND
So here is the first screen of your program code. Line numbers have been put before each ZCOBOL verb and each page shows the program name and para/section.
Firstly, we'll look at the command W which will display the variables that occur within your file and working-storage sections. They all may be DISPLAYed/MONITORed.
Command:
w


UPROGRAM 05/16/12 13:03:00
File Edit View Help

ENTER THE_COMMAND

>>>> IN-RECORD
>>>> OUT-RECORD
>>>> END-OF-FILE-FLAG
>>>> COUNT
>>>> ARRAYZ-TABLE
>>>> ARRAYZ1-ENTRY
>>>> ARRAYZ2-ENTRY
>>>> ARRAYZ3-ENTRY
>>>> AAX
>>>> AA9
>>>> SUB1
>>>> SUB2
>>>> SUB3
>>>> TOM
>>>> DICK

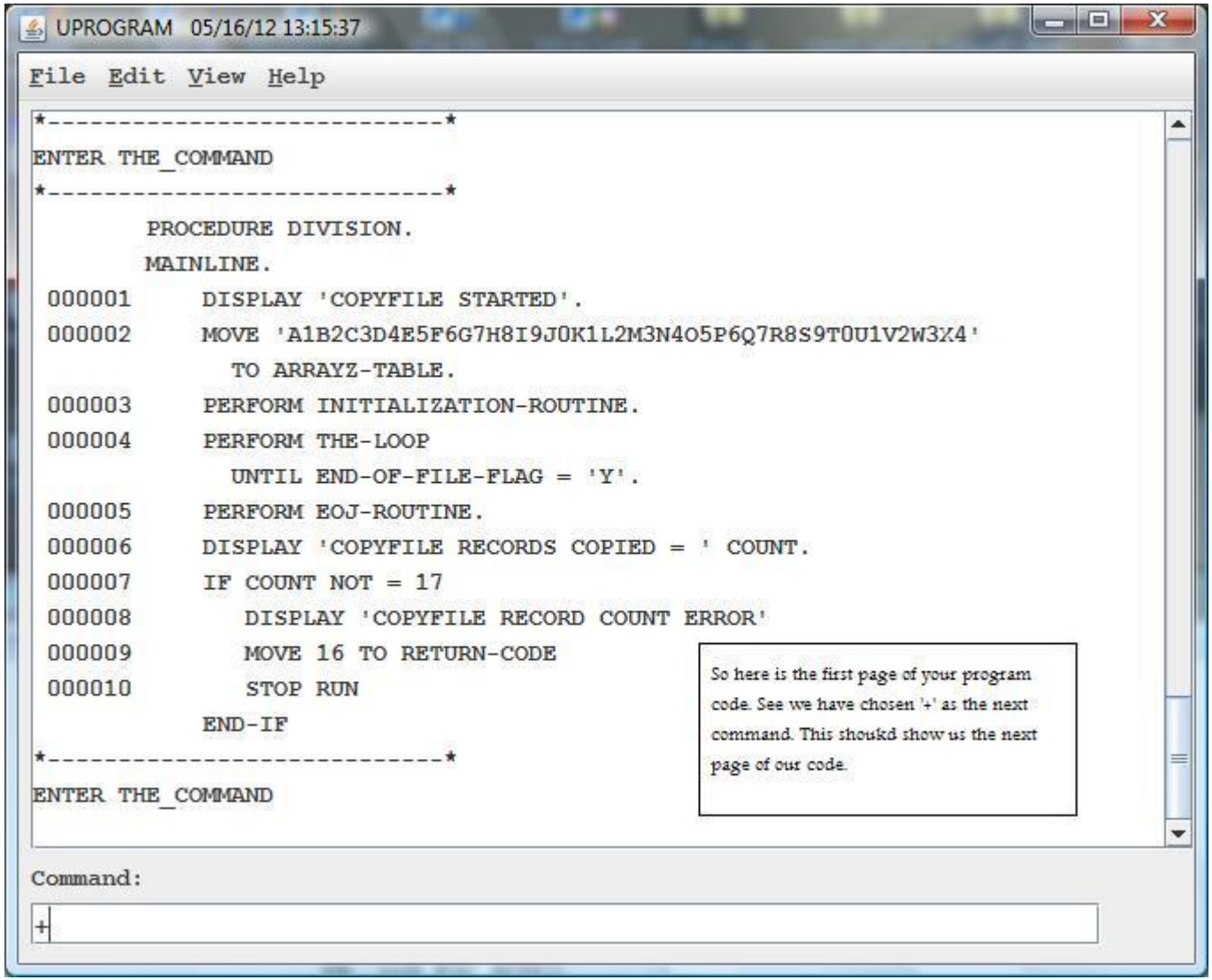
ENTER THE_COMMAND
Here you can see, firstly the variables from the file section and then some of the variables from the working-storage section. To see more, use the '+' command.
Command:
+


UPROGRAM 05/16/12 13:07:11
File Edit View Help
>>>> END-OF-FILE-FLAG
>>>> COUNT
>>>> ARRAYZ-TABLE
>>>> ARRAYZ1-ENTRY
>>>> ARRAYZ2-ENTRY
>>>> ARRAYZ3-ENTRY
>>>> AAX
>>>> AA9
>>>> SUB1
>>>> SUB2
>>>> SUB3
>>>> TOM
>>>> DICK

ENTER THE_COMMAND

>>>> HARRY
>>>> WILLIAM

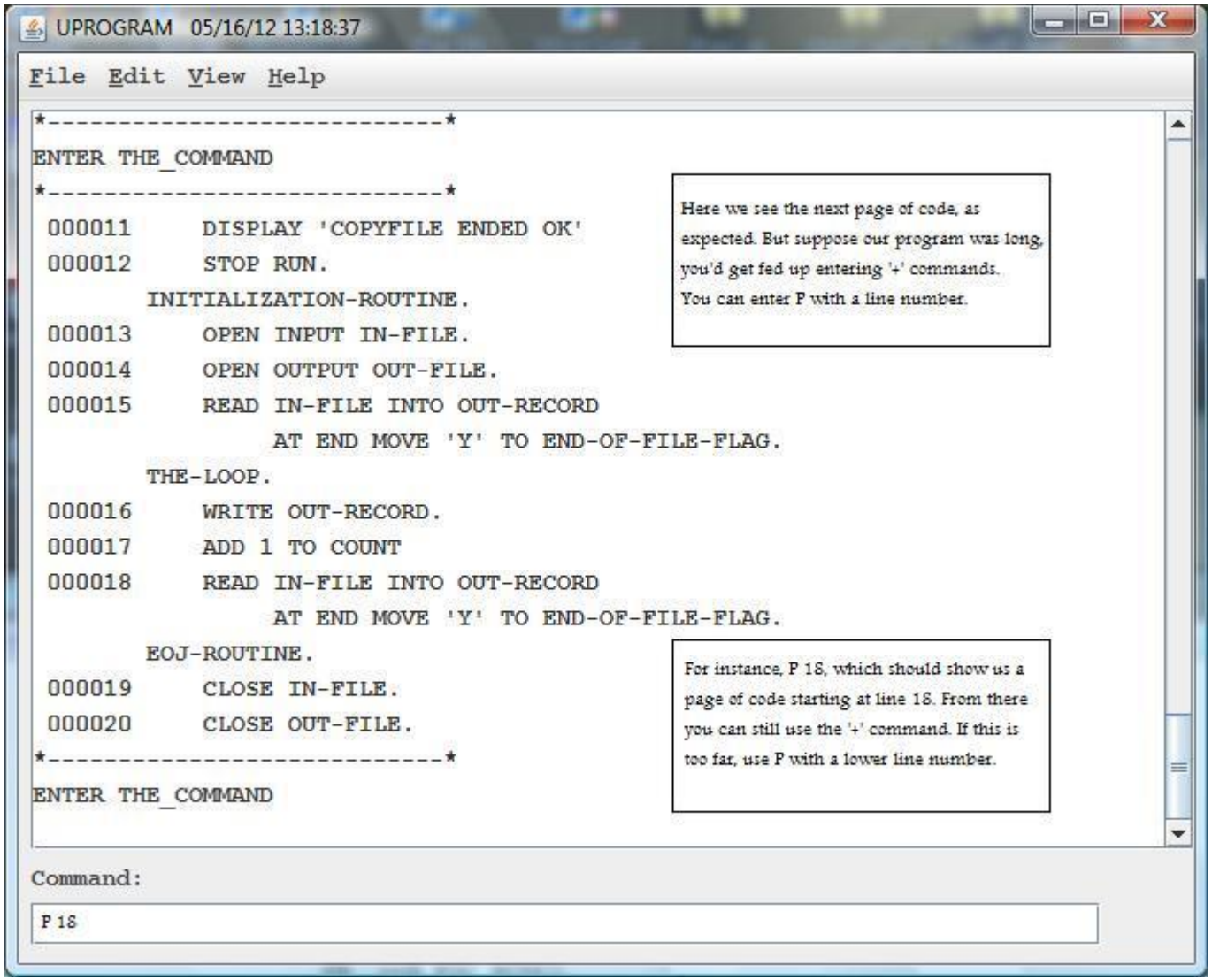
ENTER THE_COMMAND
This enables us to scroll through the remainder of the working-storage section.
The next command is P, which will show you the first page of your code, subsequent '+' commands will scroll you, page by page through your code.
Command:
P


UPROGRAM 05/16/12 13:15:37
File Edit View Help

ENTER THE_COMMAND

PROCEDURE DIVISION.
MAINLINE.
000001 DISPLAY 'COPYFILE STARTED'.
000002 MOVE 'A1B2C3D4E5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X4'
TO ARRAYZ-TABLE.
000003 PERFORM INITIALIZATION-ROUTINE.
000004 PERFORM THE-LOOP
UNTIL END-OF-FILE-FLAG = 'Y'.
000005 PERFORM EOJ-ROUTINE.
000006 DISPLAY 'COPYFILE RECORDS COPIED = ' COUNT.
000007 IF COUNT NOT = 17
000008 DISPLAY 'COPYFILE RECORD COUNT ERROR'
000009 MOVE 16 TO RETURN-CODE
000010 STOP RUN
END-IF

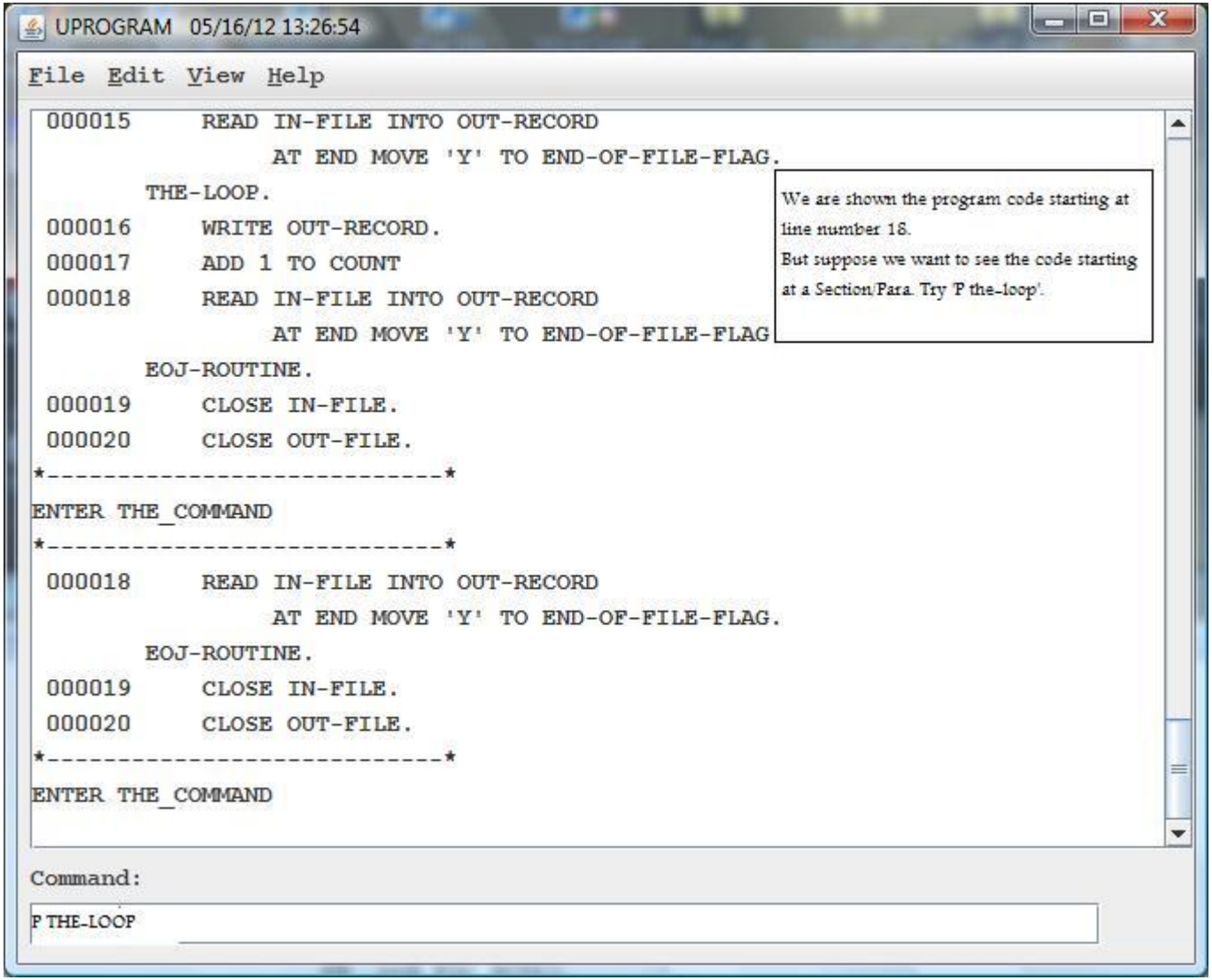
ENTER THE_COMMAND
So here is the first page of your program code. See we have chosen '+' as the next command. This shoukd show us the next page of our code.
Command:
+


UPROGRAM 05/16/12 13:18:37
File Edit View Help

ENTER THE_COMMAND

000011 DISPLAY 'COPYFILE ENDED OK'
000012 STOP RUN.
INITIALIZATION-ROUTINE.
000013 OPEN INPUT IN-FILE.
000014 OPEN OUTPUT OUT-FILE.
000015 READ IN-FILE INTO OUT-RECORD
AT END MOVE 'Y' TO END-OF-FILE-FLAG.
THE-LOOP.
000016 WRITE OUT-RECORD.
000017 ADD 1 TO COUNT
000018 READ IN-FILE INTO OUT-RECORD
AT END MOVE 'Y' TO END-OF-FILE-FLAG.
EOJ-ROUTINE.
000019 CLOSE IN-FILE.
000020 CLOSE OUT-FILE.

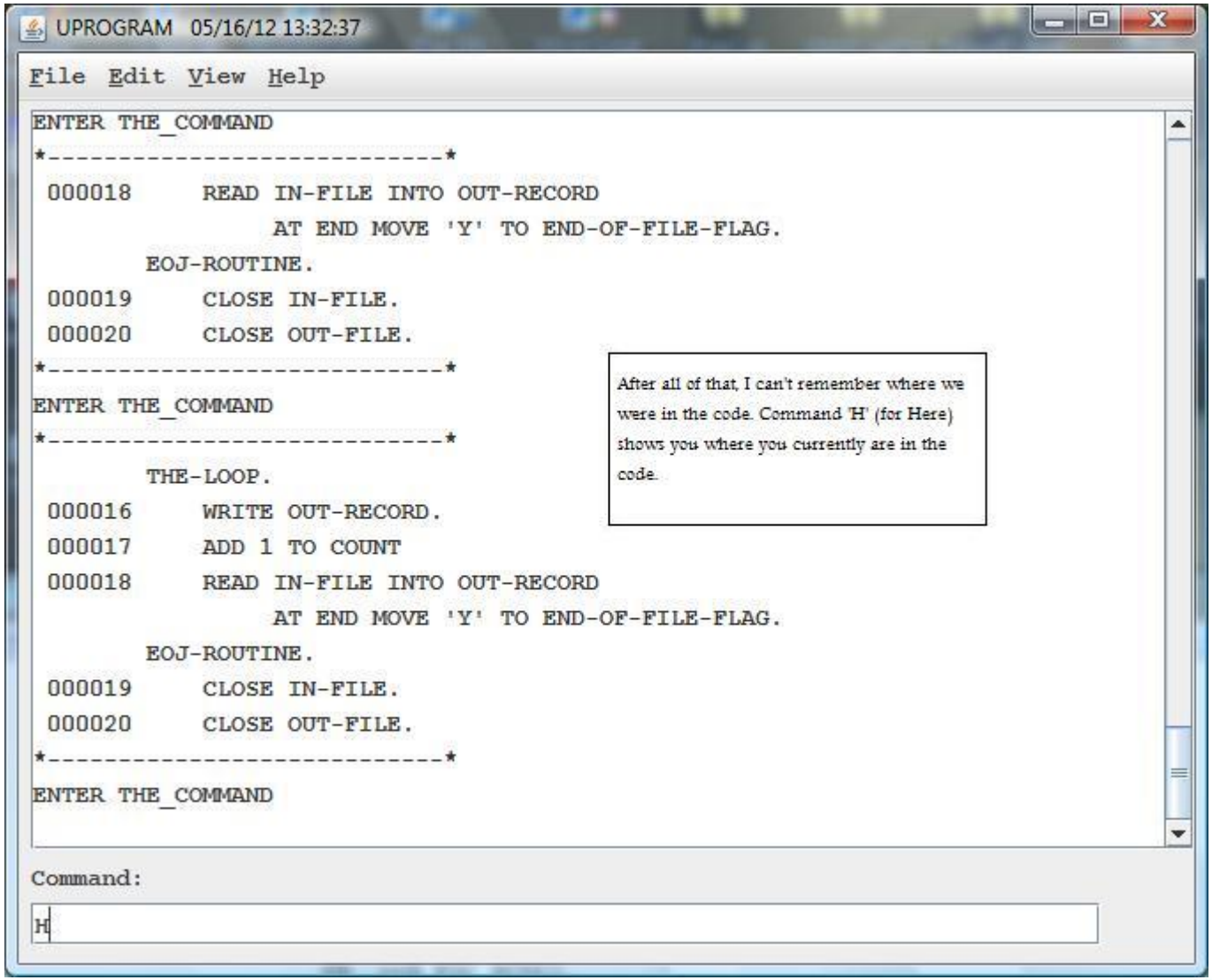
ENTER THE_COMMAND
Here we see the next page of code, as expected. But suppose our program was long, you'd get fed up entering '+' commands. You can enter P with a line number.
For instance, P 18, which should show us a page of code starting at line 18. From there you can still use the '+' command. If this is too far, use P with a lower line number.
Command:
P 18


UPROGRAM 05/16/12 13:26:54
File Edit View Help
000015 READ IN-FILE INTO OUT-RECORD
AT END MOVE 'Y' TO END-OF-FILE-FLAG.
THE-LOOP.
000016 WRITE OUT-RECORD.
000017 ADD 1 TO COUNT
000018 READ IN-FILE INTO OUT-RECORD
AT END MOVE 'Y' TO END-OF-FILE-FLAG
EOJ-ROUTINE.
000019 CLOSE IN-FILE.
000020 CLOSE OUT-FILE.

ENTER THE_COMMAND

000018 READ IN-FILE INTO OUT-RECORD
AT END MOVE 'Y' TO END-OF-FILE-FLAG.
EOJ-ROUTINE.
000019 CLOSE IN-FILE.
000020 CLOSE OUT-FILE.

ENTER THE_COMMAND
We are shown the program code starting at line number 18.
But suppose we want to see the code starting at a Section/Para. Try 'P the-loop'.
Command:
P THE-LOOP

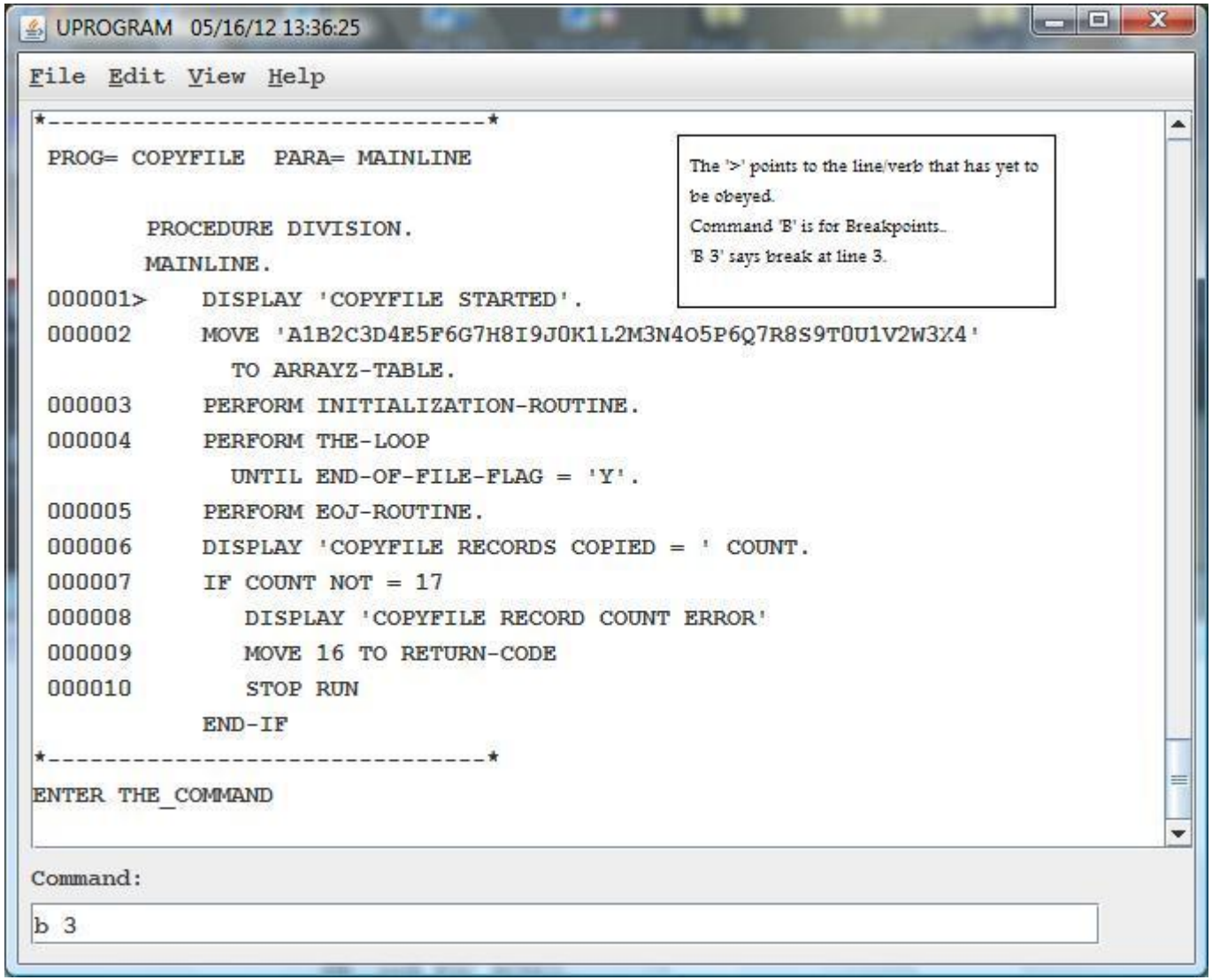

UPROGRAM 05/16/12 13:32:37
File Edit View Help
ENTER THE_COMMAND

000018 READ IN-FILE INTO OUT-RECORD
AT END MOVE 'Y' TO END-OF-FILE-FLAG.
EOJ-ROUTINE.
000019 CLOSE IN-FILE.
000020 CLOSE OUT-FILE.

