AAEC/E425

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT

LUCAS HEIGHTS

AN INTERACTIVE COMPUTING SYSTEM FOR THE AAEC DATAWAY NETWORK

by

R.J. CAWLEY
G.D. TRIMBLE

December 1977

ISBN 0 642 596352

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT

LUCAS HEIGHTS

AN INTERACTIVE COMPUTING SYSTEM FOR THE AAEC DATAWAY NETWORK

by

R. J. CAWLEY

G. D. TRIMBLE

ABSTRACT

A package of routines is described which allows the IBM360 programmer access to any terminal attached to the AAEC Dataway network.

National Library of Australia card number and ISBN 0 642 596352

The following descriptors have been selected from the INIS Thesaurus to describe the subject content of this report for information retrieval purposes. For further details please refer to IAEA-INIS-12 (INIS: Manual for Indexing) and IAEA-INIS-13 (INIS: Thesaurus) published in Vienna by the International Atomic Energy Agency.

COMPUTER NETWORKS; INTERACTIVE DISPLAY DEVICES; EQUIPMENT INTERFACES; PROGRAMMING; FORTRAN

CONTENTS

1. INTRODUCTION

A package of routines has been developed which allows users of the AAEC central IBM 360/65 computer to have access to any terminal attached to the AAEC Dataway network. The package is designed for use from FORTRAN programs but may be used from other high level languages or assembler programs.

Special intercept routines are used to re-direct FORTRAN I/O and execution time error messages. A special version of the AAEC free input routine SCAN is also available which allows input from a terminal. Finally, a version of the AAEC low level plot routine is provided to allow real-time graphics.

2. IBM 360 - DATAWAY SYSTEM

The IBM 360/65 is linked to a PDP9L computer which acts as a control unit for 128 device addresses [Richardson 1969]. Also attached to the PDP9L is the multiple user Dataway [Ellis 1970] which connects to several computer and single user terminals. The PDP9L maps IBM 360 device addresses to Dataway addresses and so each Dataway terminal has a unique IBM 360 I/O address.

Any communication sequence on the Dataway consists of one computer signalling another, with a single byte specifying the operation required. This byte is the command code (corresponding to a channel command in the IBM 360). The most commonly used codes are described briefly in Table 1. Quite often the remote computer will be unable to satisfy the command immediately; for example, if the command is X'02', the computer has to wait for the user to type in a line of data. In this case the remote computer gives busy status response to the command and then prompts the input from the user. When the user has typed in the data the remote computer signals ready status with the X'03' command and the I/O is re-initialised by the PDP9L.

The remote computer signalling codes are sometimes referred to as 'primary write' (X'05') and 'nop control' (X'03').

When terminals signal asynchronous attention, a task is initiated in the IBM 360. The task to be started is determined by up to 24 bytes of diagnostic data which are sent along with the attention request to the IBM 360.

For interactive jobs a terminal is considered to have attention capability when it is set up to give X'CF02' as the diagnostic data accompanying the asynchronous attention request. These data cause a task, called AEF02, to be executed. This task uses the local SVC 202 [Richardson 1976] to notify the user's interactive job that asynchronous attention has been signalled from the terminal.

The other type of attention that can be signalled is synchronous attention. This is achieved when the remote computer signals unit exception status

TABLE 1

DATAWAY SIGNALLING CODES

Remote Computer Initialised Signalling

Code	Data	Operation
05	(up to)24 bytes	Asynchronous attention
03	none	Status presentation

Central Computer Initialised Signalling

Code	Data	Operation
00	none	test I/O, status request
01	(up to)132 bytes	write, no trailing carriage return/line feed
02	(up to)80 bytes	read, echo characters
05	(up to)132 bytes	write, append carriage return/line feed
06	(up to)80 bytes	read, no character echo
07	none	rewind - signals start
0F	none	rewind unload - signals end
41	(up to)18 bytes	write, attention and tab data
42	(up to)18 bytes	read, attention and tab data
45	(up to)6 bytes	write, controlling information
46	(up to)6 bytes	read, controlling information
7F	none	illegal - invalid asynchronous attention

to a write from the IBM 360. The user requests either type of attention by pressing the question mark character at the terminal. Terminals always have synchronous attention capability.

3. PROGRAMMING VIEWPOINT

3.1 Beginning Interaction

The following routines are used to set up the interactive environment. Their functions are to

(i) ensure the user is logged on at a remote console;

(ii) arrange for HASP to transfer terminal communication from the INTERNAL CONSOLE [Johnstone 1974] to the user's task;

(iii) give the user's terminal asynchronous attention capability;

(iv) inform the operator that user-job interaction has been initiated; and

(v) inform the user at the terminal that interactive communication has been initialised.

These routines use the local internal console SVC 205 [James 1976] to test for the presence of the user and gain control of the terminal unit control block (UCB).

The subroutine and calling sequences are as follows:

DWTEST CALL DWTEST (&N)

This entry tests whether the user is logged on at a remote console. If not, control is transferred to statement number N; otherwise return is made to the statement following the CALL.