ENTER THE_COMMAND

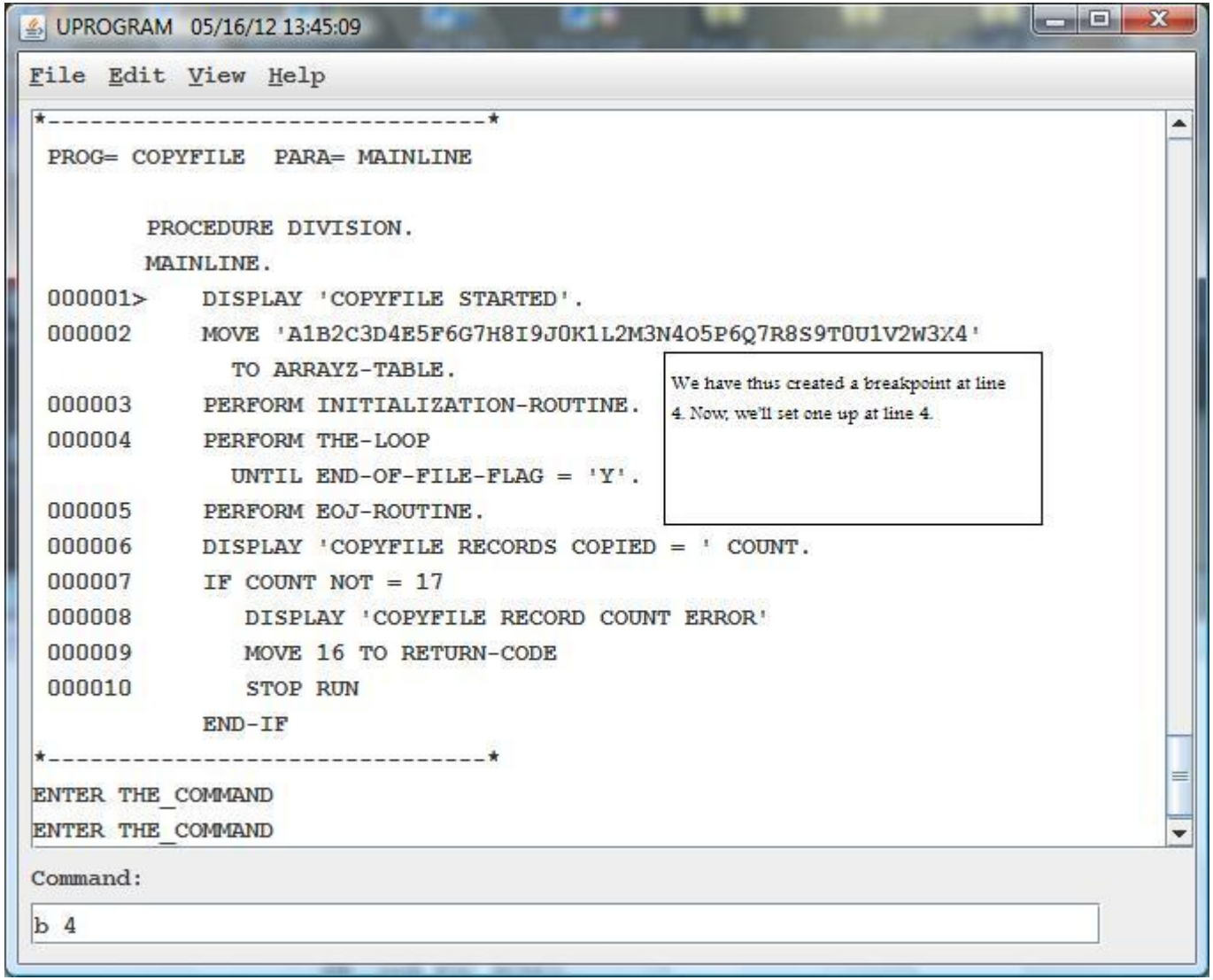
THE-LOOP.
000016 WRITE OUT-RECORD.
000017 ADD 1 TO COUNT
000018 READ IN-FILE INTO OUT-RECORD
AT END MOVE 'Y' TO END-OF-FILE-FLAG.
EOJ-ROUTINE.
000019 CLOSE IN-FILE.
000020 CLOSE OUT-FILE.

ENTER THE_COMMAND
Command:
H
After all of that, I can't remember where we were in the code. Command 'H' (for Here) shows you where you currently are in the code.


UPROGRAM 05/16/12 13:36:25
File Edit View Help

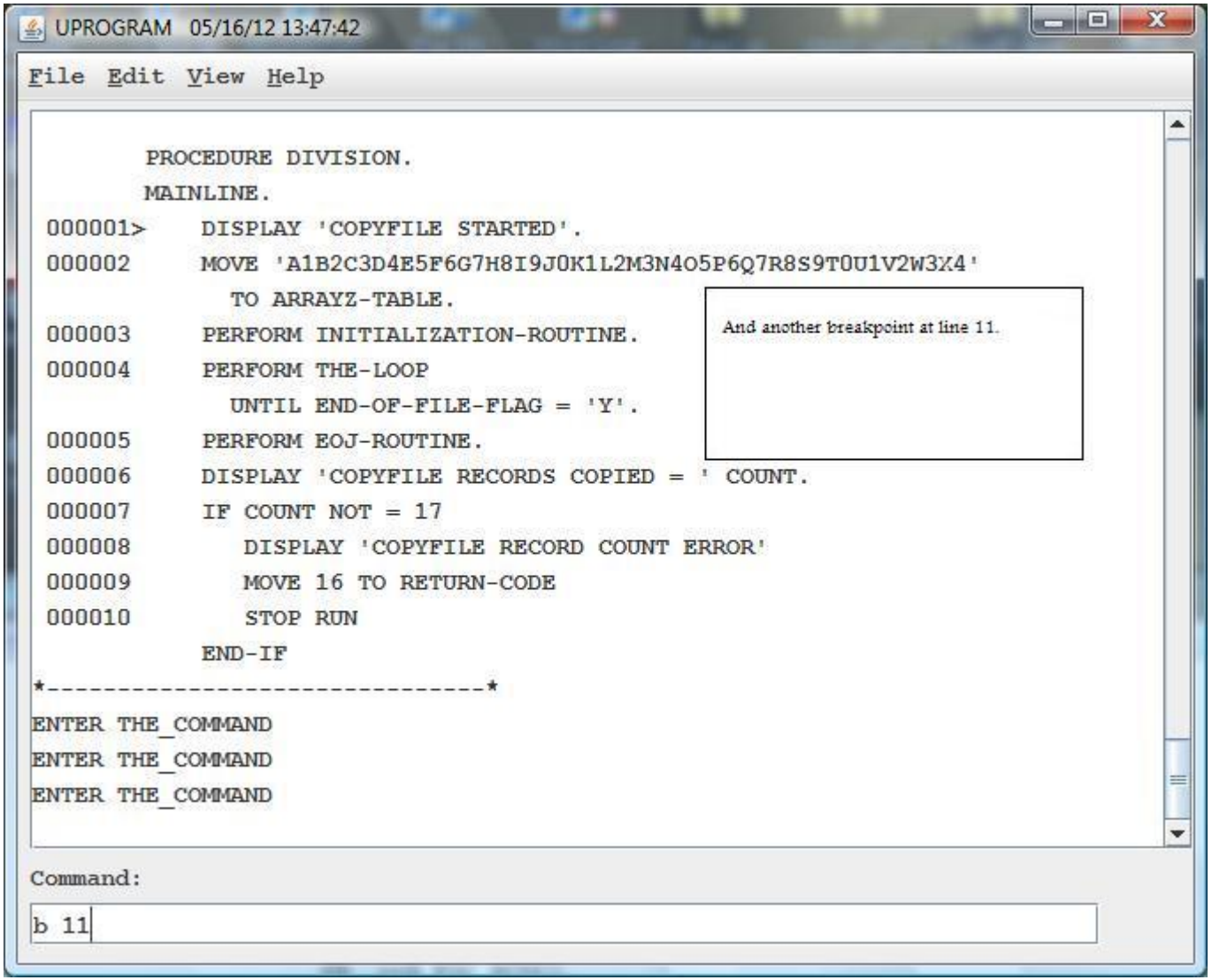
PROG= COPYFILE PARA= MAINLINE
PROCEDURE DIVISION.
MAINLINE.
000001> DISPLAY 'COPYFILE STARTED'.
000002 MOVE 'A1B2C3D4E5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X4'
TO ARRAYZ-TABLE.
000003 PERFORM INITIALIZATION-ROUTINE.
000004 PERFORM THE-LOOP
UNTIL END-OF-FILE-FLAG = 'Y'.
000005 PERFORM EOJ-ROUTINE.
000006 DISPLAY 'COPYFILE RECORDS COPIED = ' COUNT.
000007 IF COUNT NOT = 17
000008 DISPLAY 'COPYFILE RECORD COUNT ERROR'
000009 MOVE 16 TO RETURN-CODE
000010 STOP RUN
END-IF

ENTER THE_COMMAND
The '>' points to the line/verb that has yet to be obeyed.
Command 'B' is for Breakpoints..
'B 3' says break at line 3.
Command:
b 3

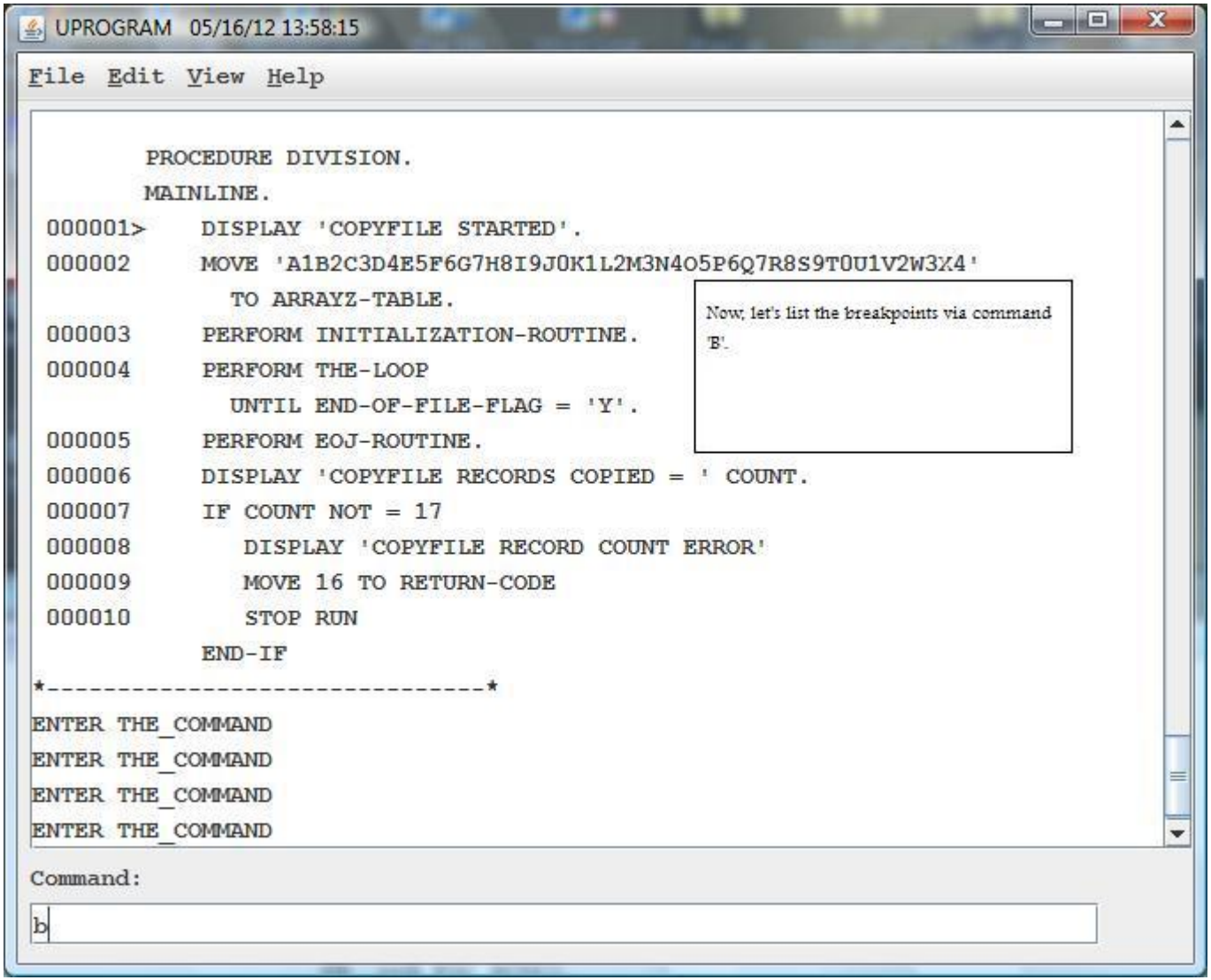

UPROGRAM 05/16/12 13:45:09
File Edit View Help

PROG= COPYFILE PARA= MAINLINE
PROCEDURE DIVISION.
MAINLINE.
000001> DISPLAY 'COPYFILE STARTED'.
000002 MOVE 'A1B2C3D4E5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X4'
TO ARRAYZ-TABLE.
000003 PERFORM INITIALIZATION-ROUTINE.
000004 PERFORM THE-LOOP
UNTIL END-OF-FILE-FLAG = 'Y'.
000005 PERFORM EOJ-ROUTINE.
000006 DISPLAY 'COPYFILE RECORDS COPIED = ' COUNT.
000007 IF COUNT NOT = 17
000008 DISPLAY 'COPYFILE RECORD COUNT ERROR'
000009 MOVE 16 TO RETURN-CODE
000010 STOP RUN
END-IF

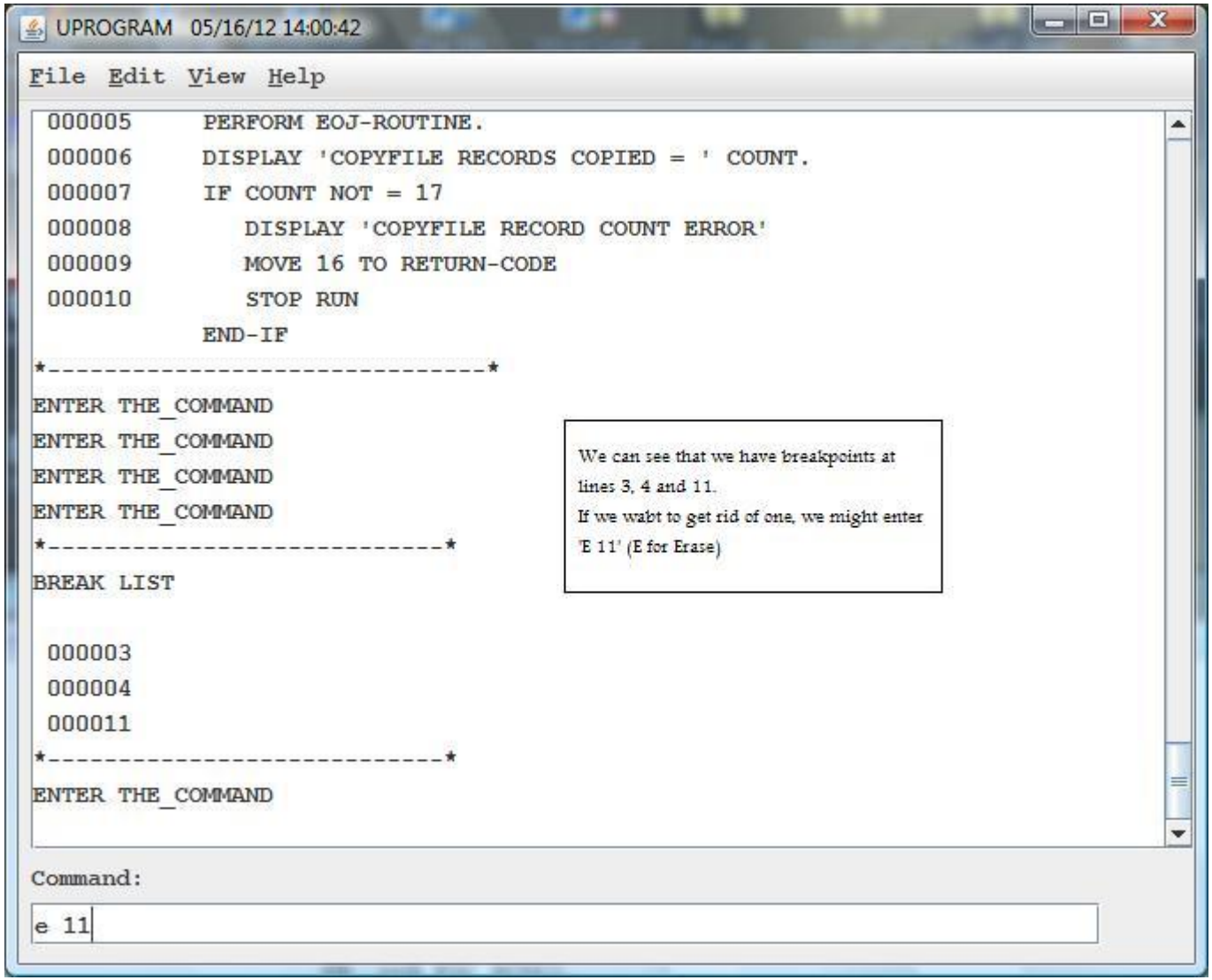
ENTER THE_COMMAND
ENTER THE_COMMAND
Command:
b 4
We have thus created a breakpoint at line 4. Now, we'll set one up at line 4.


UPROGRAM 05/16/12 13:47:42
File Edit View Help
PROCEDURE DIVISION.
MAINLINE.
000001> DISPLAY 'COPYFILE STARTED'.
000002 MOVE 'A1B2C3D4E5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X4'
TO ARRAYZ-TABLE.
000003 PERFORM INITIALIZATION-ROUTINE.
000004 PERFORM THE-LOOP
UNTIL END-OF-FILE-FLAG = 'Y'.
000005 PERFORM EOJ-ROUTINE.
000006 DISPLAY 'COPYFILE RECORDS COPIED = ' COUNT.
000007 IF COUNT NOT = 17
000008 DISPLAY 'COPYFILE RECORD COUNT ERROR'
000009 MOVE 16 TO RETURN-CODE
000010 STOP RUN
END-IF

ENTER THE_COMMAND
ENTER THE_COMMAND
ENTER THE_COMMAND
And another breakpoint at line 11.
Command:
b 11


UPROGRAM 05/16/12 13:58:15
File Edit View Help
PROCEDURE DIVISION.
MAINLINE.
000001> DISPLAY 'COPYFILE STARTED'.
000002 MOVE 'A1B2C3D4E5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X4'
TO ARRAYZ-TABLE.
000003 PERFORM INITIALIZATION-ROUTINE.
000004 PERFORM THE-LOOP
UNTIL END-OF-FILE-FLAG = 'Y'.
000005 PERFORM EOJ-ROUTINE.
000006 DISPLAY 'COPYFILE RECORDS COPIED = ' COUNT.
000007 IF COUNT NOT = 17
000008 DISPLAY 'COPYFILE RECORD COUNT ERROR'
000009 MOVE 16 TO RETURN-CODE
000010 STOP RUN
END-IF

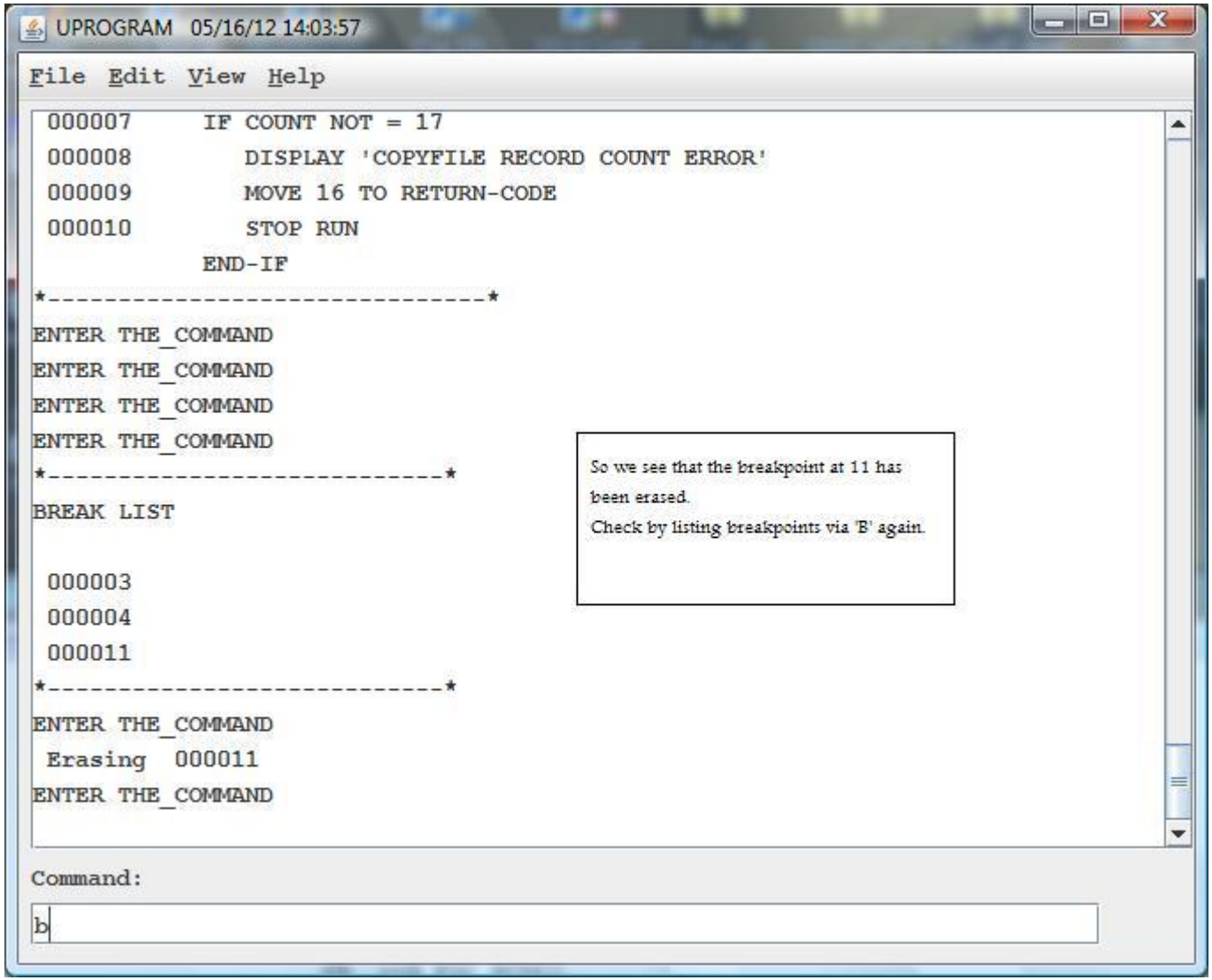
ENTER THE_COMMAND
ENTER THE_COMMAND
ENTER THE_COMMAND
ENTER THE_COMMAND
Command:
b
Now, let's list the breakpoints via command 'B'.