DWSTRT CALL DWSTRT

This entry arranges for the HASP INTERNAL CONSOLE [Johnstone 1974] to transfer terminal communication to the user's task and so begin the interactive session.

3.2 Terminal Communication

The following routines are used to perform I/O to the terminal.

DWRITE CALL DWRITE (BUF,L) or CALL DWRITE (BUF)

This entry writes a line of text to the terminal and requests the terminal to append a carriage return/line feed.

BUF is the address of the array containing the text;

L is an integer variable which specifies the length of the string to be written (in bytes or characters).

If L is omitted, the length is determined by the presence of a terminating character (see below). Trailing blanks are removed and the length adjusted before writing to the remote terminal. A maximum length of 132 characters is allowed, the length being truncated to 132 if greater.

The user's buffer is moved to an output buffer and translated from EBCDIC to ASCII before being written to the terminal.

DWRITN CALL DWRITN (BUF,L) or CALL DWRITN (BUF)

This entry writes a line of text to the terminal. No trailing carriage return/line feed is written. Arguments are the same as for DWRITE except that trailing blanks are not removed.

3.2.1 Concept of a terminating character

A fundamental parameter of any string of characters is its length. However in computer hardware the length of a string is not apparent. For this reason the use of a special (terminating) character to delimit the string is

necessary. For example, if $ is the terminating character, then the core string,

'DATA$GARBAGE.....'

would be recognised as just 'DATA'. To set a terminating character for use with the interactive routines, the routine XYTLEN is used; for example to set # to be the terminating character:

CALL XYTLEN ('#').

DWREAD CALL DWREAD (BUF,L)

This entry reads a line of text from the terminal and requests the terminal to echo input characters.

BUF is the address of an 80-character area where the input text will be stored;

L is an integer variable where the length of the input line will be stored.

The ASCII characters read from the terminal are translated to EBCDIC before returning to the user.

DWREAS CALL DWREAS (BUF,L)

This entry reads a line of text from the terminal and requests the terminal not to echo input characters. Arguments are the same as for DWREAD.

DWCAN CALL DWCAN

This entry sends the ASCII character sequence CANCEL,HOME to the terminal. If the terminal is a display screen this results in erasing the screen and positioning the screen pointer at the top left hand corner of the screen.

DWHOME CALL DWHOME

This entry sends the ASCII character HOME to the terminal. If the terminal is a display screen, the screen pointer is positioned at the top left-hand corner of the screen.

DWTABS CALL DWTABS (N1,N2,.....,N9)

This entry sends input tab information to the terminal. Up to 9 arguments may be present. N1,N2,.....,N9 are sequential tabulation positions. If no arguments are present then the terminal is given asynchronous attention capability.

If N1 is zero, asynchronous terminal attention capability is cancelled and further arguments are treated as tabs.

If N1 is non-zero, the terminal is given asynchronous attention capability and all arguments are treated as tabs. The initial default settings are 'No tabs' and the terminal has asynchronous attention capability.

DWQTST CALL DWQTST (&N)

This routine tests if asynchronous attention has been given or if a unit exception condition (synchronous attention) was received as a response to a previous WRITE to the terminal. Both these conditions are cleared. If either condition is true then return is made to statement number N, otherwise return is to the statement following the CALL.

3.3 FORTRAN I/O to the Terminal

The user may wish to use normal FORTRAN READ or WRITE statements to communicate with the terminal. The subroutine AEREREAD [Davids 1970] has been modified to facilitate this and the entry DWSETU is used to set up an intercept for terminal I/O. AEREREAD intercepts calls from IBCOM to FIOCS which is the normal FORTRAN I/O path.

DWSETU CALL DWSETU (IN,IOUT)

This entry defines the FORTRAN unit numbers to be used for terminal I/O.

IN is the unit number to be used for input;

IOUT is the unit number to be used for output.

The input unit number must not be the same as the output unit number. A value of zero for either IN or IOUT removes terminal I/O capability for input or output respectively. (May be reset by another call to DWSETU with IN or OUT non-zero.) The unit numbers specified may be altered during the terminal session if desired.

After calling DWSETU, all READs from unit IN come from the terminal and all WRITEs to unit IOUT go to the terminal. Maximum input buffer length is 80 bytes and maximum output buffer length is 133 bytes.

If unit 1 is specified for input then the terminal does a carriage return/line feed before the READ and the user is prompted by the terminal bell. For other unit numbers a normal READ is performed.

If unit 3 is specified for output then the printer carriage control characters blank, 0, -, +, 1 are simulated; otherwise a carriage return/line feed is appended to each WRITE. Hence units 1 and 3 should not be mixed with other unit numbers.

The FORTRAN END= and ERR= options for I/O statements are not currently supported. The initial setting is IN=0 and IOUT=0. Calls to DWREAD and DWRITE may be interspersed with FORTRAN terminal I/O.