UPROGRAM 05/16/12 14:00:42
File Edit View Help
000005 PERFORM EOJ-ROUTINE.
000006 DISPLAY 'COPYFILE RECORDS COPIED = ' COUNT.
000007 IF COUNT NOT = 17
000008 DISPLAY 'COPYFILE RECORD COUNT ERROR'
000009 MOVE 16 TO RETURN-CODE
000010 STOP RUN
END-IF

ENTER THE_COMMAND
ENTER THE_COMMAND
ENTER THE_COMMAND
ENTER THE_COMMAND

BREAK LIST
000003
000004
000011

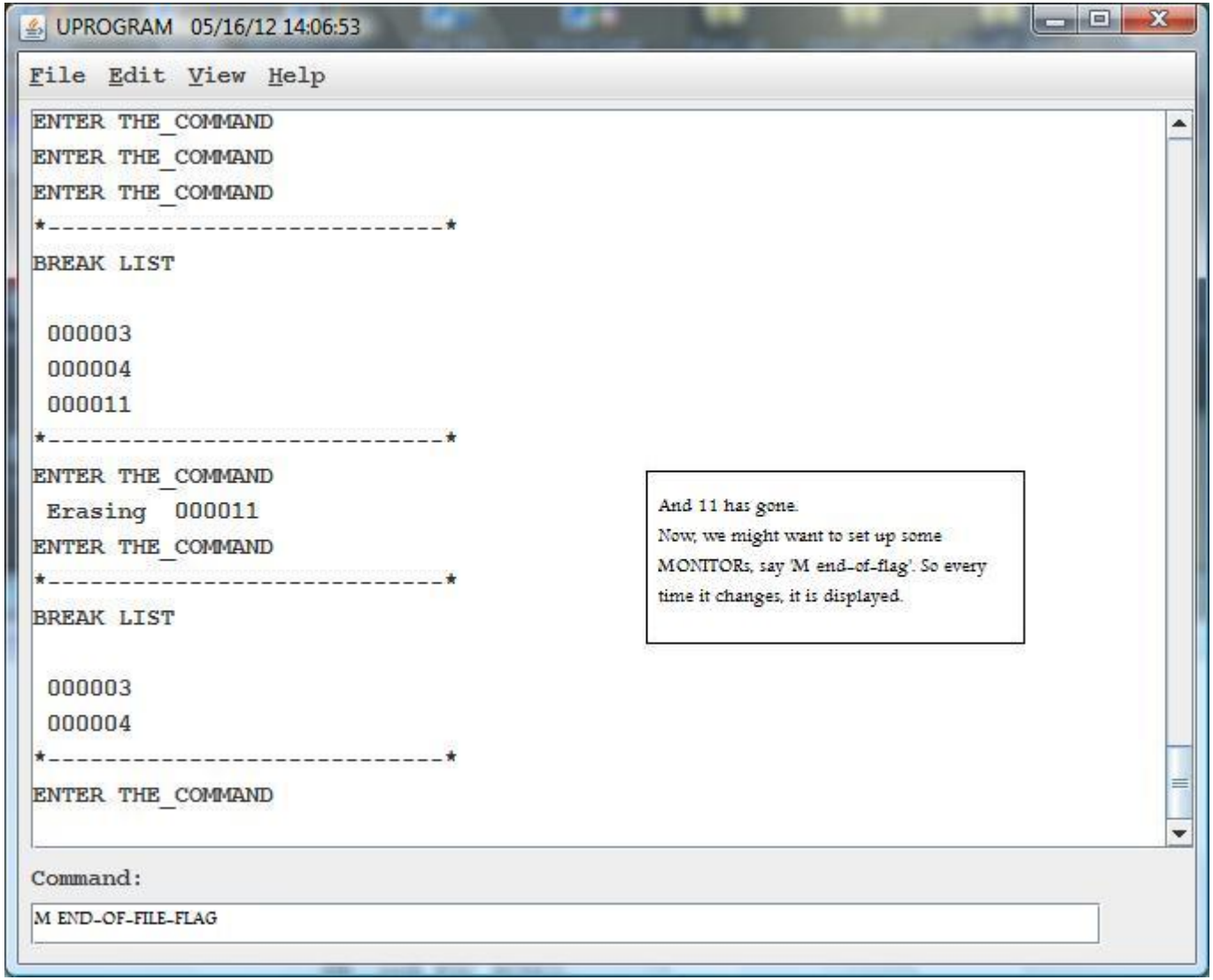
ENTER THE_COMMAND
We can see that we have breakpoints at lines 3, 4 and 11.
If we wabt to get rid of one, we might enter 'E 11' (E for Erase)
Command:
e 11


UPROGRAM 05/16/12 14:03:57
File Edit View Help
000007 IF COUNT NOT = 17
000008 DISPLAY 'COPYFILE RECORD COUNT ERROR'
000009 MOVE 16 TO RETURN-CODE
000010 STOP RUN
END-IF

ENTER THE_COMMAND
ENTER THE_COMMAND
ENTER THE_COMMAND
ENTER THE_COMMAND

BREAK LIST
000003
000004
000011

ENTER THE_COMMAND
Erasing 000011
ENTER THE_COMMAND
So we see that the breakpoint at 11 has been erased.
Check by listing breakpoints via 'B' again.
Command:
b

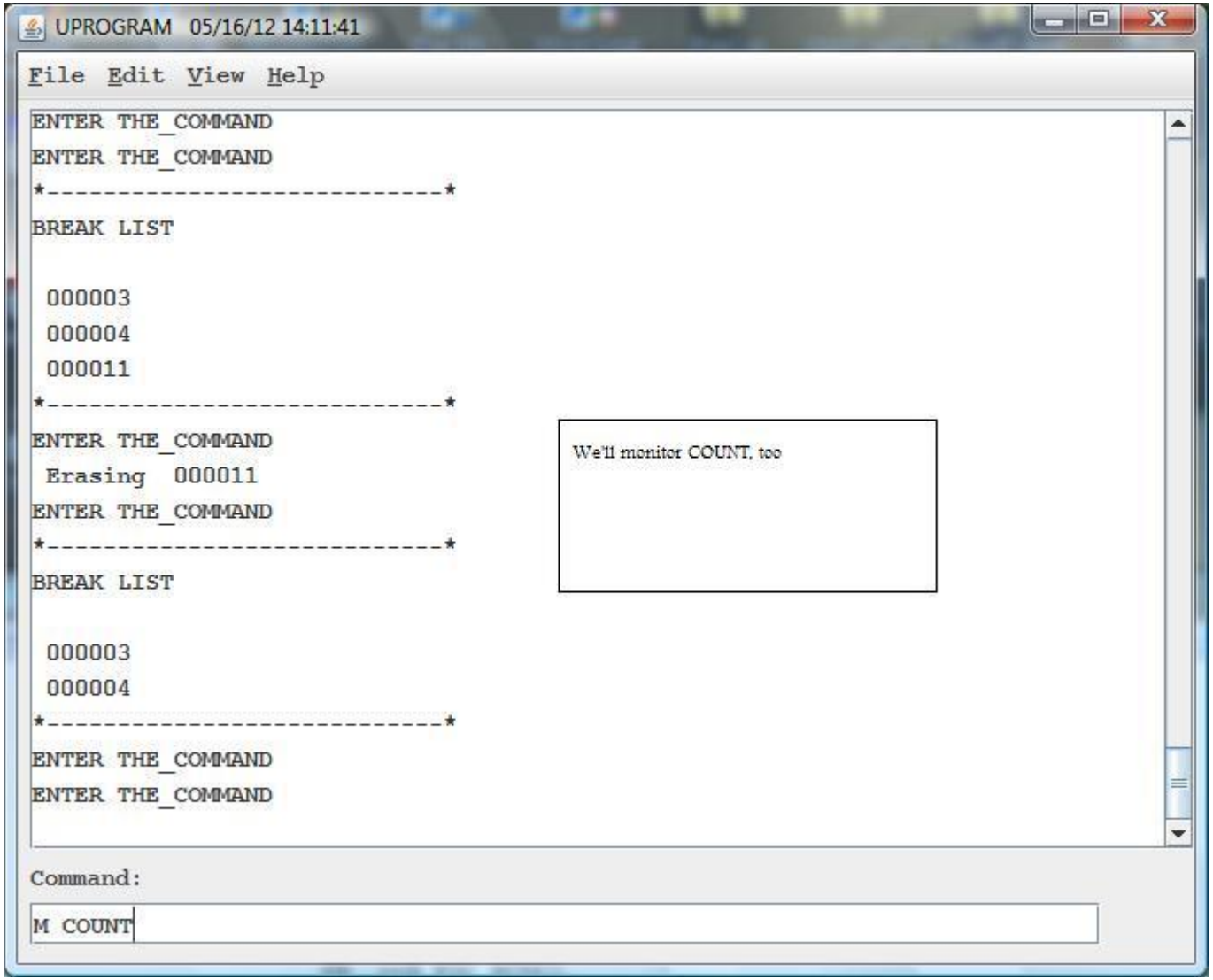

UPROGRAM 05/16/12 14:06:53
File Edit View Help
ENTER THE_COMMAND
ENTER THE_COMMAND
ENTER THE_COMMAND

BREAK LIST
000003
000004
000011

ENTER THE_COMMAND
Erasing 000011
ENTER THE_COMMAND

BREAK LIST
000003
000004

ENTER THE_COMMAND
Command:
M END-OF-FILE-FLAG
And 11 has gone.
Now, we might want to set up some MONITORs, say 'M end-of-flag'. So every time it changes, it is displayed.

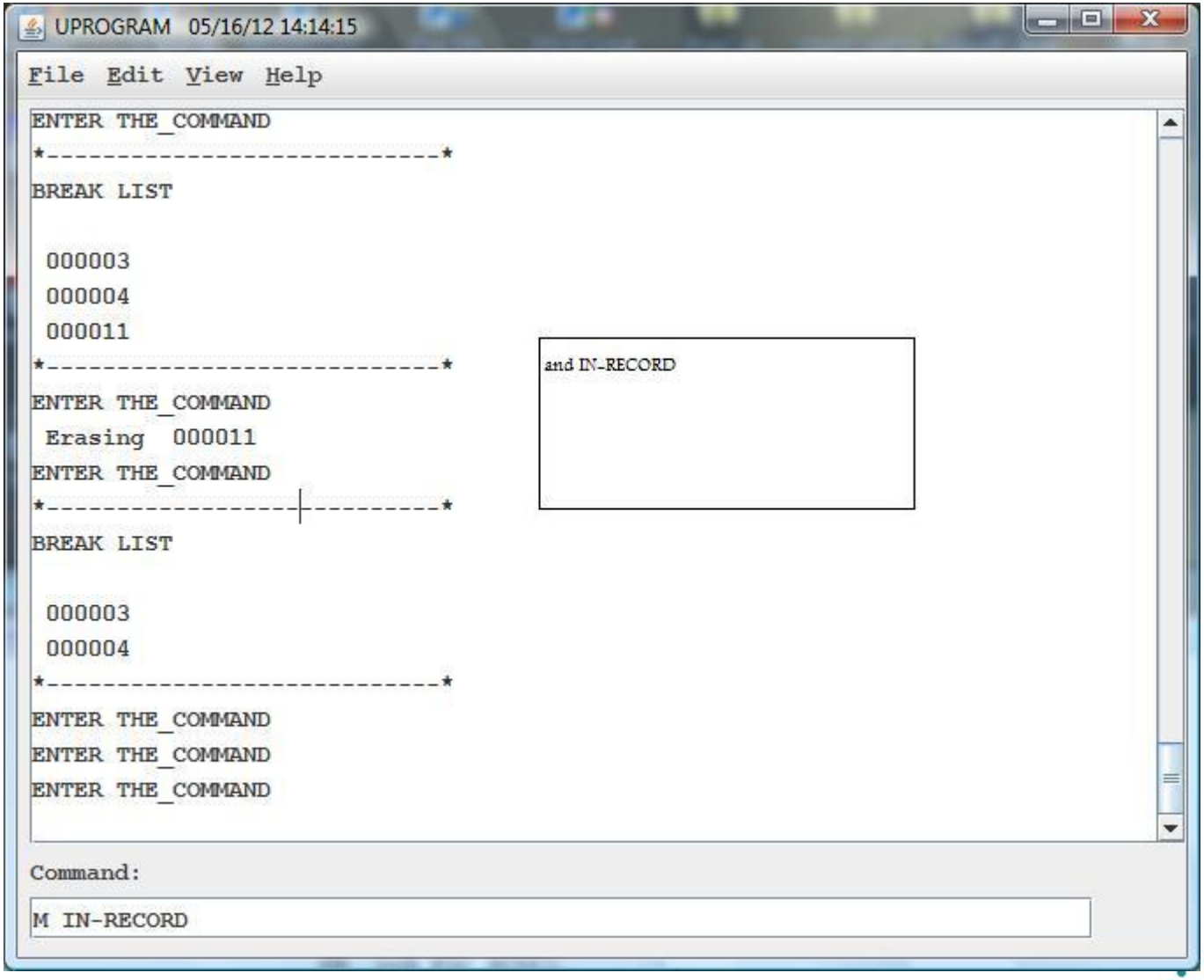

UPROGRAM 05/16/12 14:11:41
File Edit View Help
ENTER THE_COMMAND
ENTER THE_COMMAND

BREAK LIST
000003
000004
000011

ENTER THE_COMMAND
Erasing 000011
ENTER THE_COMMAND

BREAK LIST
000003
000004

ENTER THE_COMMAND
ENTER THE_COMMAND
Command:
M COUNT
We'll monitor COUNT, too

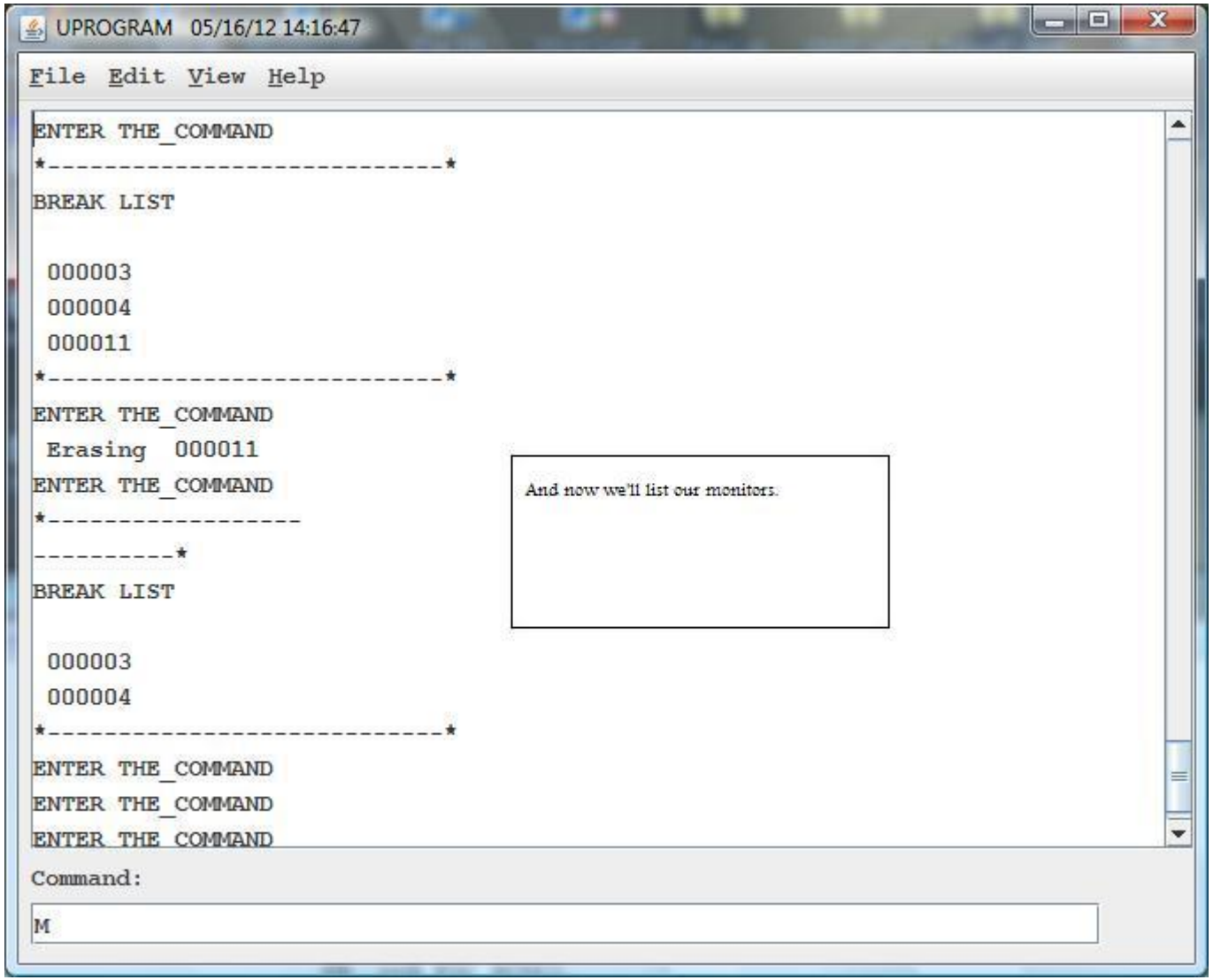

UPROGRAM 05/16/12 14:14:15
File Edit View Help
ENTER THE_COMMAND

BREAK LIST
000003
000004
000011

ENTER THE_COMMAND
Erasing 000011
ENTER THE_COMMAND

BREAK LIST
000003
000004

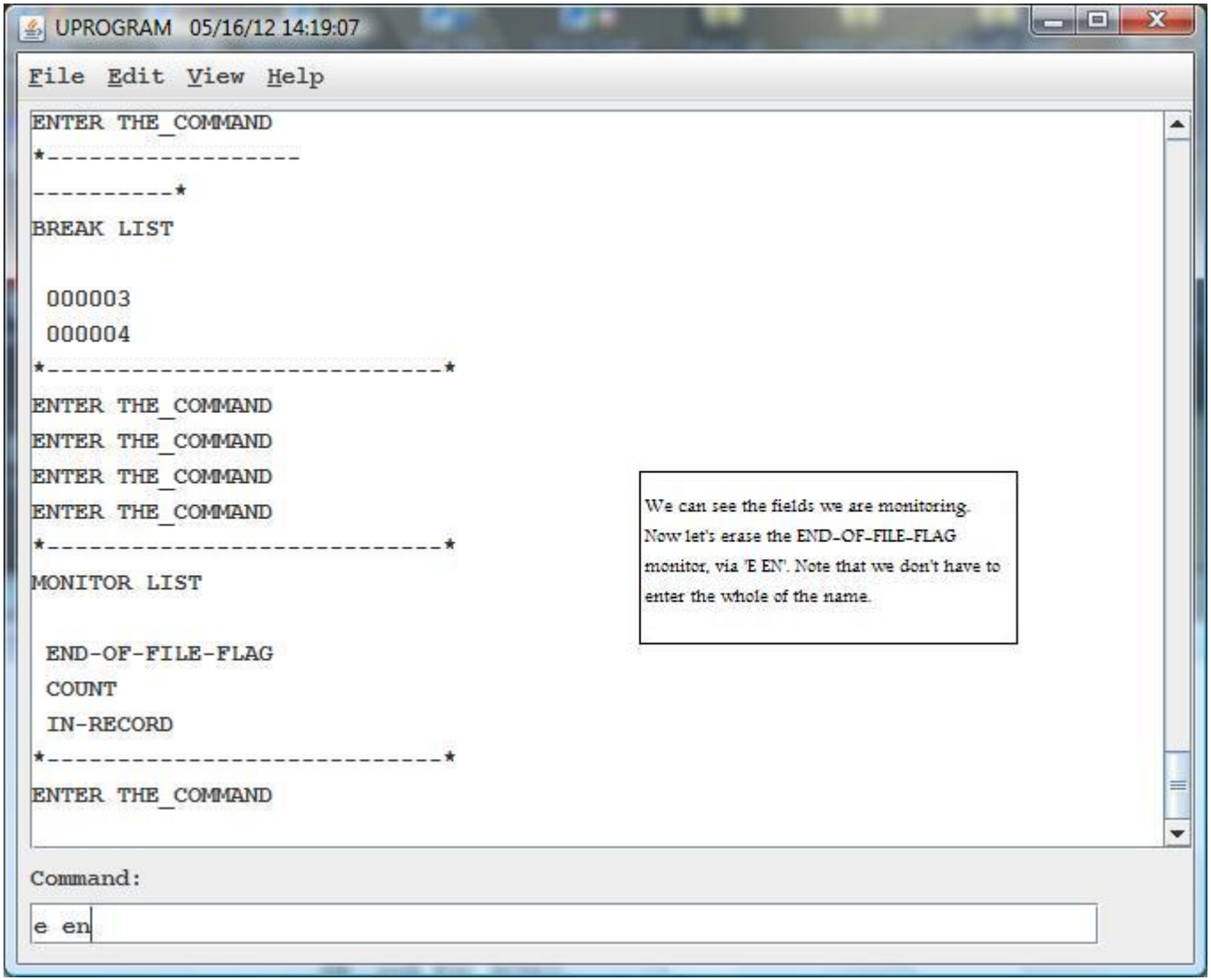
ENTER THE_COMMAND
ENTER THE_COMMAND
ENTER THE_COMMAND
and IN-RECORD
Command:
M IN-RECORD


UPROGRAM 05/16/12 14:16:47
File Edit View Help
ENTER THE_COMMAND

BREAK LIST
000003
000004
000011

ENTER THE_COMMAND
Erasing 000011
ENTER THE_COMMAND
*----------
----------*
BREAK LIST
000003
000004

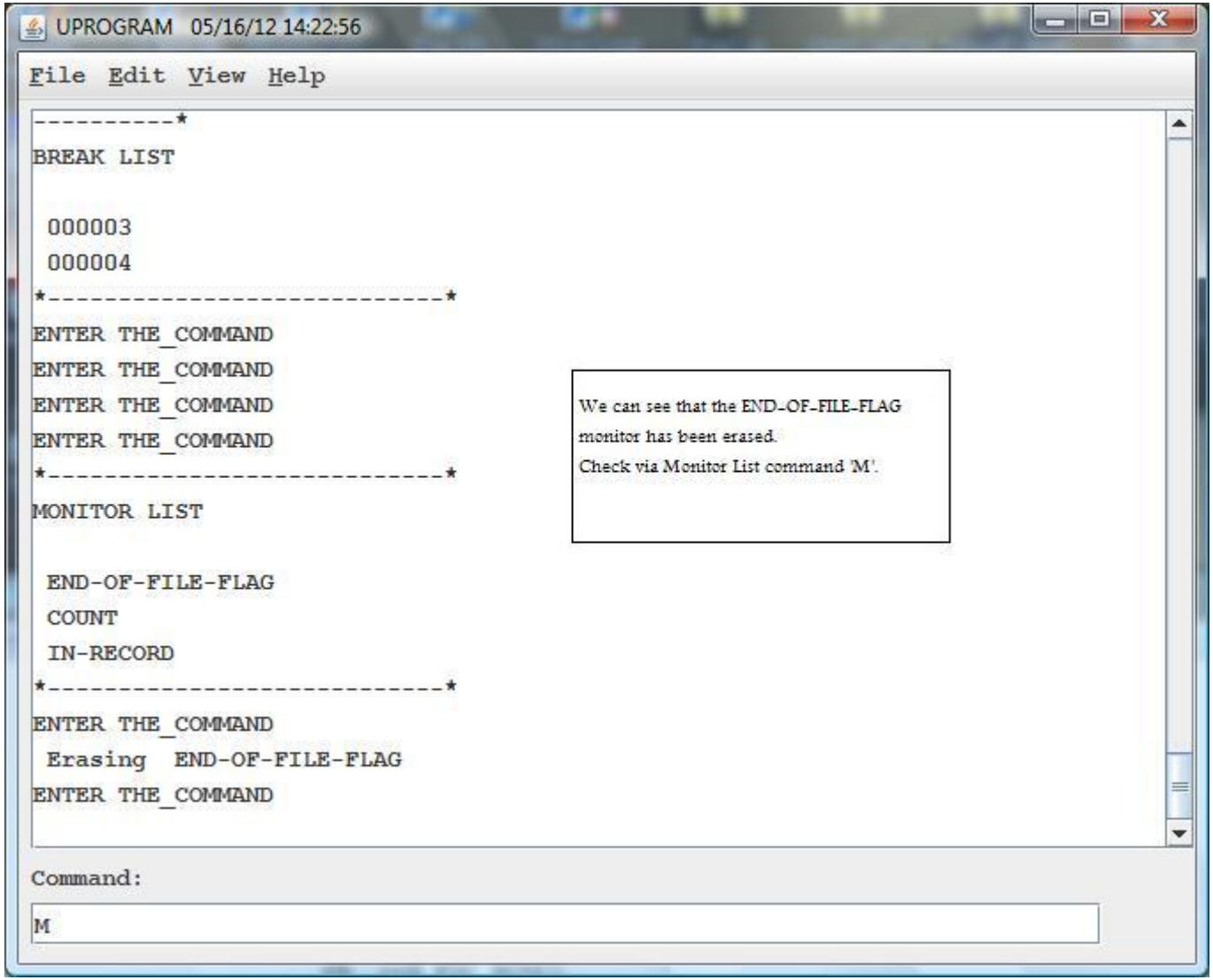
ENTER THE_COMMAND
ENTER THE_COMMAND
ENTER THE_COMMAND
Command:
M
And now we'll list our monitors.


UPROGRAM 05/16/12 14:19:07
File Edit View Help
ENTER THE_COMMAND
*------------------
----------*
BREAK LIST
000003
000004

ENTER THE_COMMAND
ENTER THE_COMMAND
ENTER THE_COMMAND
ENTER THE_COMMAND

MONITOR LIST
END-OF-FILE-FLAG
COUNT
IN-RECORD

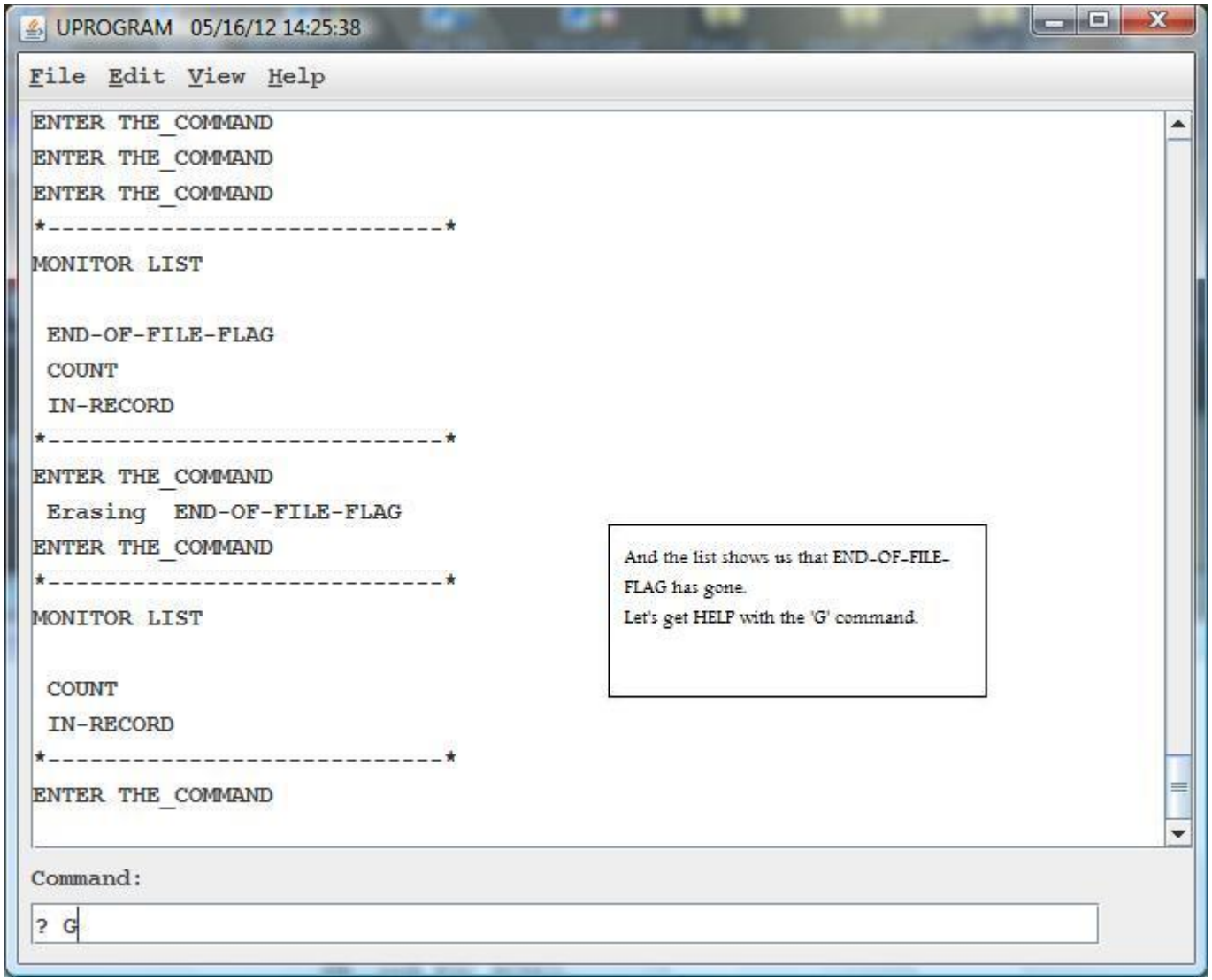
ENTER THE_COMMAND
Command:
e en
We can see the fields we are monitoring. Now let's erase the END-OF-FILE-FLAG monitor, via 'E EN'. Note that we don't have to enter the whole of the name.


UPROGRAM 05/16/12 14:22:56
File Edit View Help
----------*
BREAK LIST
000003
000004

ENTER THE_COMMAND
ENTER THE_COMMAND
ENTER THE_COMMAND
ENTER THE_COMMAND

MONITOR LIST
END-OF-FILE-FLAG
COUNT
IN-RECORD

ENTER THE_COMMAND
Erasing END-OF-FILE-FLAG
ENTER THE_COMMAND
We can see that the END-OF-FILE-FLAG monitor has been erased.
Check via Monitor List command 'M'.
Command:
M

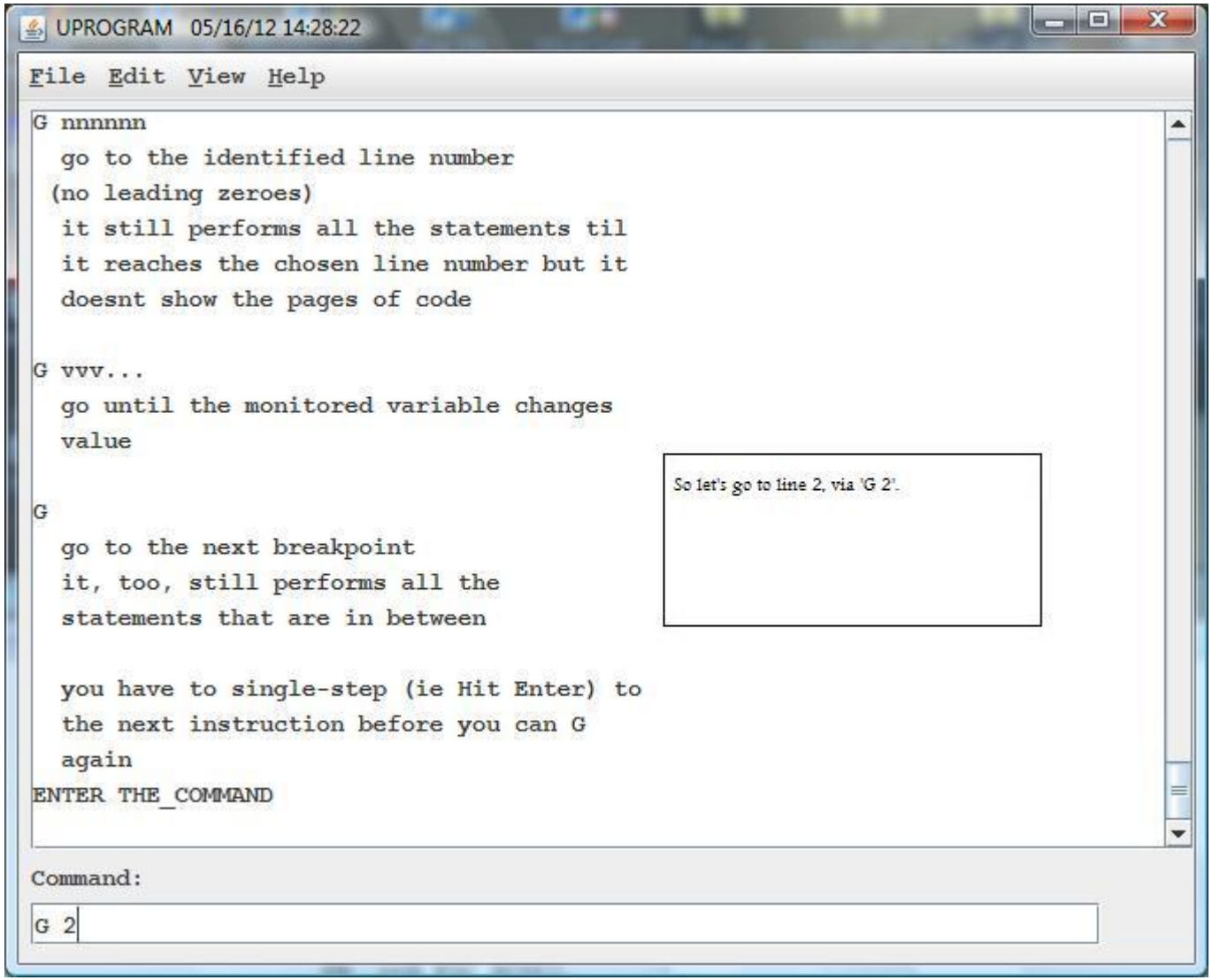

UPROGRAM 05/16/12 14:25:38
File Edit View Help
ENTER THE_COMMAND
ENTER THE_COMMAND
ENTER THE_COMMAND

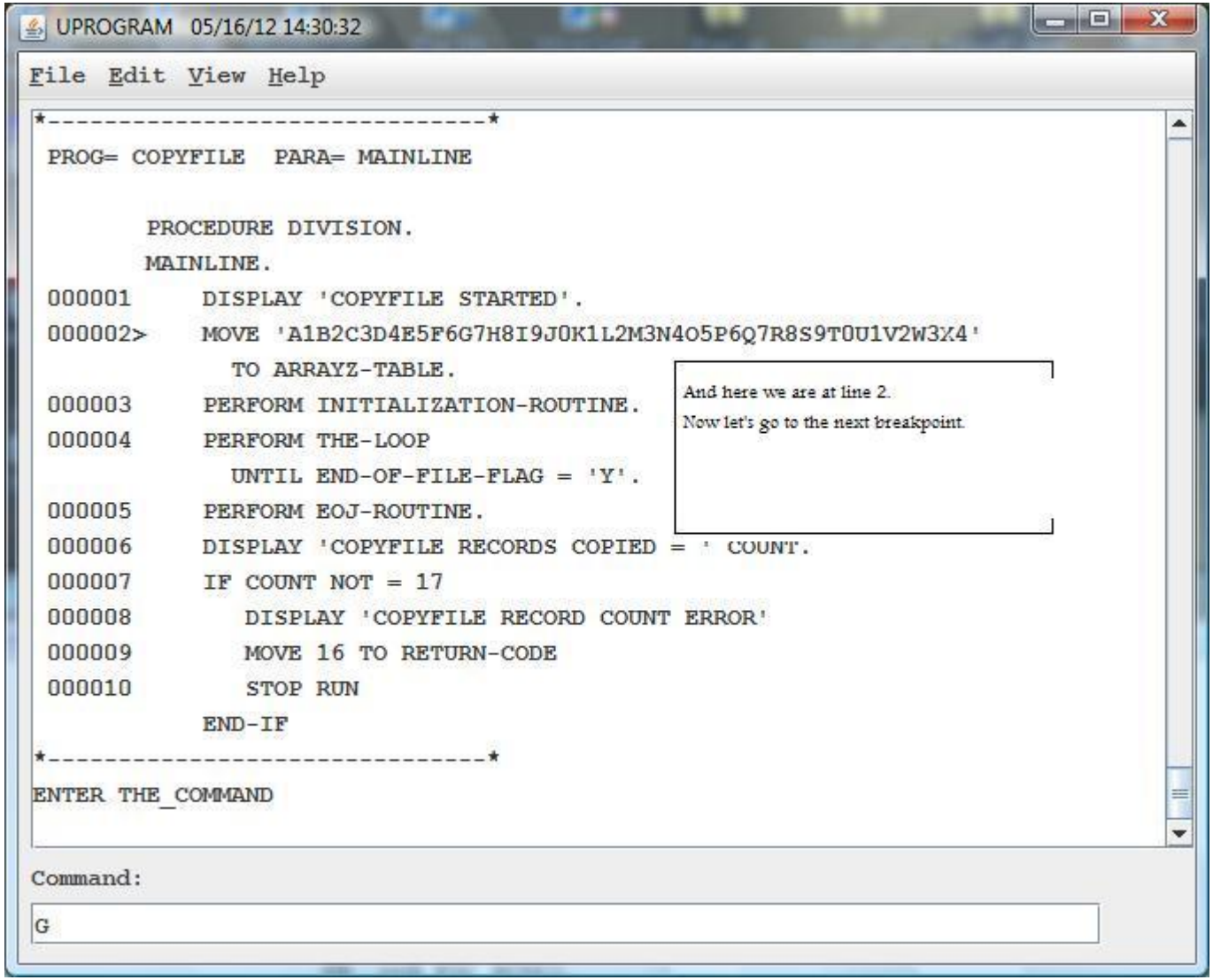
MONITOR LIST
END-OF-FILE-FLAG
COUNT
IN-RECORD

ENTER THE_COMMAND
Erasing END-OF-FILE-FLAG
ENTER THE_COMMAND

MONITOR LIST
COUNT
IN-RECORD

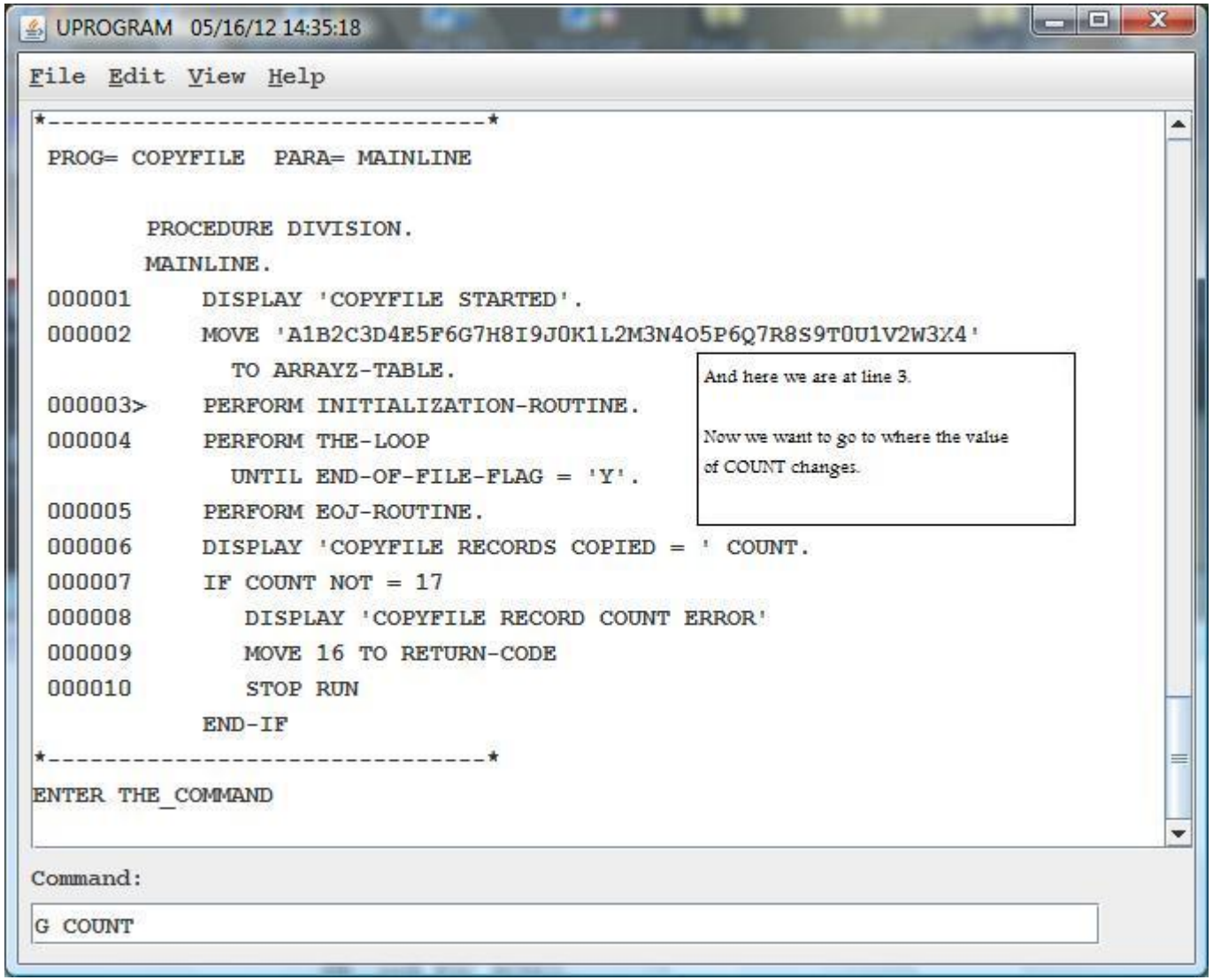
ENTER THE_COMMAND
Command:
? G
And the list shows us that END-OF-FILE-FLAG has gone.
Let's get HELP with the 'G' command.


UPROGRAM 05/16/12 14:28:22
File Edit View Help
G nnnnnn
go to the identified line number
(no leading zeroes)
it still performs all the statements til
it reaches the chosen line number but it
doesnt show the pages of code
G vvv...
go until the monitored variable changes
value
G
go to the next breakpoint
it, too, still performs all the
statements that are in between
you have to single-step (ie Hit Enter) to
the next instruction before you can G
again
ENTER THE_COMMAND
So let's go to line 2, via 'G 2'.
Command:
G 2


UPROGRAM 05/16/12 14:30:32
File Edit View Help

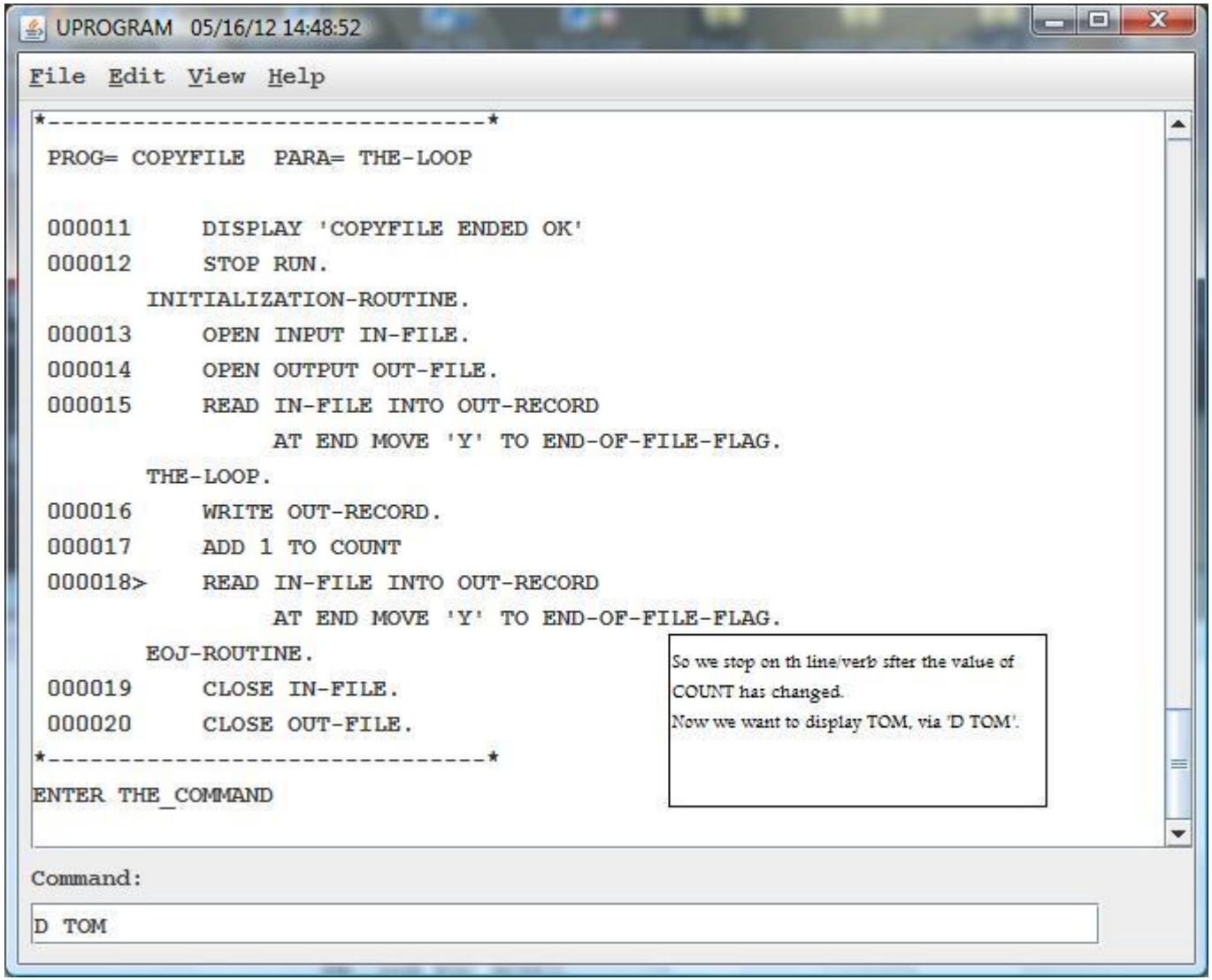
PROG= COPYFILE PARA= MAINLINE
PROCEDURE DIVISION.
MAINLINE.
000001 DISPLAY 'COPYFILE STARTED'.
000002> MOVE 'A1B2C3D4E5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X4'
TO ARRAYZ-TABLE.
000003 PERFORM INITIALIZATION-ROUTINE.
000004 PERFORM THE-LOOP
UNTIL END-OF-FILE-FLAG = 'Y'.
000005 PERFORM EOJ-ROUTINE.
000006 DISPLAY 'COPYFILE RECORDS COPIED = ' COUNT.
000007 IF COUNT NOT = 17
000008 DISPLAY 'COPYFILE RECORD COUNT ERROR'
000009 MOVE 16 TO RETURN-CODE
000010 STOP RUN
END-IF