3.4 FORTRAN Error Monitor Messages

FORTRAN error monitor messages may be directed to the terminal with the subroutine DWERMS. Its calling sequence is

CALL DWERMS (OPTION)

If OPTION contains the character string 'ON', then error monitor messages are directed to the terminal. If OPTION contains the character string 'OFF', then error monitor messages revert to FORTRAN unit 3. A CALL to DWSETU specifying unit 3 for terminal output does not override this. This is because error messages are not written by the normal IBCOM-FIOCS path, but by a direct call from the error monitor to an entry FIOCSBEP in FIOCS.

3.5 Terminal Input Using SCAN Subroutine

A special version of the low level subroutine RDSCAN (DWRDSCAN) allows the subroutine SCAN [Bennett & Pollard 1967] to be used for terminal input.

The user should specify an input unit number of -1 or -2 in his SCAN call if he wishes to read from the terminal.

UNIT = -1 reads from the terminal and input characters are echoed;

UNIT = -2 reads from the terminal and input characters are not echoed.

Note that the maximum input buffer size is 80 characters.

If the terminal responds with three consecutive carriage returns to a read by SCAN, then RDSCAN writes 'END OF FILE' and terminates the job unless the end of file exit option has been set by a call to OPSCAN. The standard version of SCAN operates from the terminal by use of a positive unit number and corresponding DWSETU call. However, no facility for end of file is provided.

3.6 Ending Interaction

The interactive terminal session is concluded by the routine DWEND. Its calling sequence is

```
CALL DWEND
```

The user should not call any of the 'DW' routines except DWSTRT or DWTEST after calling DWEND. Any attempt to do so results in a no-operation. This routine uses SVC 205 to relinquish control of the terminal and return it to the internal console.

3.7 Low Level Routines

The low level routines described here perform the following functions:

(i) Setting up the DDLESS I/O environment at start-up.

(ii) All I/O using the EXCP macro.

(iii) Optional translation of I/O buffers from ASCII to EBCDIC and EBCDIC to ASCII respectively.

(iv) Removing the DDLESS I/O environment at termination.

These routines are in general for the internal use of the package; however, users with special requirements may sometimes need to call them directly.

DWTRAN#

This module contains three entry points - DWOPEN, DWTRAN and DWCLSE - a separate CSECT which contains the two translate tables, and a three word COMMON block.

DWOPEN

Called by DWSTRT to set up a DEB, IOB and DCB for terminal I/O. This routine uses the local SVC 248 [Richardson 1974] to establish these control blocks.

DWCLSE

Called by DWEND to close and remove the DCB, IOB and DEB. (Uses SVC 248).

DWTRAN

This is the entry which performs all terminal I/O. Its calling sequence is

```
CALL DWTRAN(BUF,L,$CMD,ISW,&N1,&N2,&N3)
```

where

BUF is the address of the I/O buffer;

L is an integer variable which specifies the length of BUF in bytes;

$CMD is a 1-byte variable which specifies the channel command to be used;

ISW is an integer variable which specifies if translation is to occur:
ISW = 0 no translation,
ISW = 1 translate BUF from EBCDIC to ASCII before performing I/O,
ISW = 2 translate BUF from ASCII to EBCDIC after performing I/O;

N1 is the statement number to which control is transferred if the channel status at conclusion of I/O is 'UNIT EXCEPTION';
(No data will have been transferred in this case.)

N2 is the statement number to which control is transferred if the I/O failed;

N3 is the statement to which control is transferred if no open DCB is present for terminal communications.

After return, general register 0 contains the number of bytes transferred.

COMMON DWCOM$

This common block has three entries, each of one word:

```
COMMON/DWCOM$/IDWEL,DWCS,IDWQM
```

where

IDWEL is an integer variable which contains the number of bytes transferred by the last CALL to DWTRAN;

DWCS contains in the low order 8 bits the status bits from the channel status word at the conclusion of the last terminal I/O;

IDWQM is an integer variable which is 0 if unit exception has not been signalled, and 1 if unit exception status has resulted from a previous I/O. A CALL to DWQTST resets this flag to zero.

The user should not attempt to extend the length of this COMMON block.

COMMON DWTRN$

This COMMON block contains the 2 translate tables used by DWTRAN:

COMMON/DWTRN$/E2A,A2E

Each table is 256 bytes long. E2A is the EBCDIC to ASCII translate table (see Appendix A), A2E is the reverse (see Appendix A).

The user should not attempt to extend the length of this COMMON block.

DWCASE

This entry alters the ASCII to EBCDIC translate table in COMMON/DWTRN$/ . Its calling sequence is

CALL DWCASE (OPTION)

where

OPTION is a 1-byte character:

'L' sets table to translate lower case ASCII alphabetic characters to lower case EBCDIC characters;

'U' sets table to translate lower case ASCII alphabetic characters to upper case EBCDIC characters.

The default ASCII to EBCDIC translate table corresponds to the option 'U'.

DWMSG

This routine is called by the DWCOM$ module to print error messages on FORTRAN unit 3. Its calling sequence is

CALL DWMSG (BUF,L)

BUF is the address of the message;

L is its length.

DWEXIT

This routine is called by the DWCOM$ module when a permanent I/O error occurs. DWEXIT does a FORTRAN STOP.

3.8 Wait Routine

This routine sends the user's IBM 360 task into a wait state. The call is

CALL SWAIT (N)

where the wait state persists for N/100 seconds.

4. LINKAGE EDITOR CONSIDERATIONS

The routines described are all available from the load module library 'PHYS.FORTLIB'. However the following precautions should be observed if the routines DWSETU, DWERMS or the special RDSCAN are required.

These routines replace standard versions of the following routines and as such must be specifically included in the user's load module.

DWSETU replaces AEREREAD

DWERMS replaces FIOCSBEP and FIOCS#

DWRDSCAN replaces RDSCAN

The user of these routines is thus advised to include the following in his linkage editor input,

```
INCLUDE SYSLIB(DWSETU,DWERMS,DWRDSCAN)
```

The entire package is available with an

```
INCLUDE SYSLIB(DWCOMM)
```

5. USE OF THE PACKAGE FROM OTHER LANGUAGES

The basic package is written in assembler language and uses standard linkage conventions. If the user provides his own versions of the low level routines DWMSG and DWEXIT all routines except DWSETU, DWERMS and DWRDSCAN are available for use.

If the user's DWEXIT routine is entered he should not return to the package as it does not expect a return from DWEXIT. An OC1 abend occurs if a return is made. The package calls DWEND before calling DWEXIT.

The error returns described for FORTRAN calling sequences correspond to the following return codes in register 15:

&RET1 is R15 = 4

&RET2 is R15 = 8

&RET3 is R15 = 12

Normal return corresponds to R15 = 0 .

6. TERMINAL VIEWPOINT

The steps in running an interactive job are

1. the user submits his job as a CLASS=I job, (HASP will automatically hold the job;
2. he starts a HASP INTERNAL CONSOLE at his terminal and logs on using his ID;
3. he issues a '$RUN' command to the console and nominates his job (either by name or number) in reply to the 'JOB:' question. If an I class initiator is free, the user's job is released to start; if not, he signs off (see step 9) and returns to step 2 later;

4. he types carriage return when prompted by '=' ;
5. the console types

   'START BY JOB xxx'

   when his job is ready to interact;
6. the user issues a '$RUN' command to the console and types an immediate carriage return to the 'JOB:' question;
7. his job should now begin communicating with the terminal;
8. when the user's job finishes the interactive session, the HASP CONSOLE prompts the terminal with the message 'END OF INTERACTION' and then '=' ;
9. the user should terminate the console by replying 'E' .

7. INTERACTIVE GRAPHICS

Several terminals on the AAEC network have graphic display capabilities. These are a TEKTRONIX T4002 display attached to a NOVA computer plus three GT40-type computers which have software to simulate a TEKTRONIX. All batch graphics on the central IBM 360 computer use a package of low level routines [Cox *et al.* 1969] which at their base have two routines (GPSEND and GPLOT) for graph initialisation and pen (or beam) movement. To allow interactive jobs to send plots directly to a graphics terminal, new versions of GPLOT and GPSEND have been written which, instead of generating data for later plotting, send the display information directly to the terminal in TEKTRONIX format.

Only two of the TEKTRONIX modes are used. These are character mode, set by a US character (9F hex) and vector mode, set by a GS character (9D hex). Several other characters also reset the TEKTRONIX to character mode but these will not cause problems as long as the procedures described below are observed.

The interactive graphics routines are written so that the plot initialisation call:

```
CALL GPSEND (1)
```

automatically sets to vector mode and the plot termination call:

```
CALL GPSEND (2)
```

resets to character mode and 'homes' the display. Hence mode problems only occur when interspersed text and graphics are required. To allow for this, the entries GPSUSP and GPREST are provided. A

```
CALL GPSUSP
```

suspends vector mode, sets to character mode and homes the display while a

```
CALL GPREST
```

homes the display and resets to vector mode.

NOTES:

(i) GPSEND(1) makes the display origin (bottom left hand corner) the GPLOT origin.

(ii) One GPLOT unit (normally 1 inch) becomes 68.75 display units. Hence the screen represents an area approximately 14.89 by 11.17 inches.

(iii) The display information is buffered in the IBM 360. To produce plots in real time a

```
CALL GTEND
```

transmits the current buffer contents to the terminal (even if only partly full). A call to GPSUSP also empties the buffer.

Cursor

Several routines are provided for the use of the cursor on the TEKTRONIX and its light pen equivalent on the GT40 displays. These are

(i) `CALL DWCURS (ICX,ICY)`

which returns in ICX and ICY the location of the cursor in screen units ($0 \leqslant ICX \leqslant 1023$; $0 \leqslant ICY \leqslant 767$).