ENTER THE_COMMAND
And here we are at line 2.
Now let's go to the next breakpoint.
Command:
G


UPROGRAM 05/16/12 14:35:18
File Edit View Help

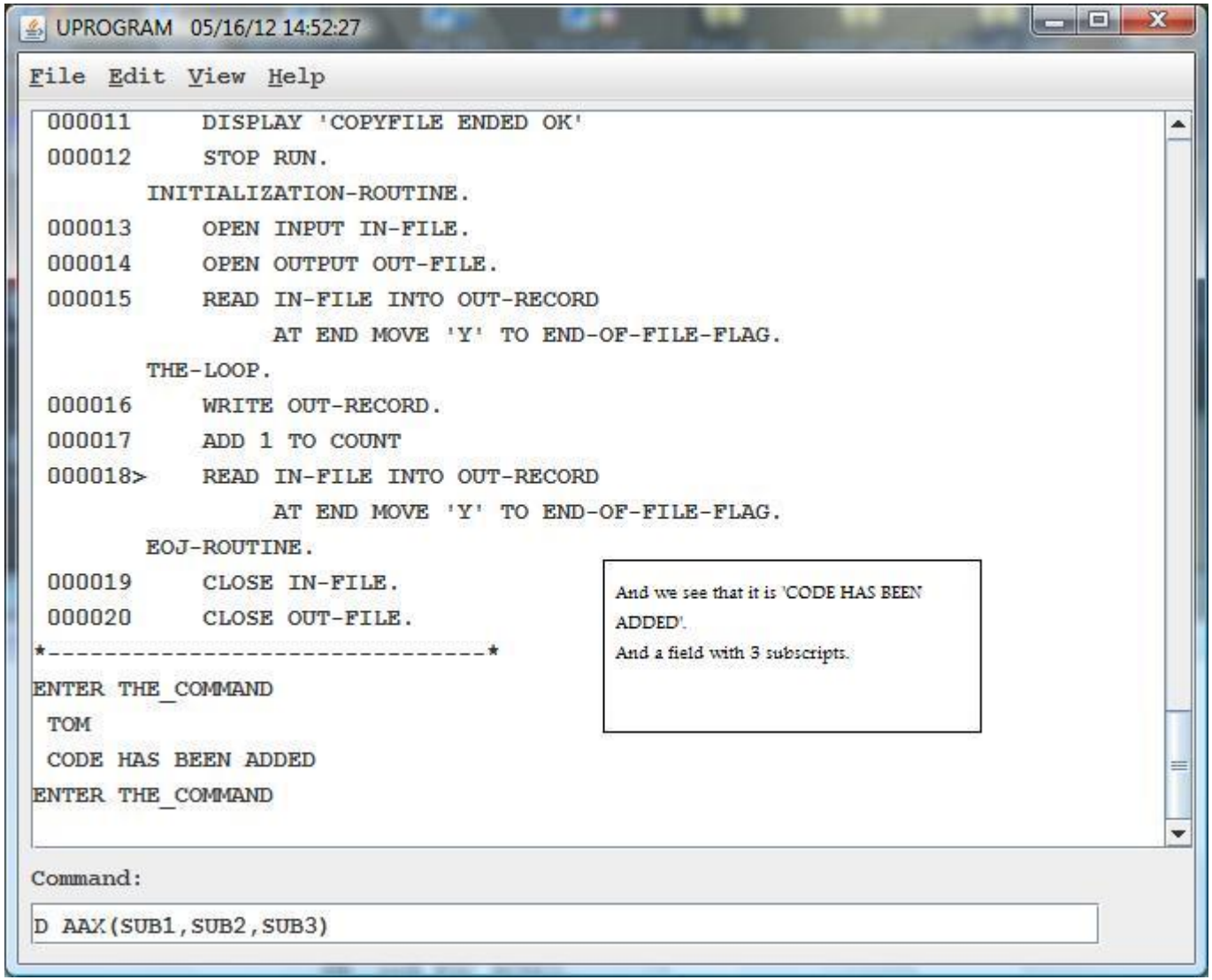
PROG= COPYFILE PARA= MAINLINE
PROCEDURE DIVISION.
MAINLINE.
000001 DISPLAY 'COPYFILE STARTED'.
000002 MOVE 'A1B2C3D4E5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T0U1V2W3X4'
TO ARRAYZ-TABLE.
000003> PERFORM INITIALIZATION-ROUTINE.
000004 PERFORM THE-LOOP
UNTIL END-OF-FILE-FLAG = 'Y'.
000005 PERFORM EOJ-ROUTINE.
000006 DISPLAY 'COPYFILE RECORDS COPIED = ' COUNT.
000007 IF COUNT NOT = 17
000008 DISPLAY 'COPYFILE RECORD COUNT ERROR'
000009 MOVE 16 TO RETURN-CODE
000010 STOP RUN
END-IF

ENTER THE_COMMAND
And here we are at line 3.
Now we want to go to where the value of COUNT changes.
Command:
G COUNT

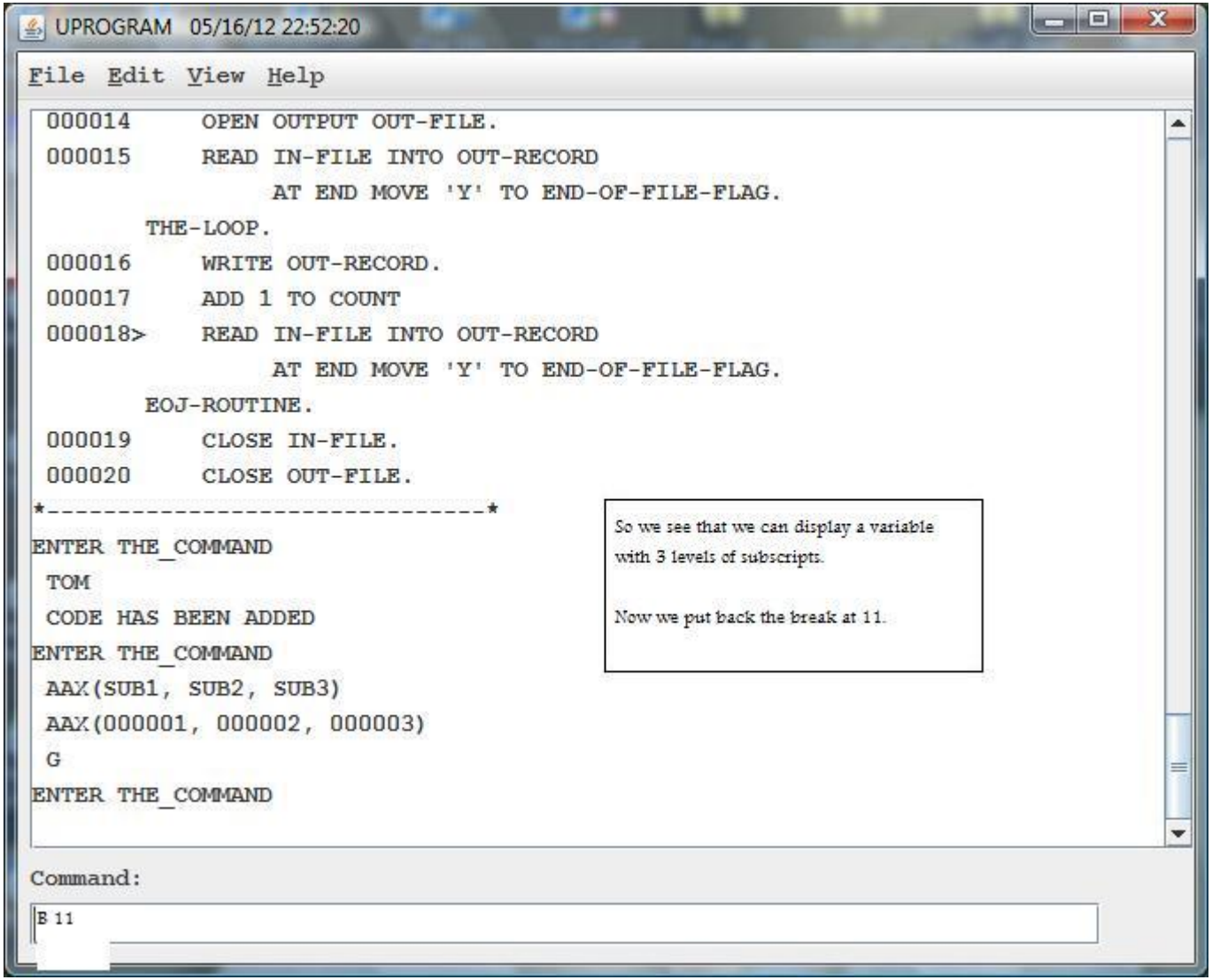

UPROGRAM 05/16/12 14:48:52
File Edit View Help

PROG= COPYFILE PARA= THE-LOOP
000011 DISPLAY 'COPYFILE ENDED OK'
000012 STOP RUN.
INITIALIZATION-ROUTINE.
000013 OPEN INPUT IN-FILE.
000014 OPEN OUTPUT OUT-FILE.
000015 READ IN-FILE INTO OUT-RECORD
AT END MOVE 'Y' TO END-OF-FILE-FLAG.
THE-LOOP.
000016 WRITE OUT-RECORD.
000017 ADD 1 TO COUNT
000018> READ IN-FILE INTO OUT-RECORD
AT END MOVE 'Y' TO END-OF-FILE-FLAG.
EOJ-ROUTINE.
000019 CLOSE IN-FILE.
000020 CLOSE OUT-FILE.

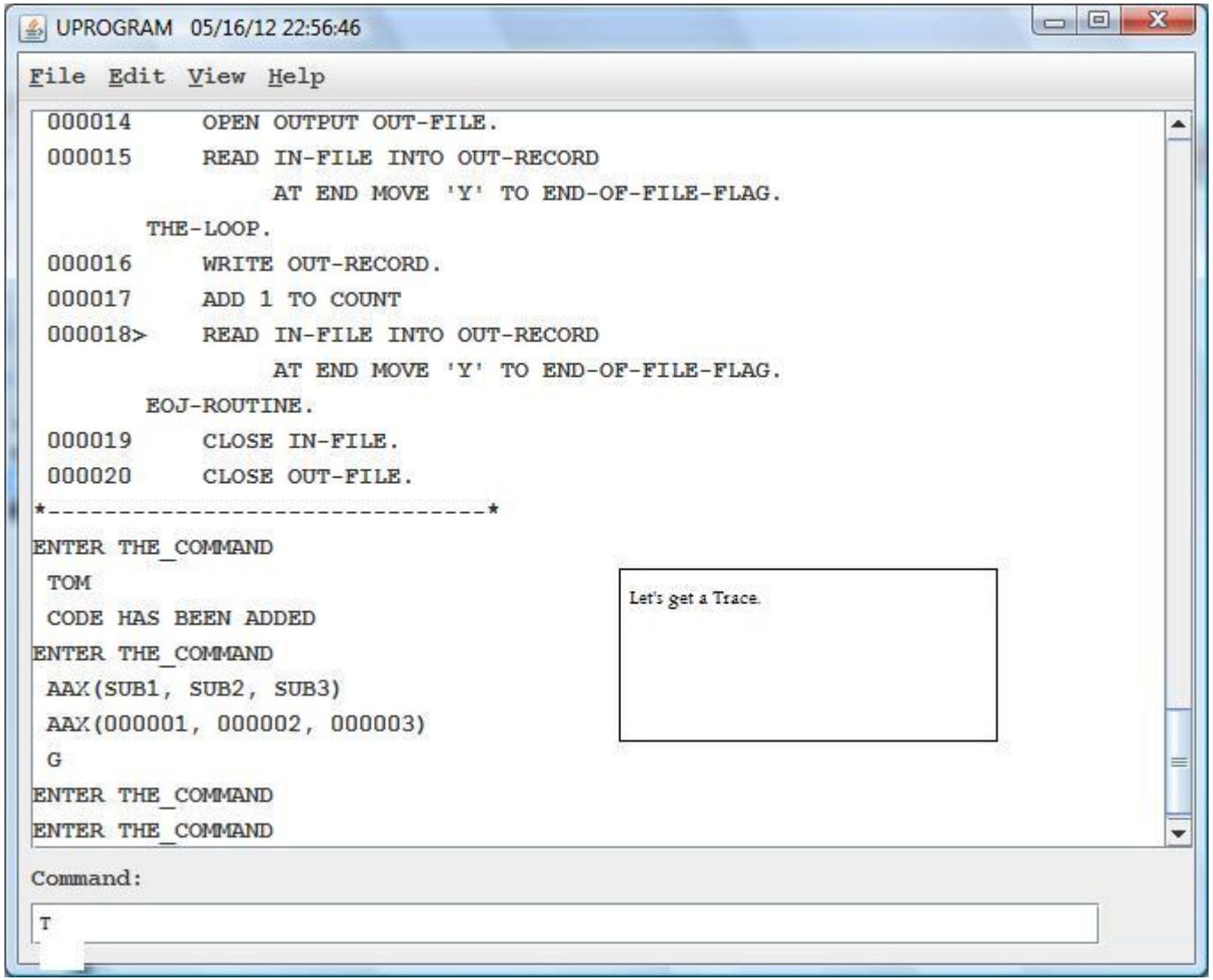
ENTER THE_COMMAND
So we stop on th line/verb after the value of COUNT has changed.
Now we want to display TOM, via 'D TOM'.
Command:
D TOM


UPROGRAM 05/16/12 14:52:27
File Edit View Help
000011 DISPLAY 'COPYFILE ENDED OK'
000012 STOP RUN.
INITIALIZATION-ROUTINE.
000013 OPEN INPUT IN-FILE.
000014 OPEN OUTPUT OUT-FILE.
000015 READ IN-FILE INTO OUT-RECORD
AT END MOVE 'Y' TO END-OF-FILE-FLAG.
THE-LOOP.
000016 WRITE OUT-RECORD.
000017 ADD 1 TO COUNT
000018> READ IN-FILE INTO OUT-RECORD
AT END MOVE 'Y' TO END-OF-FILE-FLAG.
EOJ-ROUTINE.
000019 CLOSE IN-FILE.
000020 CLOSE OUT-FILE.

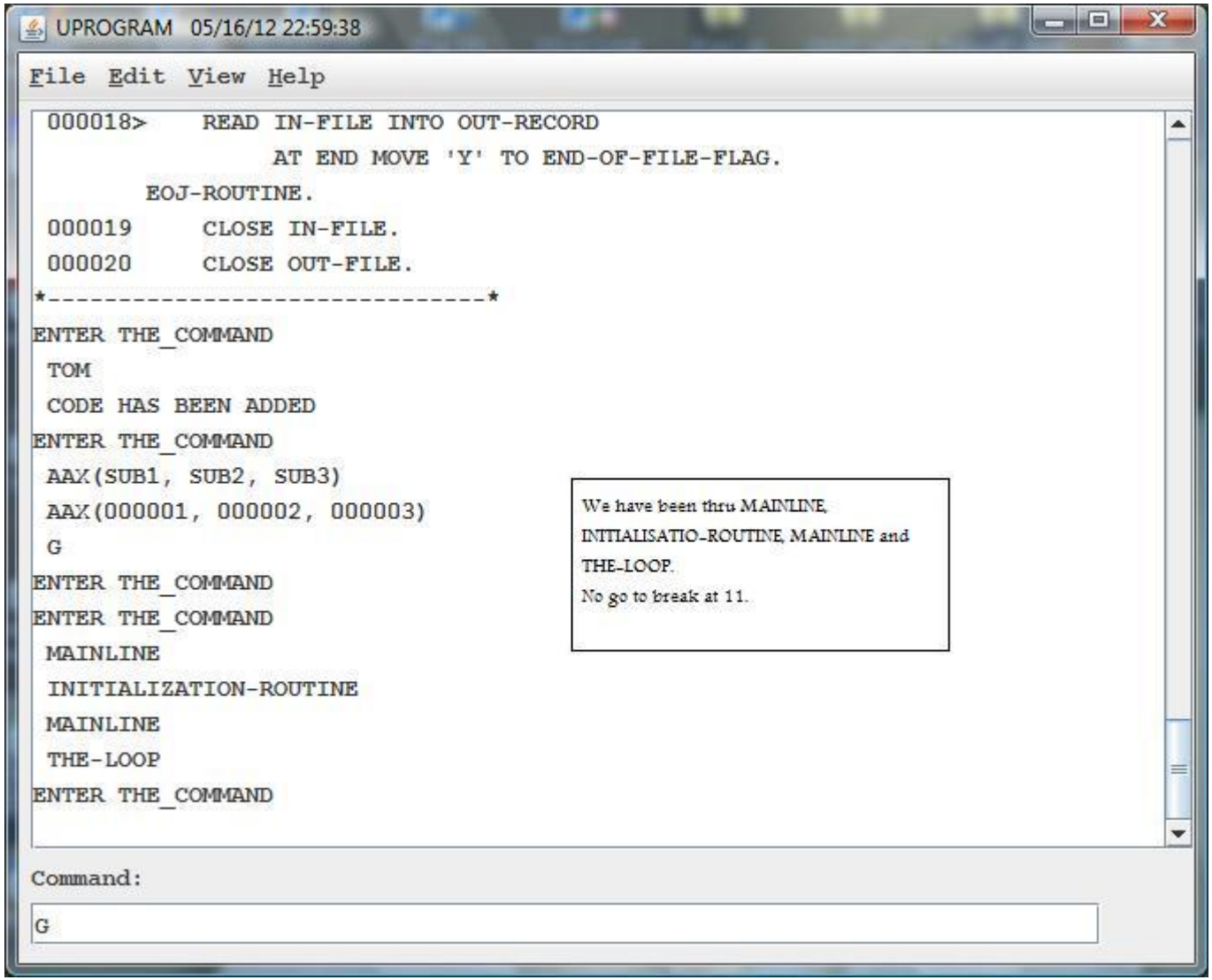
ENTER THE_COMMAND
TOM
CODE HAS BEEN ADDED
ENTER THE_COMMAND
And we see that it is 'CODE HAS BEEN ADDED'.
And a field with 3 subscripts.
Command:
D AAX(SUB1,SUB2,SUB3)


UPROGRAM 05/16/12 22:52:20
File Edit View Help
000014 OPEN OUTPUT OUT-FILE.
000015 READ IN-FILE INTO OUT-RECORD
AT END MOVE 'Y' TO END-OF-FILE-FLAG.
THE-LOOP.
000016 WRITE OUT-RECORD.
000017 ADD 1 TO COUNT
000018> READ IN-FILE INTO OUT-RECORD
AT END MOVE 'Y' TO END-OF-FILE-FLAG.
EOJ-ROUTINE.
000019 CLOSE IN-FILE.
000020 CLOSE OUT-FILE.

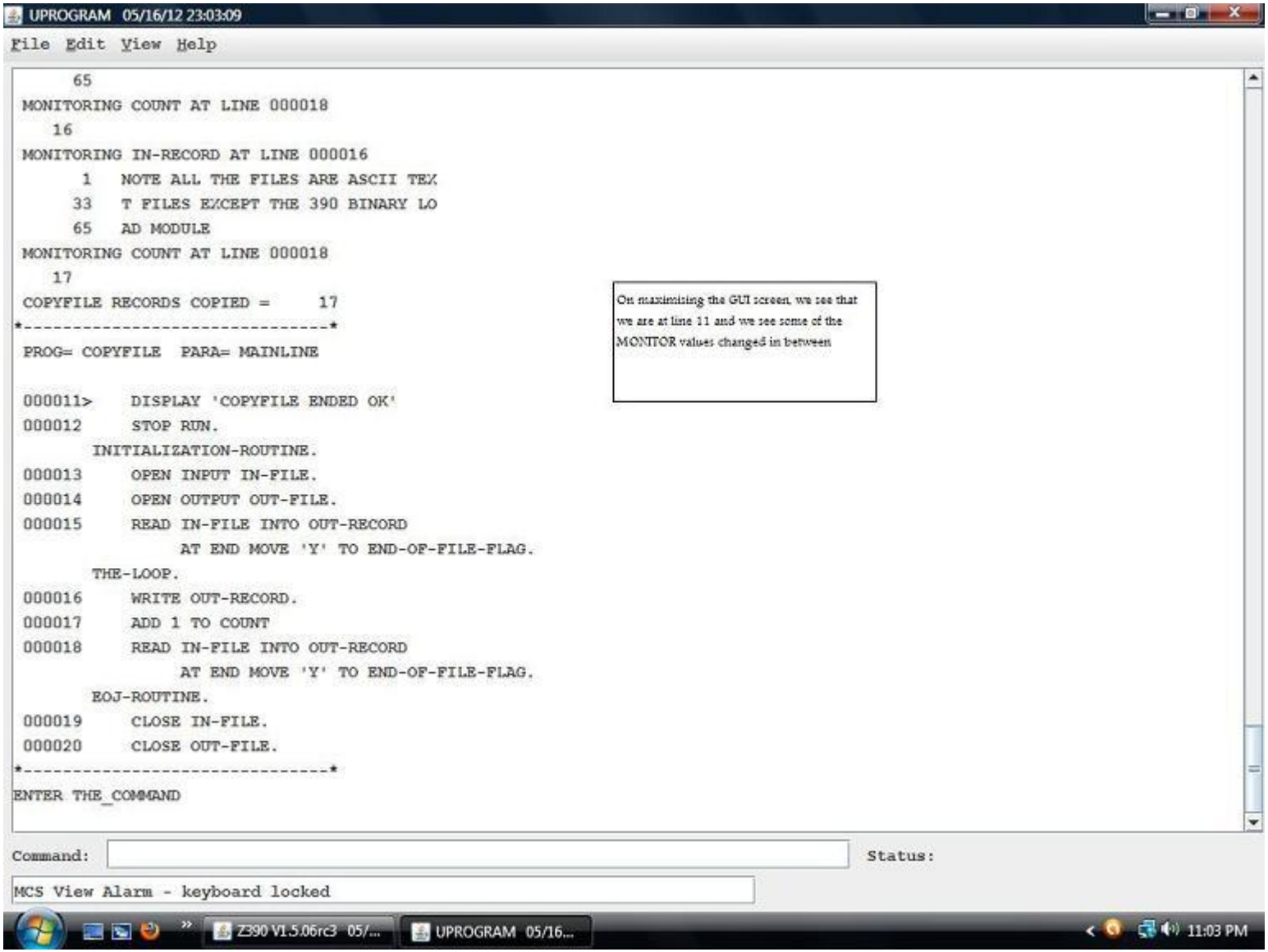
ENTER THE_COMMAND
TOM
CODE HAS BEEN ADDED
ENTER THE_COMMAND
AAX(SUB1, SUB2, SUB3)
AAX(000001, 000002, 000003)
G
ENTER THE_COMMAND
Command:
B 11
So we see that we can display a variable with 3 levels of subscripts.
Now we put back the break at 11.


UPROGRAM 05/16/12 22:56:46
File Edit View Help
000014 OPEN OUTPUT OUT-FILE.
000015 READ IN-FILE INTO OUT-RECORD
AT END MOVE 'Y' TO END-OF-FILE-FLAG.
THE-LOOP.
000016 WRITE OUT-RECORD.
000017 ADD 1 TO COUNT
000018> READ IN-FILE INTO OUT-RECORD
AT END MOVE 'Y' TO END-OF-FILE-FLAG.
EOJ-ROUTINE.
000019 CLOSE IN-FILE.
000020 CLOSE OUT-FILE.

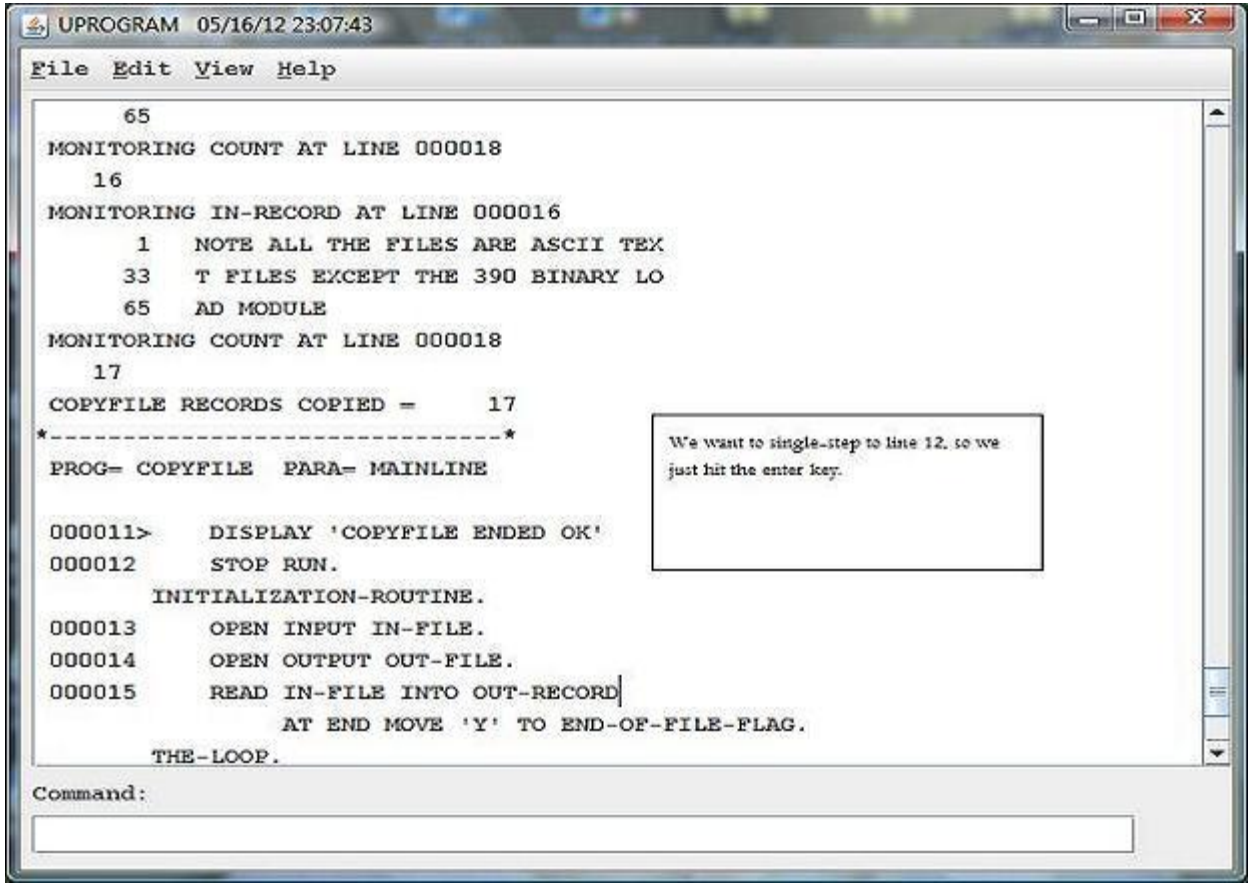
ENTER THE_COMMAND
TOM
CODE HAS BEEN ADDED
ENTER THE_COMMAND
AAX(SUB1, SUB2, SUB3)
AAX(000001, 000002, 000003)
G
ENTER THE_COMMAND
ENTER THE_COMMAND
Let's get a Trace.
Command:
T


UPROGRAM 05/16/12 22:59:38
File Edit View Help
000018> READ IN-FILE INTO OUT-RECORD
AT END MOVE 'Y' TO END-OF-FILE-FLAG.
EOJ-ROUTINE.
000019 CLOSE IN-FILE.
000020 CLOSE OUT-FILE.

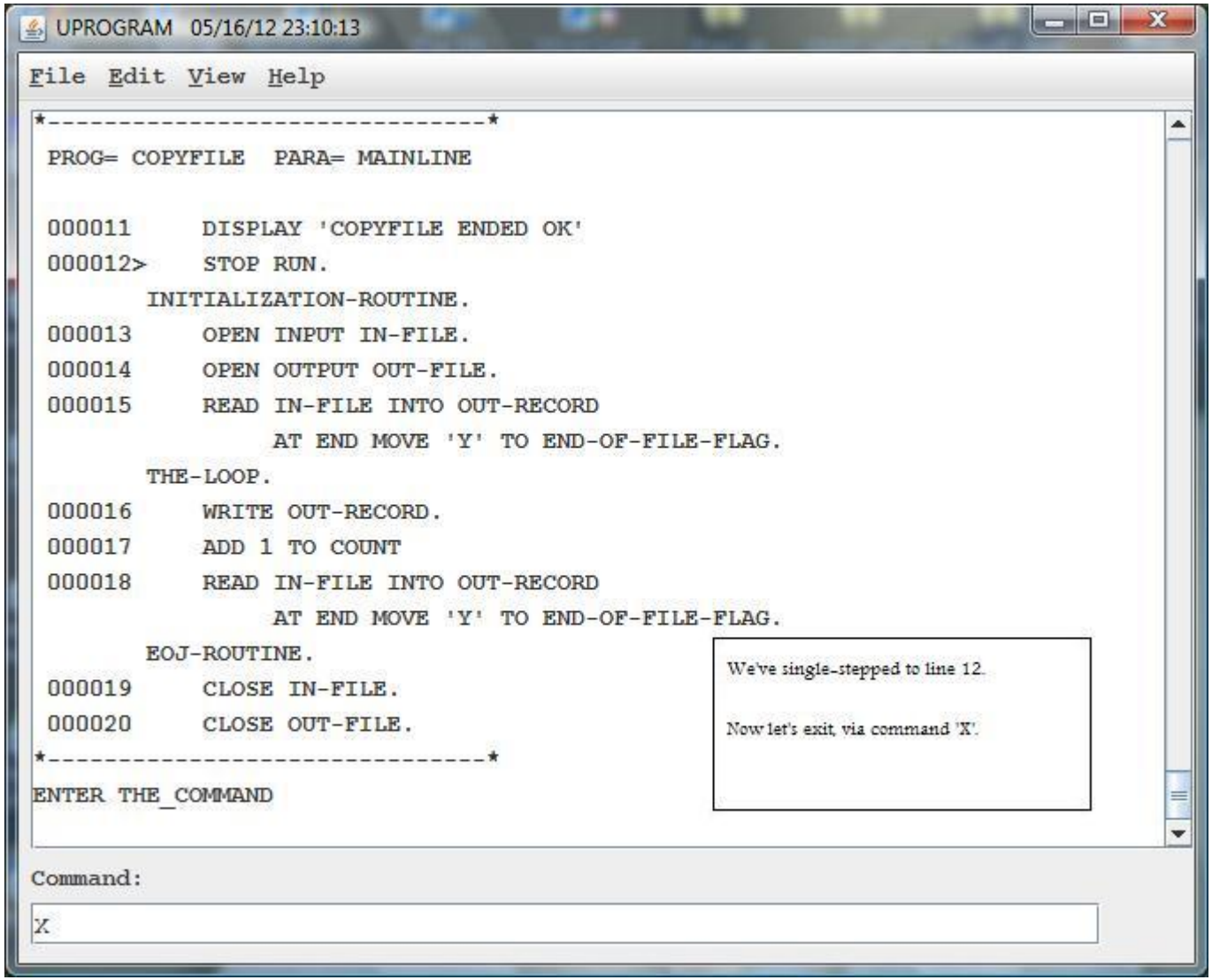
ENTER THE_COMMAND
TOM
CODE HAS BEEN ADDED
ENTER THE_COMMAND
AAX(SUB1, SUB2, SUB3)
AAX(000001, 000002, 000003)
G
ENTER THE_COMMAND
ENTER THE_COMMAND
MAINLINE
INITIALIZATION-ROUTINE
MAINLINE
THE-LOOP
ENTER THE_COMMAND
We have been thru MAINLINE, INITIALISATIO-ROUTINE, MAINLINE and THE-LOOP.
No go to break at 11.
Command:
G



On maximizing the GUI screen, we see that we are at line 11 and we see some of the MONITOR values changed in between

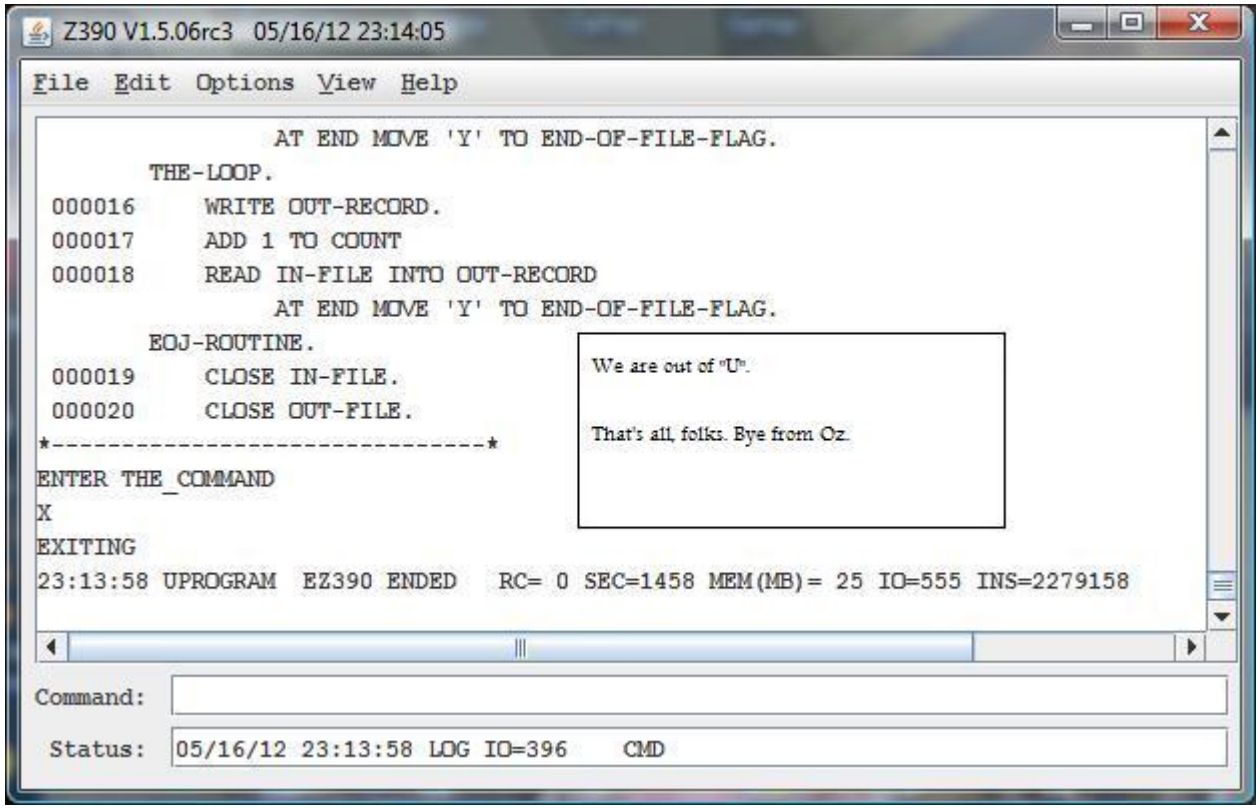

UPROGRAM 05/16/12 23:07:43
File Edit View Help
65
MONITORING COUNT AT LINE 000018
16
MONITORING IN-RECORD AT LINE 000016
1 NOTE ALL THE FILES ARE ASCII TEX
33 T FILES EXCEPT THE 390 BINARY LO
65 AD MODULE
MONITORING COUNT AT LINE 000018
17
COPYFILE RECORDS COPIED = 17

PROG= COPYFILE PARA= MAINLINE
000011> DISPLAY 'COPYFILE ENDED OK'
000012 STOP RUN.
INITIALIZATION-ROUTINE.
000013 OPEN INPUT IN-FILE.
000014 OPEN OUTPUT OUT-FILE.
000015 READ IN-FILE INTO OUT-RECORD
AT END MOVE 'Y' TO END-OF-FILE-FLAG.
THE-LOOP.
Command:
We want to single-step to line 12, so we just hit the enter key.


UPROGRAM 05/16/12 23:10:13
File Edit View Help

PROG= COPYFILE PARA= MAINLINE
000011 DISPLAY 'COPYFILE ENDED OK'
000012> STOP RUN.
INITIALIZATION-ROUTINE.
000013 OPEN INPUT IN-FILE.
000014 OPEN OUTPUT OUT-FILE.
000015 READ IN-FILE INTO OUT-RECORD
AT END MOVE 'Y' TO END-OF-FILE-FLAG.
THE-LOOP.
000016 WRITE OUT-RECORD.
000017 ADD 1 TO COUNT
000018 READ IN-FILE INTO OUT-RECORD
AT END MOVE 'Y' TO END-OF-FILE-FLAG.
EOJ-ROUTINE.
000019 CLOSE IN-FILE.
000020 CLOSE OUT-FILE.

ENTER THE_COMMAND
We've single-stepped to line 12.
Now let's exit, via command 'X'.
Command:
X


Z390 V1.5.06rc3 05/16/12 23:14:05
File Edit Options View Help
AT END MOVE 'Y' TO END-OF-FILE-FLAG.
THE-LOOP.
000016 WRITE OUT-RECORD.
000017 ADD 1 TO COUNT
000018 READ IN-FILE INTO OUT-RECORD
AT END MOVE 'Y' TO END-OF-FILE-FLAG.
EOJ-ROUTINE.
000019 CLOSE IN-FILE.
000020 CLOSE OUT-FILE.

ENTER THE_COMMAND
X
EXITING
23:13:58 UPROGRAM EZ390 ENDED RC= 0 SEC=1458 MEM(MB)= 25 IO=555 INS=2279158
We are out of "U".
That's all, folks. Bye from Oz.
Command:
Status: 05/16/12 23:13:58 LOG IO=396 CMD

zCICS

zCICS

- Developed by Melvyn Maltz
- Supports CICS command level COBOL and HLASM programs
- Supports remote TN3270 access
- Provides SOA platform
- Emulates most CICS functionality
- Time constraints prohibit a full discussion in this session
- Please see the handout for SHARE Session 9280 (Summer 2011, Orlando) for more information

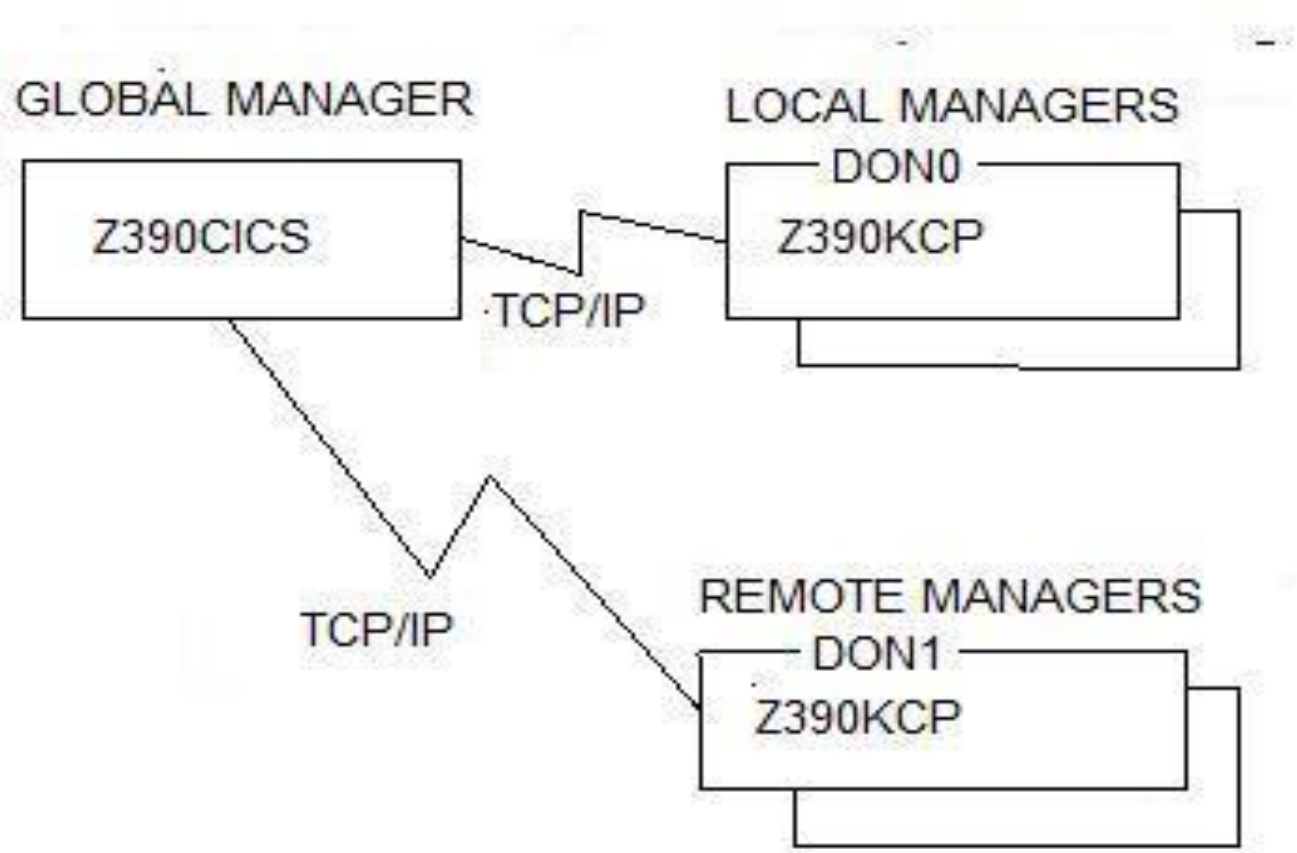
zCICS
What's new

- V11 released with z390 v1.5.0.6
- Full support for containers and channels

zCICS Overview

- The following slides were taken from the Session 9280 presentation
- That's why the formatting is different
- Not all slides are presented, see the 9280 slides for the full package
- This is just a quick overview of zCICS