(ii) `CALL GPCURS (X,Y)`

which returns in X and Y the cursor location in GPLOT units.

(iii) `CALL DWREAC (LOGC)`

which returns in LOGC (a LOGICAL*1 variable) the ASCII character that was pressed when the cursor position was returned from the terminal. This call must immediately follow the call to GPCURS or DWCURS.

Note that the cursor operates in character mode, so a call to GPSUSP would be necessary if the cursor is to be used while in vector mode. A call to GPREST would then be necessary to reset to vector mode.

XYPLOT

The XYPLOT package [Trimble 1977] functions normally with the interactive routines. Plot initialisation (XYPAPE etc.) automatically sets vector mode and plot termination (XYEND) resets to character mode. Additionally routines XYSUSP and XYREST should be used in place of GPSUSP and GPREST. An additional routine

```
CALL XYCURS (X,Y,&OUT)
```

returns in X and Y the cursor location in user units windowed onto the graph. The error exit is taken if the cursor is outside the graph.

PLOTSW

For some applications it is desirable to be able to switch the graphics backwards and forwards between the terminal and the normal batch plotting

system. To allow this a special module DWPLOS (see availability below) is provided which is activated by a

```
CALL PLOTSW ('DIRN')
```

where 'DIRN' is a 4-character string which may either be 'DISP' or 'CALC'. If the argument is 'DISP', graphics information is sent directly to the terminal. The argument 'CALC' (which is the initial setting) puts graphics information into the normal batch graphics output queue. Note however that switching must not be made while plotting is active.

Availability

The normal interactive graphics package is available from the load module library PHYS.FORTLIB and should be linked into the user's program with a card containing

```
INCLUDE SYSLIB(DWPLOT)
```

as input to the linkage editor. If PLOTSW is required, this should be replaced by a card containing

```
INCLUDE SYSLIB(DWPLOS)
```

8. DYNAMIC GRAPHICS

The GT40-type computers have refresh display units and are therefore capable of dynamic displays. Any picture displayed on these machines is represented in the computer memory as one or more display files. The display files are programs for the display processor which runs independently of the computer's central processor.

As an initial dynamic display system, a fairly simple package has been implemented which allows up to three separate display files to be handled. File 1, the basic display file, is used for the normal TEKTRONIX simulation. The other two files are similar in operation but may be turned on and off under program control from the central computer. Generally these two files are alternate frames of a dynamic display, one displaying while the other is being built.

In the GT40 computers approximately 8K words are available for display files. At the initiation of an interactive job, all this area is given to file 1, and the other files have no space available. The user has control over the display files by use of the subroutine DWCTLW with calling sequences, (a) to (k).

(a) `CALL DWCTLW (0)`

which 'erases' all three display files. Following data load into file 1.

(b) CALL DWCTLW (1,N2,N3)

allocates space for files 2 and 3. N2 and N3 are the number of bytes (must be even) to be given to files 2 and 3. N2 + N3 must be less than 16000 (there are 2 bytes/word in the GT40). This call must only be made directly after a CALL DWCTLW (0).

(c) CALL DWCTLW (2,±NF) NF = 1,2,3

where +NF erases file NF. Following data load into file NF; -NF homes file NF. Following data add to file NF.

(d) CALL DWCTLW (3,NF) NF = 2,3

turns off file NF.

(e) CALL DWCTLW (4,NF) NF = 2,3

turns on file NF.

(f) CALL DWCTLW (5,NF) NF = 2,3

turns on file NF and turns off file 5-NF (the other of 2 & 3). This is useful for switching frames of a dynamic display.

(g) CALL DWCTLW (6[,IX,IY])

turns on the cursor and optionally defines its position ($0 \leqslant IX \leqslant 1023$; $0 \leqslant IY \leqslant 767$). The cursor is left in a tracking mode - see DWCTLR below.

(h) CALL DWCTLW (7)

turns off the cursor.

(i) CALL DWCTLW (8,IL) IL = 0,1,2,3

selects GT40 hardware line type.

0 - solid line

1 - long dash

2 - short dash

3 - dot dash

(j) CALL DWCTLW (9)

removes the 'sync' from display file 1. The GT40 local software attempts to maintain constant screen intensity by keeping track of how much display file has been generated. The sync (which prevents an excessively bright picture when little information is being displayed) is normally removed by the local software. However when multiple files are being used it is not always possible to calculate total display volume, and flicker may occur. This call overcomes the problem.

(k) CALL DWCTLW (10)

rings the bell in the GT40 keyboard. A further routine DWCTLR is available with a

CALL DWCTLR (IS[,IX,IY])

returning in IS the current content of the GT40 switch register ($-32768 \leqslant IS \leqslant 32767$) and optionally the current cursor location.

Warnings

(i) Unpredictable results can occur if input is requested from the terminal when file 1 is not the current display file.

(ii) Always GPSUSP/XYSUSP before switching display files because the file switching forces character mode. Then GPREST/XYREST before resuming plotting.

9. EXAMPLES

Example 1

This program accepts data of the form:

FUN X

where FUN is the function required (SIN,COS,TAN,EXP or LOG) and X is the required argument.

The program types back the value of FUN(X). If FUN is given as END then the program terminates. The source of the program is shown in Table 2 and terminal output in Table 3.

Example 2

This example demonstrates dynamic graphics. A previously created file contains records of the form:

record 1 50 x values
record 2 50 y values (first set)
record 3 50 y values (second set)
etc.

A 'movie' of y:x will be generated. The program is shown in Table 4. Note that the XYPAPE 'box' is placed into file 1 and is displayed permanently.

Example 3

This program demonstrates the use of the cursor in tracking mode. The cursor is displayed and its trail is drawn if any of the low order console switches are up. Console switch 14 erases the current picture and restarts. Console switch 15 terminates the job. The program is shown in Table 5.

TABLE 2

PROGRAM FOR EXAMPLE 1

```
      INTEGER IFUN(5)/'SIN','COS','TAN','EXP','LOG'/
      INTEGER IEND/'END'/
      CALL XYTLEN('#')                    set terminating character
      CALL DWSTRT                         start interaction
      CALL DWSETU(0,4)                    set unit 4 for output
    1 IC=73                               scan pointer
      CALL SCAN(4,-1,IC,IKEY,1)           get keyword
      IF(IKEY.EQ.IEND) GO TO 900          test for end
      DO 10 I=1,5                         test
      IF(IKEY.EQ.IFUN(I)) GO TO 20        for
   10 CONTINUE                            keyword match
      GO TO 1                             if not
   20 CALL SCAN(2,-1,IC,X,1)              read x
      GO TO (100,200,300,400,500),I       take appropriate service
  100 Y=SIN(X)                            sin(x)
      GO TO 800
  200 Y=COS(X)                            cos(x)
      GO TO 800
  300 Y=TAN(X)                            tan(x)
      GO TO 800
  400 Y=EXP(X)                            exp(x)
      GO TO 800
  500 Y=ALOG(X)                           ln(x)
  800 WRITE(4,801) IFUN(I),X,Y            type result
  801 FORMAT(A3,'(',1PE13.5,')=',E13.5)
      GO TO 1                             continue
  900 CALL DWRITE('END OF RUN#')          type end message
      CALL DWEND                          end interaction
      STOP
      END
```

TABLE 3

TERMINAL OUTPUT FROM EXAMPLE 1

```
$#CONSOLE
ID:
=?
 JOB 147 USRINT    AWAITING EXEC     I PRIO 13 HOLD    2
=$RUN
JOB:USRINT
JOB RELEASED TO START CLASS I
=
START BY 147
=$RUN
JOB:
JOB 147 USRINT    NOW READY (269)


SIN 1.4
SIN(  1.40000E+00) =  9.85450E-01
COS 1.4
COS(  1.40000E+00) =  1.69967E-01
LOG 4.8
LOG(  4.80000E+00) =  1.56862E+00
EXP 1.56862
EXP(  1.56862E+00) =  4.80002E+00
TAN 0.7
TAN(  7.00000E-01) =  8.42288E-01
END
END OF RUN


END OF INTERACTION
=E
END:CONSOLE
```

TABLE 4

PROGRAM FOR EXAMPLE 2

```
      DIMENSION X(50),Y(50)
      CALL XYTLEN('#')                    sets terminating char
      CALL DWSTRT                         start interaction
      CALL DWCTLW(0)                      reset display files
      CALL DWCTLW(1,1000,1000)            allocate space for 2 & 3
      IC=73
      CALL DWRITN('UNIT NUMBER:#')
      CALL SCAN(1,-1,IC,IU,1)
      CALL DWRITN('Y RANGE (MIN&MAX):#')
      CALL SCAN(2,-1,IC,YMN,1)
      CALL SCAN(2,-1,IC,YMX,1)
      READ(IU) X                          read x values
      CALL DWCAN                          erase screen
      CALL XYPAPE(-10,-8,X(1),X(50),YMN,YMX)   initialise plot
      CALL XYSUSP                         suspend plotting
      KF=2                                set file number
    1 CALL DWCTLW(2,KF)                   set the file
      READ(IU,END=2) Y                    read y
      CALL XYREST                         resume plot
      CALL XYLINE(X,Y,50)                 draw y/x
      CALL XYSUSP                         suspend plot
      CALL DWCTLW(5,KF)                   display new frame
      KF=5-KF                             swap file numbers
      GO TO 1                             continue
    2 CALL SWAIT(500)                     5 second delay
      CALL DWCTLW(0)                      reset
      CALL DWCTLW(1,0,0)                  deallocate space
      CALL DWEND                          terminate
      STOP
      END
```