zCICS Overview



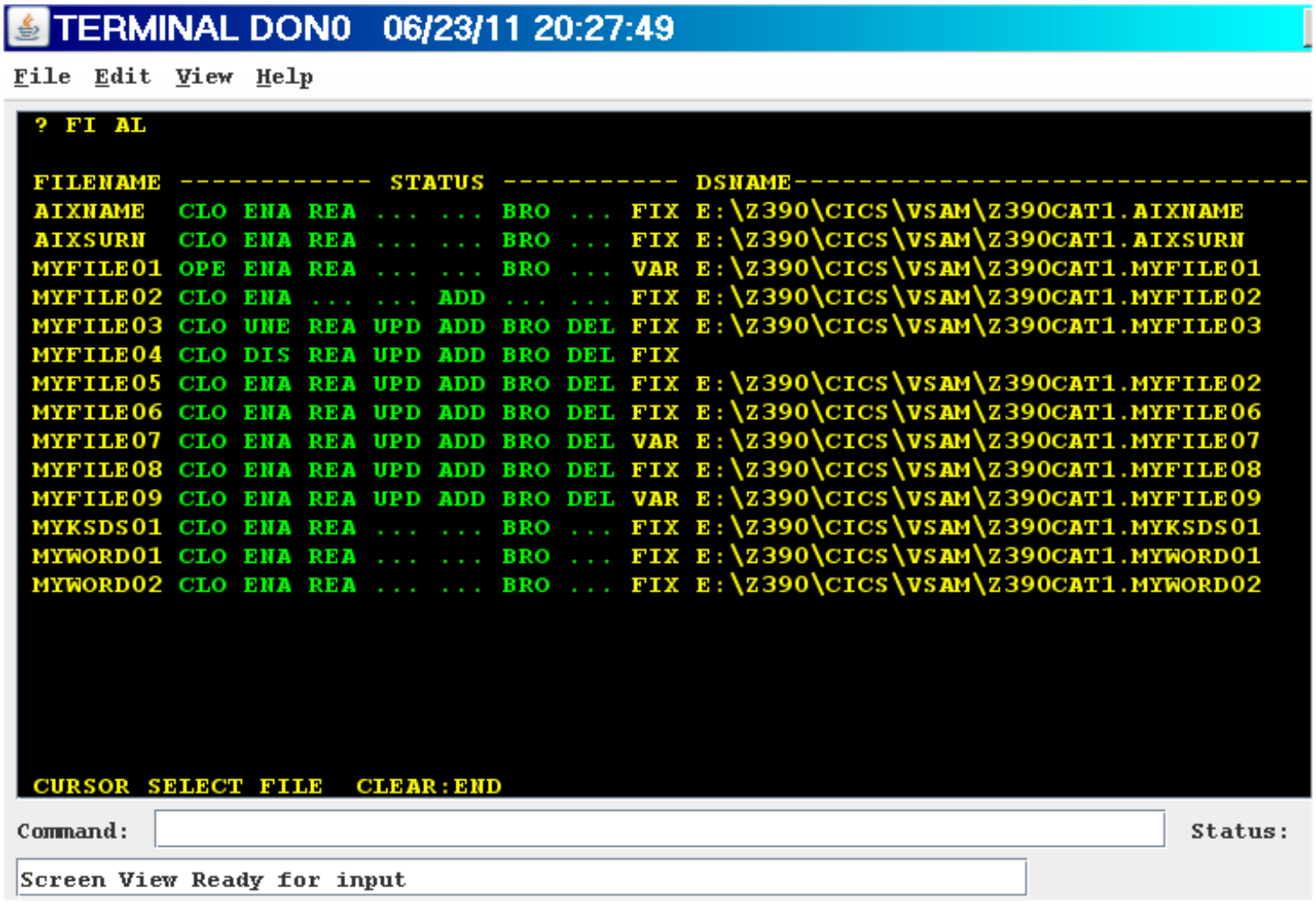
zCICS Supplied Transactions

- **Many test transactions**

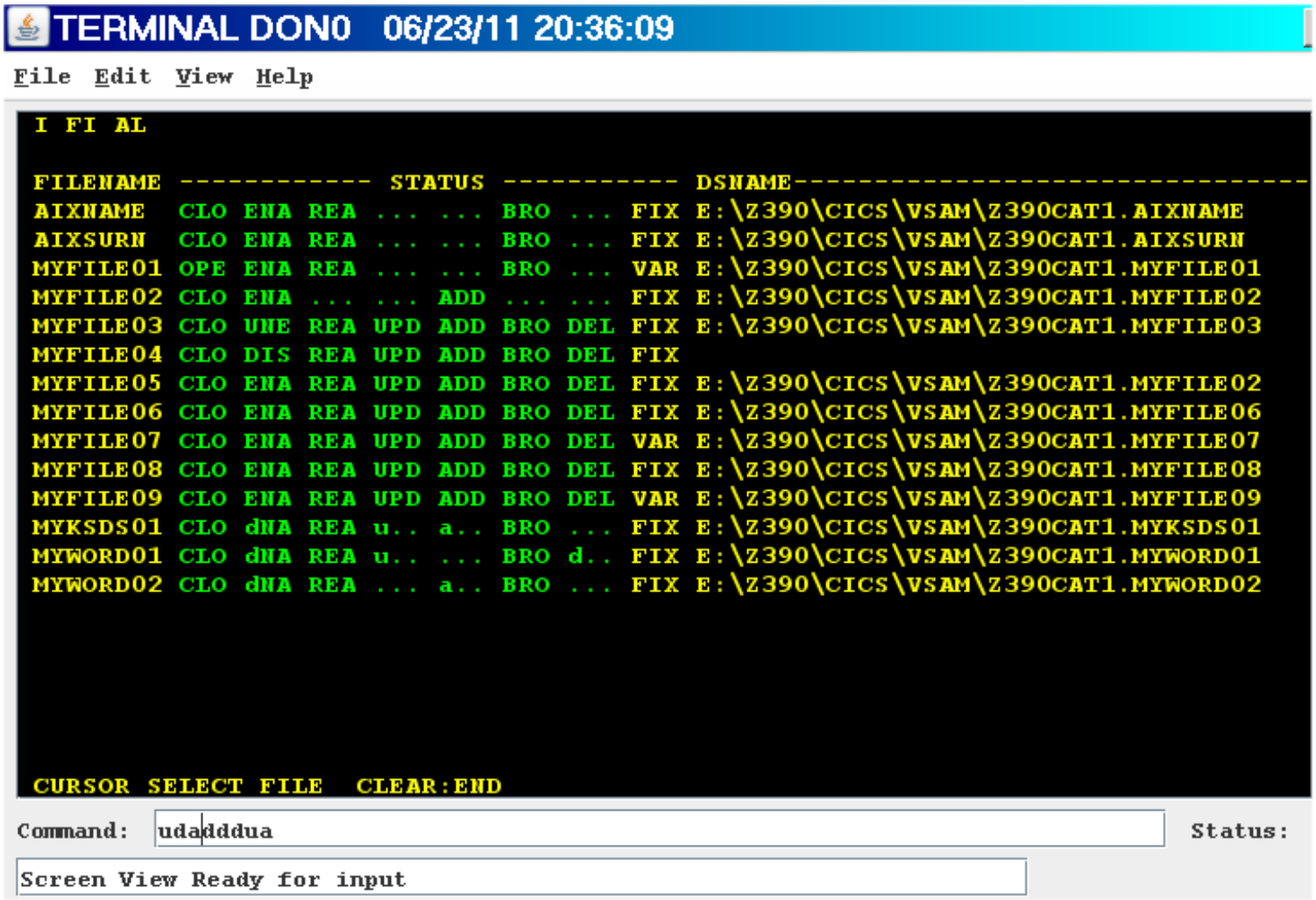
- **CEMT I TERm** **CEMT S TER OUT**
- **CEMT I TRAn** **CEMT P SHU**
- **CEMT I FILe** **CEMT P SHU IMM**
- **CEMT S FILe**
- **CEMT I SYStem**
- **CEMT I ENQueue**

- **CEBR**
- **CEDF**

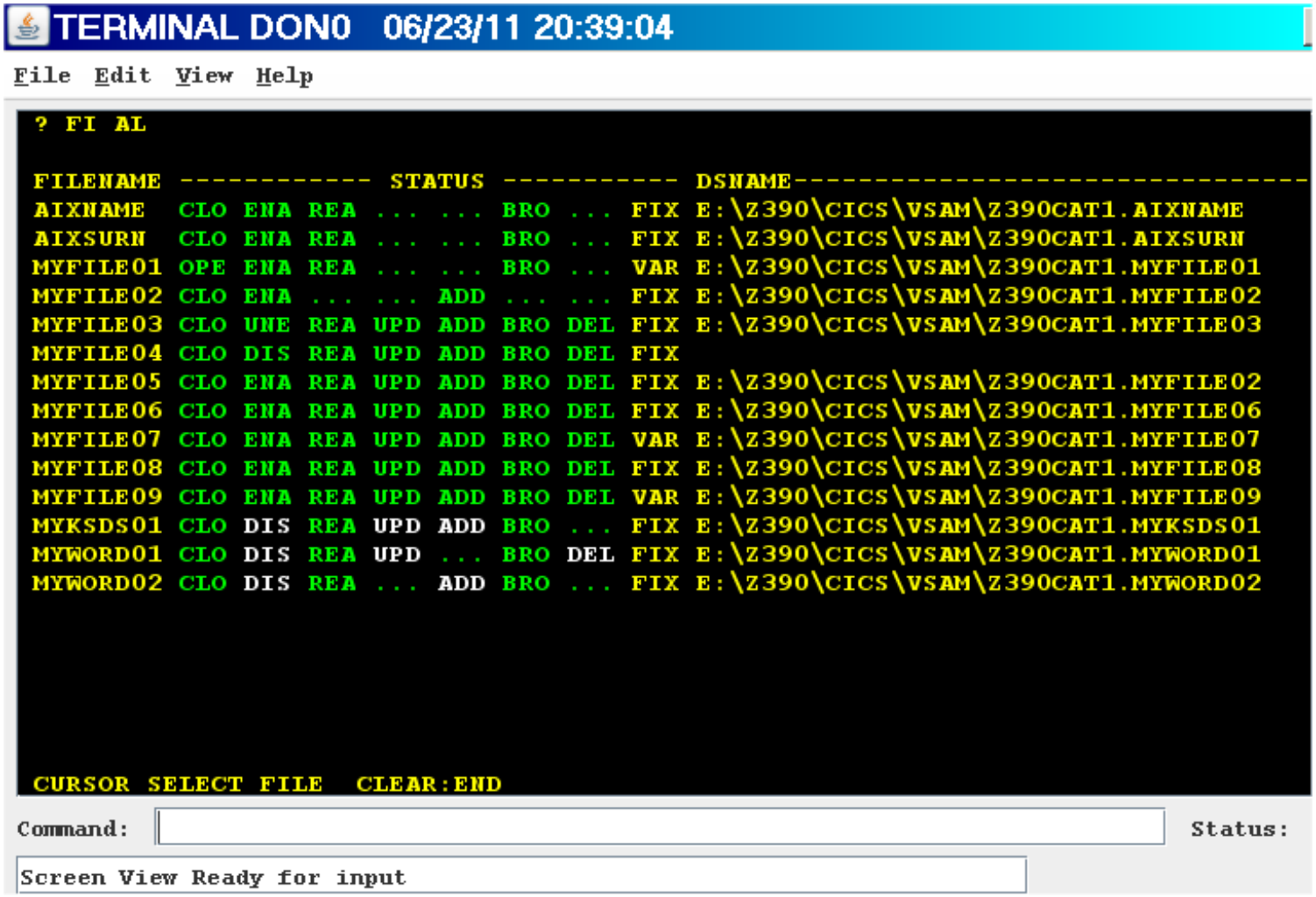
zCICS CEMT INQUIRE/SET FILE 1



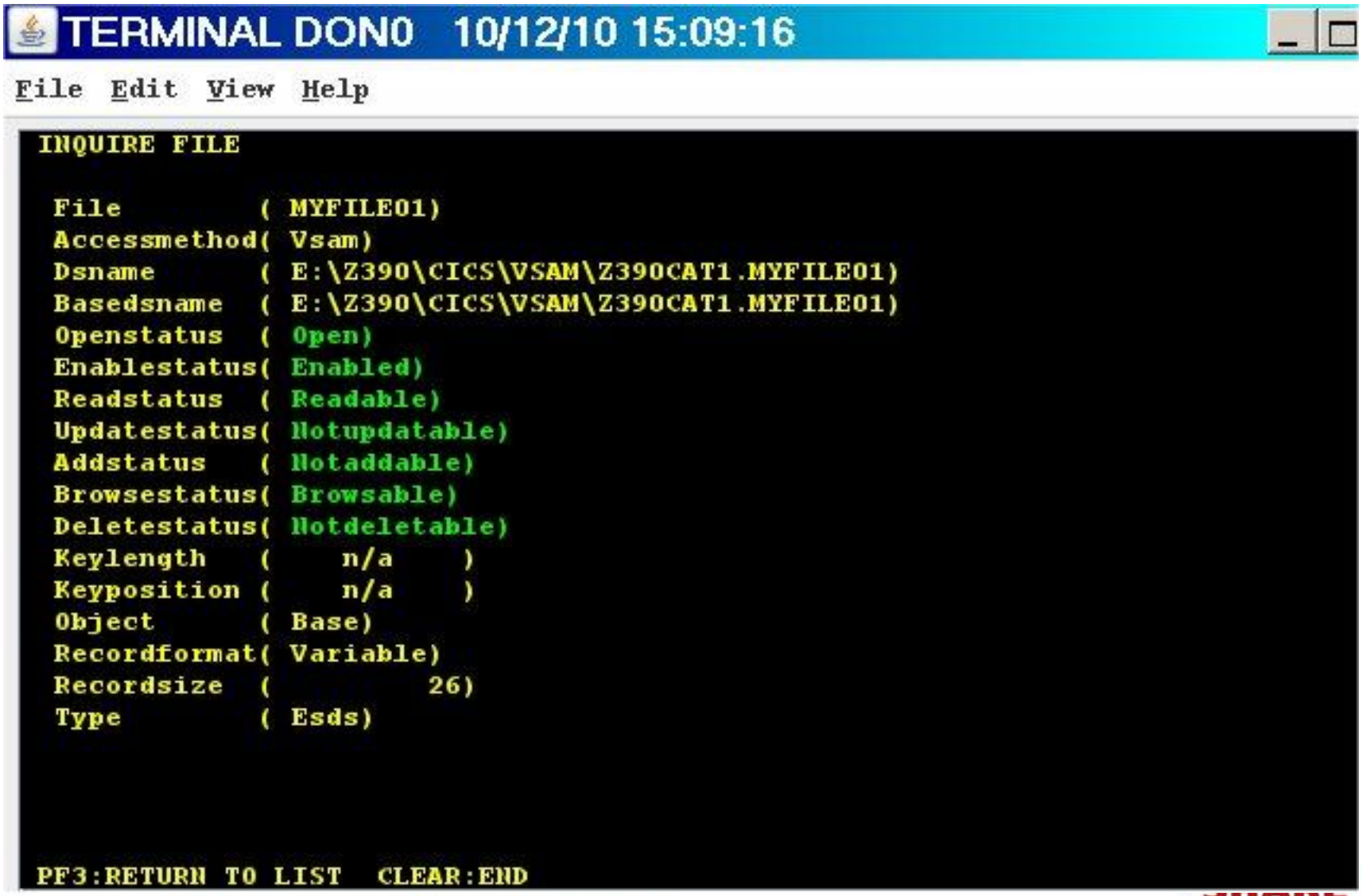
zCICS CEMT INQUIRE/SET FILE 2



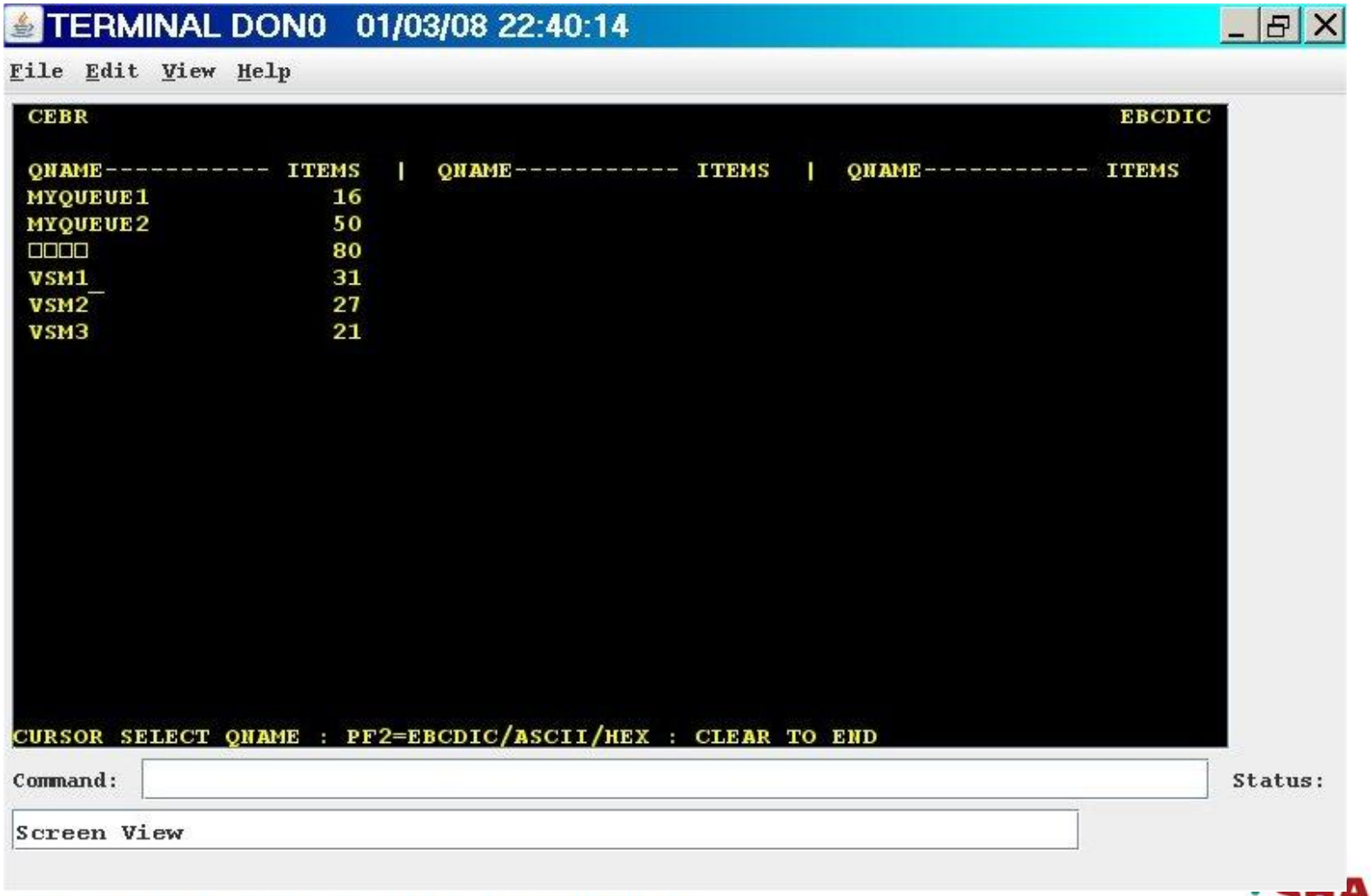
zCICS CEMT INQUIRE/SET FILE 3



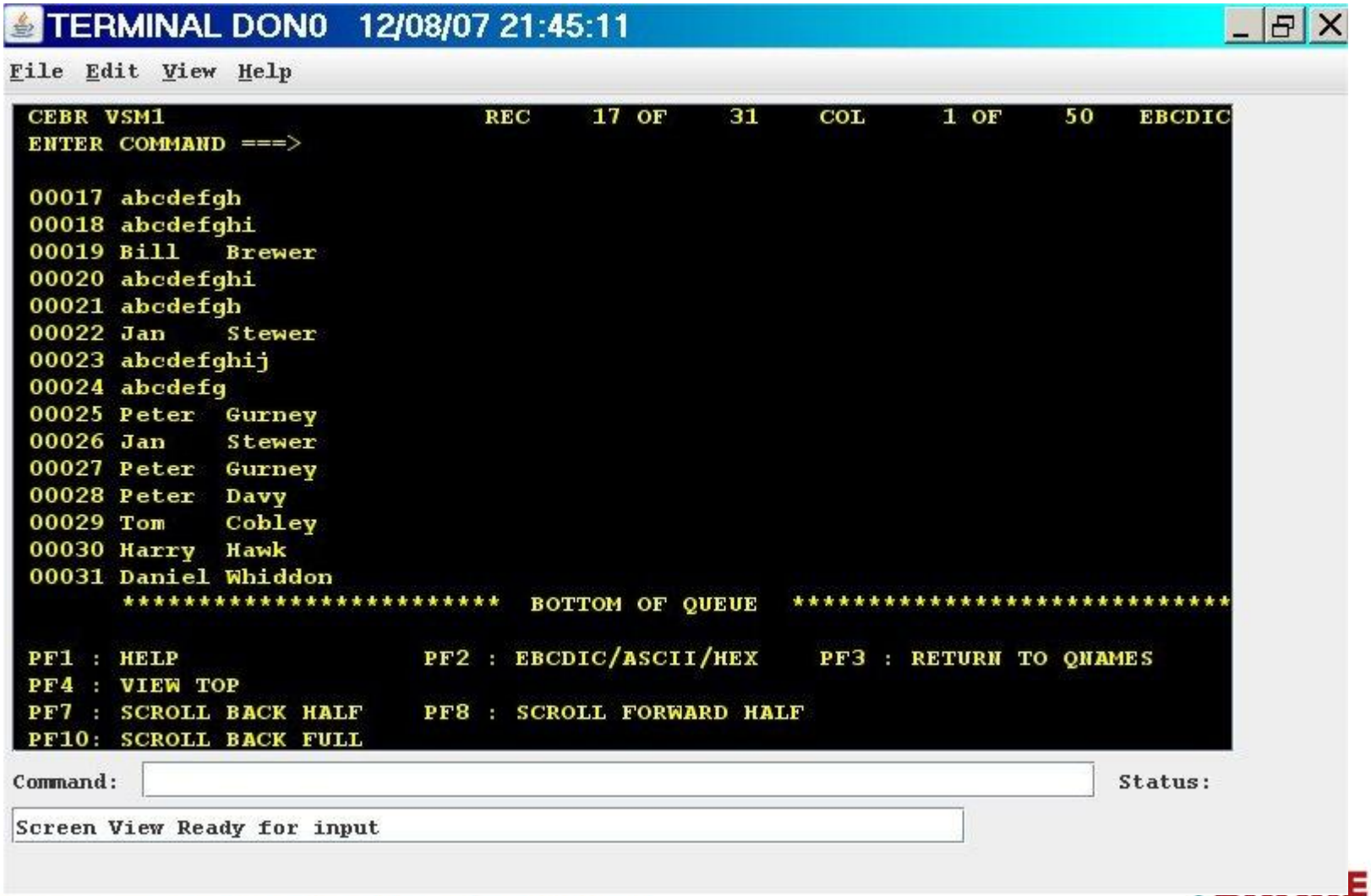
zCICS CEMT INQUIRE/SET FILE 4



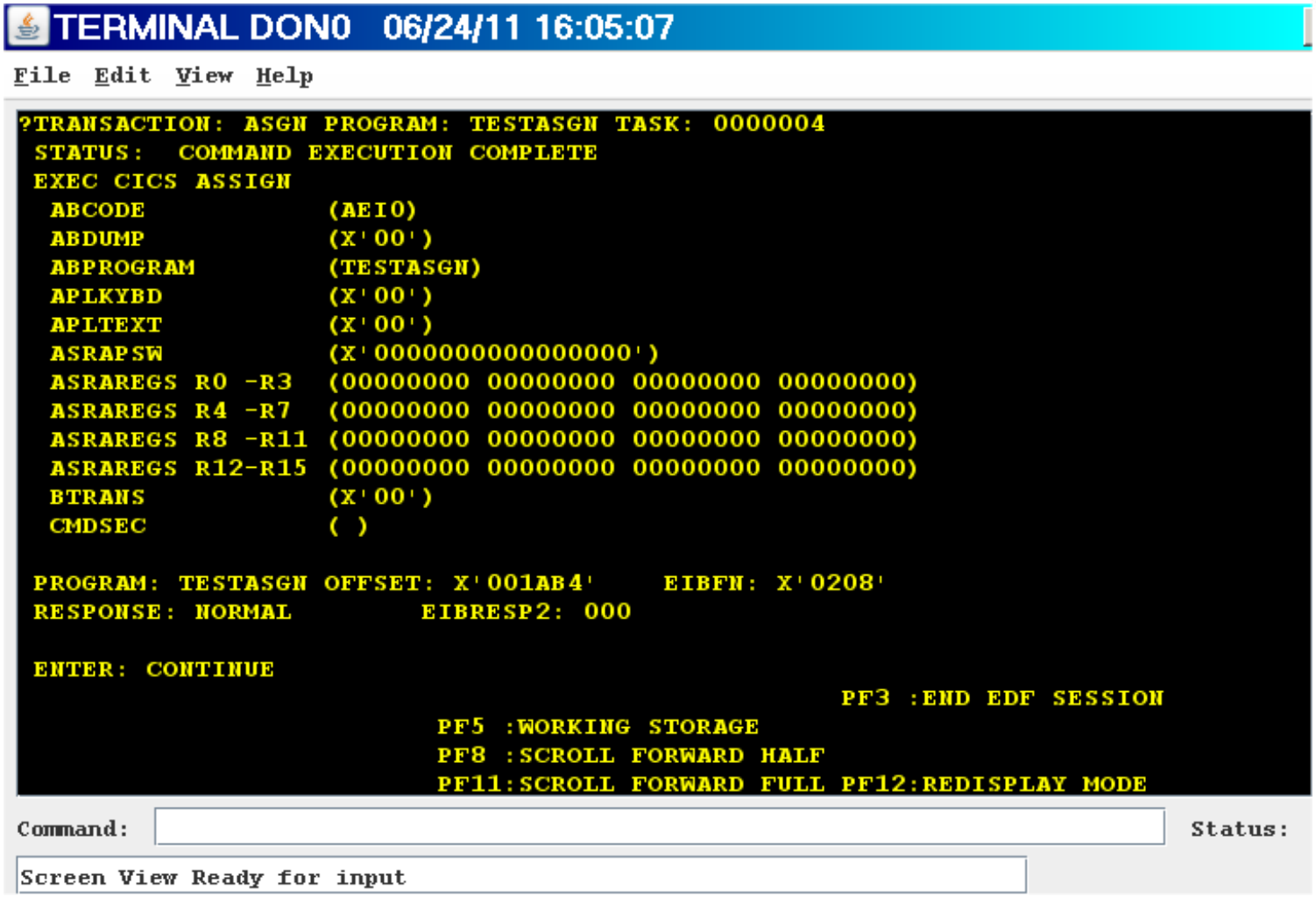
zCICS CEBR 1


TERMINAL DON0 01/03/08 22:40:14
File Edit View Help
CEBR EBCDIC
QNAME---------- ITEMS | QNAME---------- ITEMS | QNAME---------- ITEMS
MYQUEUE1 16
MYQUEUE2 50
□□□□ 80
VSM1 31
VSM2 27
VSM3 21
CURSOR SELECT QNAME : PF2=EBCDIC/ASCII/HEX : CLEAR TO END
Command:
Status:
Screen View