TABLE 5

PROGRAM FOR EXAMPLE 3

```
      DATA IXL,IYL/512,384/,SCALE/68.75/
      CALL DWSTRT                          start up
      CALL GPSEND(1)                       initialise GPLOT
      CALL DWCTLW(6,IXL,IYL)               turn on cursor
    1 IP=3                                 beam off initially
    2 CALL SWAIT(10)                       10/100 second wait
      CALL DWCTLR(KS,IX,IY)                get switch & cursor
      IF(KS.LT.0) GO TO 4                  sw15 implies finish
      IF(KS.GT.16384) GO TO 3              sw14 implies restart
      IF(IX.EQ.IXL.AND.IY.EQ.IYL) GO TO 2   ignore if no move
      IF(KS.EQ.0) GO TO 1                  no trail if no switches
      H=IX/SCALE                           get H
      V=IY/SCALE                           and V
      IXL=IX                               save new pos
      IYL=IY
      CALL GPLOT(H,V,IP)                   draw line
      IP=4                                 set beam on
      CALL GTEND                           empty buffer
      GO TO 2                              continue
    3 CALL GPSUSP                          suspend
      CALL DWCAN                           erase screen
      CALL GPREST                          resume
      GO TO 1                              restart
    4 CALL DWCTLW(7)                       cursor off
      CALL GPSEND(2)                       end plot
      CALL DWEND                           terminate
      STOP
      END
```

10. ACKNOWLEDGEMENTS

The authors thank Dr. D. J. Richardson, Mr. G. R. James and Mr. R. P. Backstrom for their assistance in interfacing this interactive system with the AAEC - HASP - OS - MVT operating system in the IBM 360 computer.

11. REFERENCES

Bennett, N.W. & Pollard, J.P. [1967] - AAEC/TM399.

Cox, G.W., Carey, J.H. & Van Klink, K.H. [1969] - AAEC unpublished report.

Davids, R.E. [1970] - AAEC/TM567.

Ellis, P.J. [1970] - AAEC/E206.

James, G.R. [1976] - AAEC unpublished report.

Johnstone, I.L. [1974] - AAEC unpublished report.

Richardson, D.J. [1969] - *Aust. Computer J.* Vol. 1, No. 5, November.

Richardson, D.J. [1974] - AAEC unpublished report.

Richardson, D.J. [1976] - AAEC unpublished report

Trimble, G.D. [1977] - AAEC/E437.

NOTES

APPENDIX A

DEFAULT TRANSLATION TABLES

EBCDIC TO ASCII TRANSLATE TABLE

		SECOND HEX DIGIT															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
F	0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0
I	1	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0
R	2	A0	A0	A0	A0	A0	8A	8D	A0	A0	A0	A0	A0	A0	A0	A0	A0
S	3	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0
T	4	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	AE	BC	A8	AB	DE
	5	A6	A0	A0	A0	A0	A0	A0	A0	A0	A0	A1	A4	AA	A9	BB	A0
H	6	AD	AF	A0	A0	A0	A0	A0	A0	A0	A0	A0	AC	A5	DF	BE	BF
E	7	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	BA	A3	C0	A7	BD	A2
X	8	A0	E1	E2	E3	E4	E5	E6	E7	E8	E9	A0	A0	A0	A0	A0	A0
	9	A0	EA	EB	EC	ED	EE	EF	F0	F1	F2	A0	A0	A0	A0	A0	A0
D	A	A0	A0	F3	F4	F5	F6	F7	F8	F9	FA	A0	A0	A0	A0	A0	A0
I	B	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0
G	C	A0	C1	C2	C3	C4	C5	C6	C7	C8	C9	A0	A0	A0	A0	A0	A0
I	D	A0	CA	CB	CC	CD	CE	CF	D0	D1	D2	A0	A0	A0	A0	A0	A0
T	E	A0	A0	D3	D4	D5	D6	D7	D8	D9	DA	A0	A0	A0	A0	A0	A0
	F	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	A0	A0	A0	A0	A0	A0

ASCII TO EBCDIC TRANSLATE TABLE

		SECOND HEX DIGIT															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
F	0	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
I	1	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
R	2	40	5A	7F	7B	5B	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
S	3	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	6F
T	4	7C	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
	5	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	40	40	40	4F	6D
H	6	7D	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
E	7	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	40	40	40	40	40
X	8	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
	9	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
D	A	40	5A	7F	7B	5B	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
I	B	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	6F
G	C	7C	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
I	D	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	40	40	40	4F	6D
T	E	7D	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
	F	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	40	40	40	40	40

NOTES

APPENDIX B

DETAILS OF FORTRAN INTERFACE

1. FORTRAN terminal I/O is achieved by intercepting calls by IBCOM# to FIOCS# using the AEREREAD subroutine [Davids 1970]. The DWSETU is an entry which has been added to AEREREAD which activates the intercept for the input and output units and redirects I/O through DWTRAN rather than FIOCS.