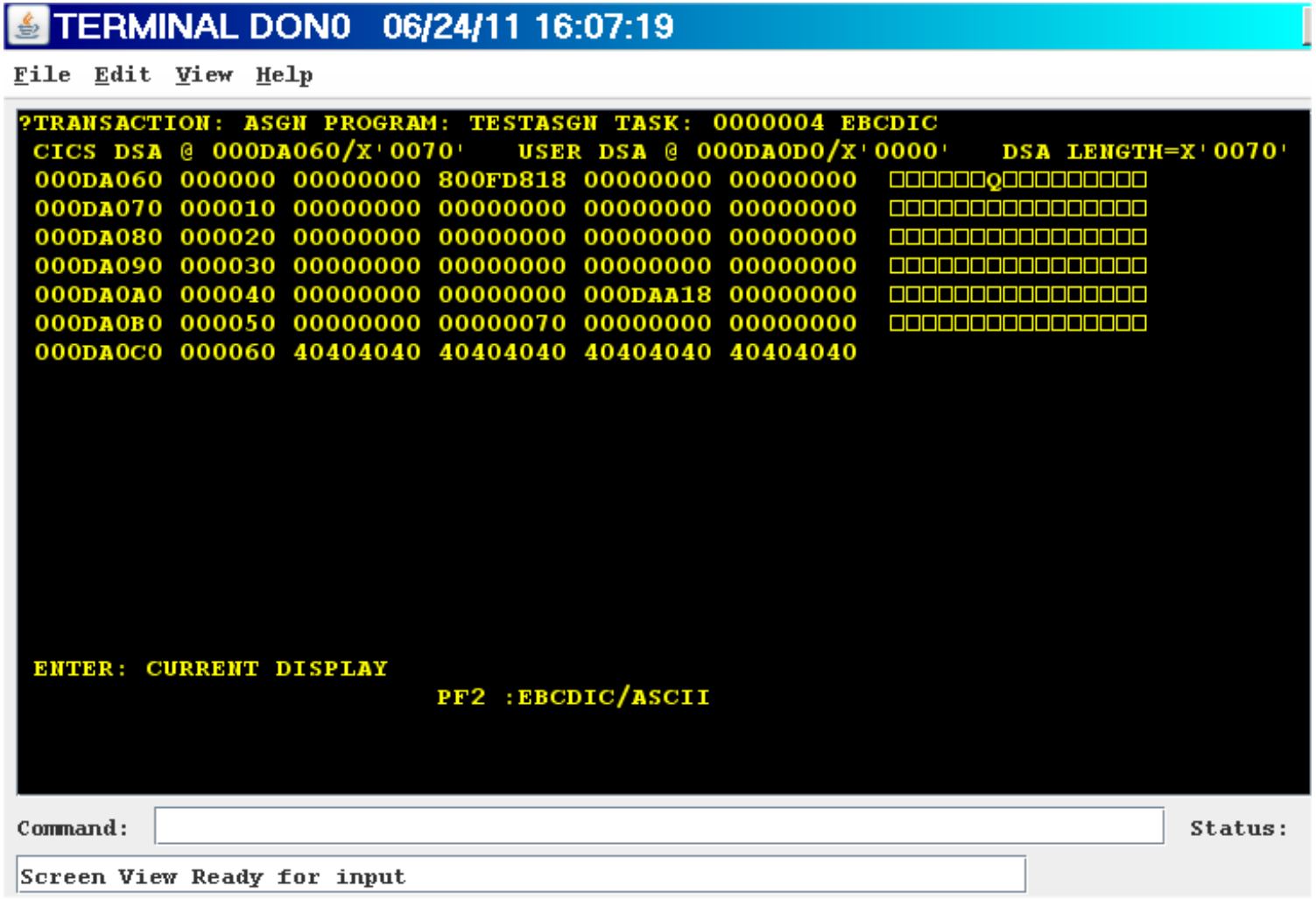
zCICS CEBR 2


TERMINAL DON0 12/08/07 21:45:11
File Edit View Help
CEBR VSM1 REC 17 OF 31 COL 1 OF 50 EBCDIC
ENTER COMMAND ===>
00017 abcdefgh
00018 abcdefghi
00019 Bill Brewer
00020 abcdefghi
00021 abcdefgh
00022 Jan Stewer
00023 abcdefghij
00024 abcdefg
00025 Peter Gurney
00026 Jan Stewer
00027 Peter Gurney
00028 Peter Davy
00029 Tom Cobley
00030 Harry Hawk
00031 Daniel Whiddon
********** BOTTOM OF QUEUE **********
PF1 : HELP PF2 : EBCDIC/ASCII/HEX PF3 : RETURN TO QNAMES
PF4 : VIEW TOP
PF7 : SCROLL BACK HALF PF8 : SCROLL FORWARD HALF
PF10: SCROLL BACK FULL
Command:
Status:
Screen View Ready for input

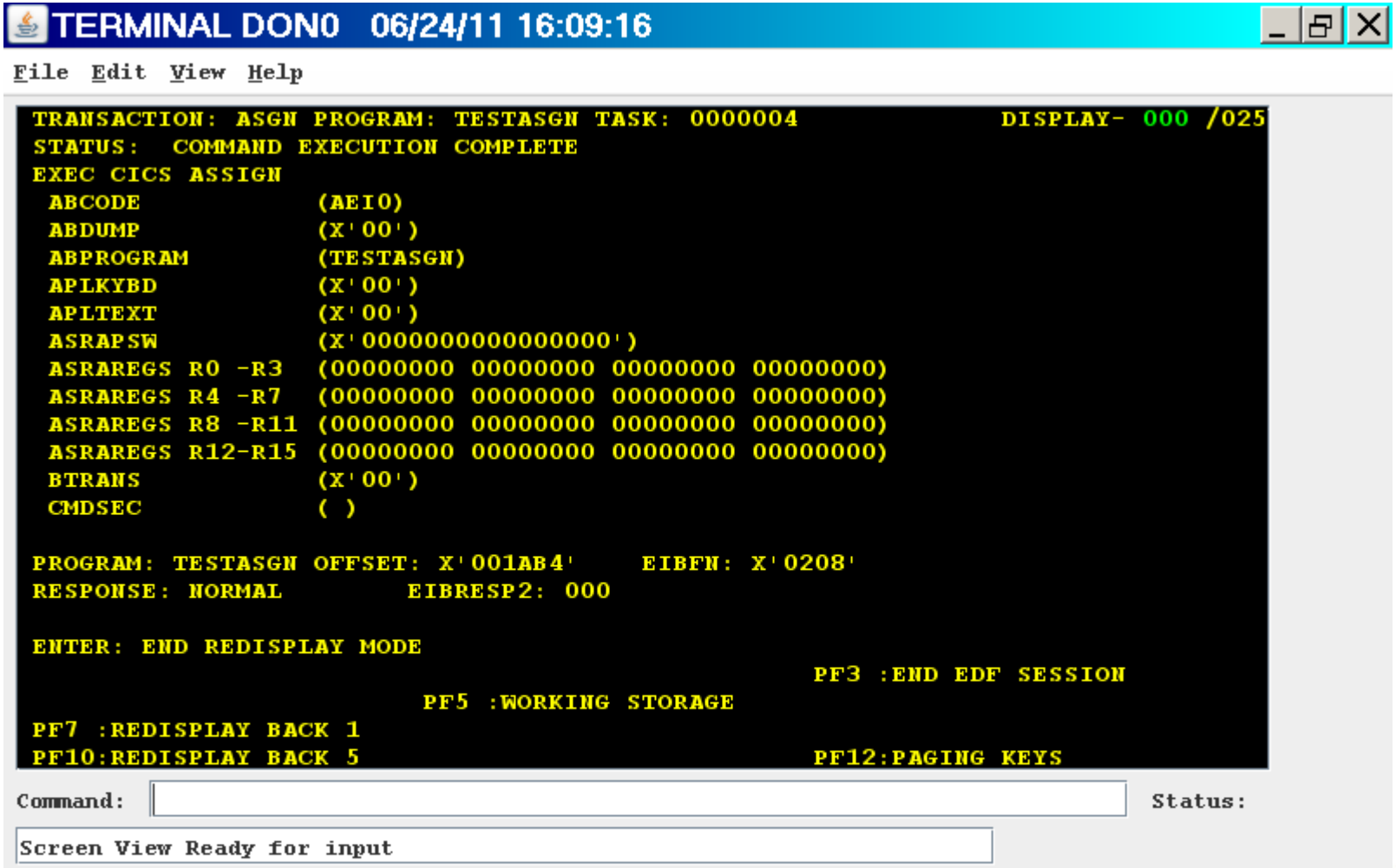
zCICS CEDF 1


TERMINAL DON0 06/24/11 16:05:07
File Edit View Help
?TRANSACTION: ASGN PROGRAM: TESTASGN TASK: 0000004
STATUS: COMMAND EXECUTION COMPLETE
EXEC CICS ASSIGN
ABCODE (AEIO)
ABDUMP (X'00')
ABPROGRAM (TESTASGN)
APLKYBD (X'00')
APLTEXT (X'00')
ASRAPSW (X'0000000000000000')
ASRAREGS R0 -R3 (00000000 00000000 00000000 00000000)
ASRAREGS R4 -R7 (00000000 00000000 00000000 00000000)
ASRAREGS R8 -R11 (00000000 00000000 00000000 00000000)
ASRAREGS R12-R15 (00000000 00000000 00000000 00000000)
BTRANS (X'00')
CMDSEC ( )
PROGRAM: TESTASGN OFFSET: X'001AB4' EIBFN: X'0208'
RESPONSE: NORMAL EIBRESP2: 000
ENTER: CONTINUE
PF3 :END EDF SESSION
PF5 :WORKING STORAGE
PF8 :SCROLL FORWARD HALF
PF11:SCROLL FORWARD FULL PF12:REDISPLAY MODE
Command:
Status:
Screen View Ready for input

zCICS CEDF 2


TERMINAL DON0 06/24/11 16:07:19
File Edit View Help
?TRANSACTION: ASGN PROGRAM: TESTASGN TASK: 0000004 EBCDIC
CICS DSA @ 000DA060/X'0070' USER DSA @ 000DA0D0/X'0000' DSA LENGTH=X'0070'
000DA060 000000 00000000 800FD818 00000000 00000000
000DA070 000010 00000000 00000000 00000000 00000000
000DA080 000020 00000000 00000000 00000000 00000000
000DA090 000030 00000000 00000000 00000000 00000000
000DA0A0 000040 00000000 00000000 000DAA18 00000000
000DA0B0 000050 00000000 00000070 00000000 00000000
000DA0C0 000060 40404040 40404040 40404040 40404040
ENTER: CURRENT DISPLAY
PF2 :EBCDIC/ASCII
Command:
Status:
Screen View Ready for input

zCICS CEDF 3


TERMINAL DON0 06/24/11 16:09:16
File Edit View Help
TRANSACTION: ASGN PROGRAM: TESTASGN TASK: 0000004 DISPLAY- 000 /025
STATUS: COMMAND EXECUTION COMPLETE
EXEC CICS ASSIGN
ABCODE (AEI0)
ABDUMP (X'00')
ABPROGRAM (TESTASGN)
APLKYBD (X'00')
APLTEXT (X'00')
ASRAPSW (X'0000000000000000')
ASRAREGS R0 -R3 (00000000 00000000 00000000 00000000)
ASRAREGS R4 -R7 (00000000 00000000 00000000 00000000)
ASRAREGS R8 -R11 (00000000 00000000 00000000 00000000)
ASRAREGS R12-R15 (00000000 00000000 00000000 00000000)
BTRANS (X'00')
CMDSEC ( )
PROGRAM: TESTASGN OFFSET: X'001AB4' EIBFN: X'0208'
RESPONSE: NORMAL EIBRESP2: 000
ENTER: END REDISPLAY MODE
PF3 :END EDF SESSION
PF5 :WORKING STORAGE
PF7 :REDISPLAY BACK 1
PF10:REDISPLAY BACK 5
PF12:PAGING KEYS
Command:
Status:
Screen View Ready for input

zCICS CEDF 4


TERMINAL DON0 06/24/11 16:11:44
File Edit View Help
TRANSACTION: ASGN PROGRAM: TESTASGN TASK: 0000004 EBCDIC DISPLAY- ?06 /025
STATUS: COMMAND EXECUTION COMPLETE
EXEC CICS SEND
FROM (C□G/R12:8008A402 R13:000DA060 R14:0008C314 R15:00000000)
LENGTH (00055)
PROGRAM: TESTASGN OFFSET: X'000726' EIBFN: X'0404'
RESPONSE: NORMAL EIBRESP2: 000
ENTER: END REDISPLAY MODE
PF2 :EBCDIC/ASCII/HEX PF3 :END EDF SESSION
PF5 :WORKING STORAGE
PF7 :REDISPLAY BACK 1 PF8 :REDISPLAY FORWARD 1
PF10:REDISPLAY BACK 5 PF11:REDISPLAY FORWARD 5
Command:
Status:
Screen View Ready for input

zCICS Sequential Terminal Support

- **Regression test your transactions.**

- **Run a transaction with INI parm SEQ_TERM=TRACE**
- **Run the extract program Z390SEQ to build the data streams**
- **Sequence all of your data streams**
- **Application changes occur**
- **Set INI parm SEQ_TERM=YES**
- **Run the simulation, you can see it happen on screen**
- **Your whole life will flash before your eyes**

- **Run the comparator Z390CMPG, review the output**
- **Refine the comparator by building an exclusion file for variable data like dates and times**

zCICS Documentation 1

- There's a lot of it.
  - None of it is meant to replace IBM's Manuals.
  - The information given refers to zCICS, its implementation, workings, extensions and command/parameter support.

zCICS Documentation 2

- Readme
- Application Programming Guide
- Diagnosis Reference
- History
- Sequential Terminal Support
- Supplied Transactions
- System Programmer's Guide
- VSAM Guide
- Basic Mapping Support

zPAR

zPAR (Program Analysis Reports)

- Originally written as part of internal testing and QA
- Generates statistical reports about assembly/compile, link, and execution of z390 and zCOBOL programs, including opcode distribution and executed opcodes
- Source Tracing Reports using TRE trace files generated by ez390 emulator execution
- www.z390.org/zpar/

Can z390 and zCOBOL do the job?

Z390/zCOBOL Exercise/Stress Test

- James Francis Cray built a demonstration/exercise/stress test system
- zCOBOL + z390 + IBM DB2 Express-C (all free)
- Environment
  - 10-year-old 2.8Ghz single core hyper-threaded Intel processor
  - Windows XP with Media Center 2005
  - 4GB RAM, 80GB hard drive

z390/zCOBOL Stress Test

- 32 simultaneous Windows processes executing
  - zCOBOL compile
  - z390 assemble
  - Link
  - 1000 times, program starts a CMD.EXE process via the CMDPROC macro to issue a “DB2 INSERT” SQL command to a single table
- Approximately 20 minutes to do all of the above on this ancient system, and in about 10 minutes on a cloud platform

z390/zCOBOL Stress Test

- Prototype system design and architecture
- “Rightsize” applications currently on the mainframe

Yes, z390 and zCOBOL can do the job at minimal cost!

z390 and zCOBOL–where can I get it?

z390 and zCOBOL–where can I get it?

- Download z390 and zCOBOL from z390 web site
- MSI file for Windows, ZIP file for Linux
- Java source and z390.jar executable
- Documentation included and also available online
- Demos and regression tests included

z390, zCOBOL, zCICS, and zPAR Documentation

- All z390, zCICS, and zPAR documentation is available online at www.z390.org
- All zCOBOL documentation is available online at www.zcobol.org
- Package for z390 includes zCOBOL, zCICS, and zPAR
  - Documentation on all components
  - Demo Programs
  - User Guides
  - NIST ANSI 85 COBOL Test Suite Results
  - Options
  - Regression Test Programs

z390 and zCOBOL-what's just happened?

z390 and zCOBOL Project Updates

- Big Changes™ have occurred
- More Changes™ are coming

Recent personnel changes

- Don Higgins has retired (finally)
- New co-administrators
  - M. Ray Mullins, Cat Herder Software, LLC
  - Abe Kornelis, B.V. Bixoft
- New z390/zCOBOL developers and their primary areas
  - Ray & Abe
  - James Francis Cray, independent consultant
  - John Hennesy, independent consultant
- Melvyn Maltz continues with zCICS
- Don Higgins continues to assist, at his pace

Recent personnel changes

- New webmaster and designer
  - Jill M. Sheehan of JillThePill Design, LLC
  - Designed catherdersoftware.com
- Kristin Bryant, QA
  - Software engineer looking to expand her horizons

WWW–related changes

- Web host consolidated on SourceForge
  - Current Web pages hosted on Sourceforge
  - Problem & enhancement ticket databases (Tracker)
  - Wiki (slowly being populated)
  - Source code repository
    - Git (read-write for project developers)
    - SVN (read-only)
  - Possible future host for mailing lists and discussions (depending on the future of Yahoo!)
  - Redesign of web site and branding

Support changes

- Manual RPI system and form discontinued
- New SourceForge Tracker databases
  - Problems
  - Enhancements
  - Open and closed RPIs imported into data bases
    - Ticket number is *not* the same as RPI number
    - RPI number is in summary and description for easy searching
    - Some dates may not carry forward
- Old RPI web pages kept for reference
- Allows for a more open process and monitoring capability

Support changes

- Approximately 25 RPIs which were received since v1.5.0.6 was released are being manually entered into the Tracker databases
- This process should be complete by the end of next week

Discussion, news, general front-line support: Yahoo!® Groups

- z390 Group – join at groups.yahoo.com/groups/z390 or send an email to z390–subscribe@yahoogroups.com
- zCOBOL Group – join at groups.yahoo.com/groups/zcobol or send an email to zcobol–subscribe@yahoogroups.com
- Still around but low traffic:
  - z390-assembler-contest Group – join at groups.yahoo.com/groups/z390-assembler-contest or send an email to z390-assembler-contest-subscribe@yahoogroups.com
  - PC370 Group – join at groups.yahoo.com/groups/PC370 or send an email to pc370-subscribe@yahoogroups.com

Other z390 and zCOBOL Internet presences

- z390-asm.blogspot.com
- Twitter: @z390_assembler
- Twitter: @zCOBOL
- www.facebook.com/z390assembler
- www.facebook.com/zCOBOL
- Google+ Pages for both z390 and zCOBOL
  - No URL because Google makes it difficult
  - Click on the G+ icon on the respective home page

z390 and zCOBOL-what's next?

Plans for the next year

- Bug fixes (tentative release date end Q2)
- Improved internal opcode table
  - Allows implementation of OPCODE/MACHINE options to better mimic HLASM
- LIST(121)/(133), cross-reference and list options, and ASA/MCC options to produce listings compatible with HLASM
- Port ASMXML from Flat Assembler
- SORT enhancements, including E15/E35 emulation
- Enhanced z/OS and z/VSE macro/service emulation

Plans for the next year (and beyond)

- zEC12 instruction support
- VSAM emulation enhancements
  - Alternate index and path support, including update (z390, zCOBOL, zCICS)
- Other zCICS enhancements based on the above
- Other assembler options
- More COBOL verbs and NIST ANSI 85 test suite completion
- Partial COBOL ISO 2002 implementation (possibly)

Plans for the next year (and beyond)

- Expansion of zCOBOL intermediate code generation to other architectures
- SQL support (z390, zCOBOL, zCICS)

And down the road, maybe…

- More TSO emulation?
- z/VM CMS emulated environment?
- Access register support?
- BS2000/OSD emulated environment?
- Other System/370- and System/390-based operating system emulations?
- Integration with scripting languages like REXX, Perl, and Python
- TSO CLIST and VM/370 EXEC2? ☺

What you've just heard about

- z390 Portable Mainframe Assembler
- zCOBOL Portable Mainframe Compiler
- zCICS
- zPAR
- Recent administrative and developer changes
- Changes in support
- What's next

Thank you, Brie Adams, the graduate assistant for my Sac State COMS 103 class in Fall 2011, for making this presentation possible, and for going over it several times, and not being shy with suggestions, even though you don't know anything about mainframes.

Questions? des Questions? Fragen? ¿Preguntas? Spurningar?
Answers? des Réponses? Antworten? Respuestas? Svör?

Bueller?

So you think you can program? Do you have some free time?

- We are always looking for more hands with some spare time for volunteering (meaning the only compensation is a feeling of a job done well)
- Good opportunity to learn Java in a non-traditional environment (batch and basic GUI)
- Good opportunity to keep those programmatic juices flowing
- We would like Mac OS X, Linux, FreeBSD, other UNIX, or, heck, even z/OS UNIX® System Services ☺ developers
- Looking for someone currently working with BS2000/OSD
- Contact Ray Mullins or Abe Kornelis for more information

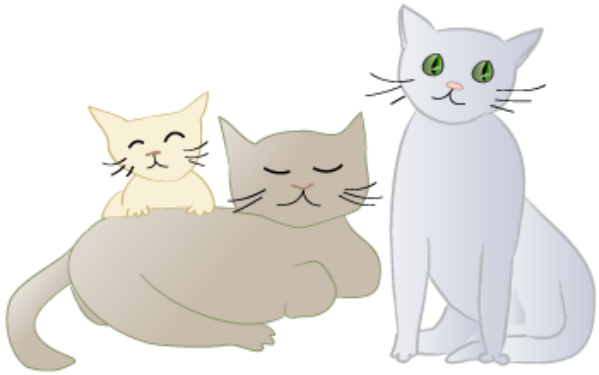
Project Contact Information

Abe Kornelis–B.V. Bixoft
Contact Information

- www.bixoft.com
- Email: bixoft@bixoft.nl
- Twitter: @abekornelis
- Facebook: www.facebook.com/abe.kornelis.7
- LinkedIn: nl.linkedin.com/in/abekornelis
- XING: www.xing.com/profile/Abe_Kornelis
- Follow him on Google+ (personal profile)

M. Ray Mullins–Cat Herder Software, LLC Contact Information

- www.catherdersoftware.com
- Email: catherdersoftware@gmail.com
- Twitter: @catherdersoft
- Facebook: www.facebook.com/cat.herder.software
- LinkedIn: www.linkedin.com/in/raymullins/
- XING: www.xing.com/profile/Ray_Mullins
- Follow us on Google+



Cat Herder Software

Legal mumbo-jumbo

- The z390 and zCOBOL Open Source Project is now co-administered by M. Ray Mullins of Cat Herder Software, LLC, and Abe Kornelis of B.V. Bixoft
- Copyright of the core z390 sources was assigned to Cat Herder Software, LLC from Automated Software Tools, Inc. in August 2012
- Other developers continue to hold their respective copyrights
- All z390 software is distributed under the GNU Public License V2

#SHAREorg

SHARE
Technology · Connections · Results

z390, zCOBOL, and zCICS: What It Is, What's New, and What's Next

M. Ray Mullins
Cat Herder Software, LLC

Tuesday, 5 February 2013 1100–1200ish
Session 12252

I can be found on all of the following social networks:

SHARE in San Francisco 2013