2. FORTRAN error monitor messages are redirected to the terminal by a special intercept routine (listed below).

3. These intercepts depend upon the OS360 release because they assume that the special internal linkage between the various FORTRAN run-time libraries is constant.

(continued)

APPENDIX B (continued)

```
//AS      EXEC  ASMFCL,PARM.LKED='LIST,MAP,NCAL,LET'
FIOCSBEP CSECT
         USING FIOCSBEP,R1
         BC    15,TOFIOCS              INITIALLY BRANCH TO FIOCS
         STM   R14,R12,12(R13)         SAVE REGISTERS
         LR    R14,R13                 EXCHANGE
         LA    R13,SAVE                SAVE
         ST    R13,8(,R14)             AREAS
         ST    R14,4(,R13)             ..
         B     CHECK
TOFIOCS  L     R1,=V(FIOCS$$)          FIOCS ERROR ENTRY
         BR    R1                      BRANCH TO FIOCS
         DROP  R1
         USING SAVE,R13
SAVE     DS    18A
*
*        R0=> PARAMS (RETURN ADDRESS IS R0+6)
*
CHECK    DS    0Y
         LR    R1,R0                   GET ARGUMENT ADDRESS
         CLI   0(R1),X'00'             CHECK IF INITIALISING CALL
         BE    INIT                    BRANCH IF SO
         CLI   0(R1),X'02'             CHECK IF A WRITE
         BNE   INIT                    BRANCH IF NOT
         LA    R1,132                  MAXIMUM LENGTH
         LA    R2,BUF+132              END OF BUFFER
DBL      CLI   0(R2),C' '              TEST FOR BLANK
         BNE   DBL2                    BRANCH IF NOT
         BCTR  R2,0                    REDUCE POINTER
         BCT   R1,DBL                  REDUCE COUNT AND CONTINUE
         B     INIT                    BRANCH IF ALL BLANK
DBL2     ST    R1,LEN                  SAVE THE MESSAGE LENGTH
SEND     LA    R1,DWCON                CARRIAGE CONTROL LIST
         L     R15,=V(DWTRAN)          SEND CARRIAGE CONTROL
         BALR  R14,R15                 ..
         C     R15,=A(4)               TEST FOR UNIT EXCEPTION
         BE    SEND                    RETRY IF SO
         LA    R1,DWARG                MESSAGE LIST
TDO      DS    0Y
         L     R15,=V(DWTRAN)          SEND MESSAGE
         BALR  R14,R15                 ..
         C     R15,=A(4)               TEST FOR UNIT EXCEPTION
         BE    TRAN                    BRANCH IF SO
INIT     LA    R2,BUF                  RESET BUFFER POINTERS
         LA    R3,133                  FOR ERROR MONITOR
         MVI   BUF,C' '                BLANK OUT
         MVC   BUF+1(132),BUF          BUFFER AREA
         L     R13,4(,R13)             RESTORE R13
         LM    R14,R0,12(R13)          LOAD REGISTERS
         LM    R4,R12,36(R13)          ..
         LR    R1,R0                   RESET R1
         B     6(,R1)                  RETURN
TRAN     LA    R1,DWARGNO              NO TRANSLATE LIST
         B     TDO                     RESEND
         DROP  R13
*
```

(continued)

APPENDIX B (continued)

```
          ENTRY DWERMS
          USING DWERMS,R15
DWERMS    L     R15,=A(FIOCSBEP)
          USING FIOCSBEP,R15
          L     R1,0(,R1)                ARGUMENT ADDRESS
          CLC   0(2,R1),=CL2'ON'         TEST IF 'ON'
          BNE   STOF                     BRANCH IF NOT
          MVI   FIOCSBEP+1,X'00'         SET NO BRANCH
STOF      CLC   0(3,R1),=CL3'OFF'        TEST IF 'OFF'
          BCR   7,R14                    RETURN IF NOT
          MVI   FIOCSBEP+1,X'F0'         SET BRANCH
          BR    R14                      RETURN
*
DWCON     DC    0A(0),A(DWCR,DWCR,DWW),X'80',AL3(ZIL)
DWARG     DC    0A(0),A(BUF+1,LEN,DWW),X'80',AL3(ONE)
DWARGNO   DC    0A(0),A(BUF+1,LEN,DWW),X'80',AL3(ZIL)
LEN       DC    A(132)
ONE       DC    A(1)
ZIL       DC    A(0)
DWCR      DC    A(0)                     DUMMY BUF FOR CRLF
BUF       DC    CL133' '
DWW       DC    XL1'05'                  WRITE 5 (TRAILING CRLF)
          LTORG
          REGS
          END
//LKED.SYSLMOD DD DSN=PHYS.FORTLIB,DISP=SHR
//LKED.SYSIN DD *
 CHANGE FIOCSBEP(FIOCS$$)
 INCLUDE SYSLIB(IHCEFIOS)
 NAME DWERMS(R)
/*
//LKED.SYSLIB DD DSN=SYS1.FORTLIB,DISP=SHR
//
```

NOTES