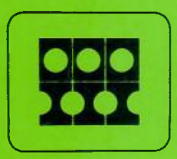

INFORMATION INTERNATIONAL®

# FR 80/COMp 80
# MONITOR SYSTEM



PUB. NO. 90441

COPY NO. 96

TABLE OF CONTENTS

# Chapter 1

## CREATING NEW SOFTWARE

1.1 The Information International Image Processing Controller comes with a powerful programming system for both tape and disk operating configurations. The first step in preparing a new program is the editing and updating of the source symbolics. These symbolics consist of standard subroutines and the application package utilizing all the necessary routines to perform some specific tasks. The user prepares a symbolic file using the Symbolic Text Editor and has either a disk image or a paper tape of the symbolics, depending on the machine configuration. This file can be modified and updated using the text editor.

1.2 Next is the assembly process. A two-pass macro-assembler for processing the source file and producing a disk or paper tape binary program is utilized by the system. At this point, the user can add the new program to his library of existing programs either on disk or magnetic tape.

1.3 To provide the user with this programming capability, Information International supplies the source to produce custom application programs. These subroutines include:

1. Operating monitor
2. Vector routines
3. Character sets
4. Character routines
5. Magnetic tape routines

Given these basic routines, the user can put together an application package with a minimum of programming effort. Detailed documentation of these routines are included in the Appendices.

### STANDARD SUBROUTINES

1.4 Standard Subroutine Parameters (III109) - This file is a set of definitions, including:

1. Machine configuration (core, tape, disk)
2. Types of cameras available
3. Special options
4. Macro definitions for standard programming conventions

Operating Monitor

1.5 The purpose of the MONITOR is to interface between the human operator and the application programs. The MONITOR is comprised of III166-series programs and III161-series programs. Included in MONITOR are the Teletype input/output routines and basic hardware functions which follow:

1. Beam positioning
2. Beam parameters
3. Camera advance
4. Magnetic tape I/O
   a. Density select
   b. Drive select
   c. Space forward
   d. Space backward
   e. Error Retry
5. Teletype command decoding
6. Error responses
7. Image rotation
8. Focus pattern
9. Program load and dump

1.6 The Disk I/O file, III183, contains routines to create, read, and write files formatted within the disk operating structure.

Vector Routines (III162)

1.7 This file contains a general set of vector drawing routines. After calling the subroutine which sets head and tail coordinates of a vector, one of three vector routines may be called:

1. Solid vector
2. Dotted vector
3. Dashed vector

Character Sets

1.8 Character Dispatch Tables (III164) - This file contains dispatch tables for individual character sets. The set is defined by using conditional assembly features. Some of the sets available are:

1. III master set
2. BCD
3. EBCDIC (including lower case)

1.9 Character Descriptors (III164 FILM) - This file contains the character descriptors for the character generator hardware. There are two other

fonts available from Information International: the NMA Microfont (III164 MICRO) and OCR-B (III164 OCR-B).

1,10 Character Routines (III147) - This file contains routines to accept magnetic tape characters and convert them into FR 80 internal codes by using III164; the routine then performs the I/O instruction to start the character generator plotting the specified character.

1,11 Magnetic Tape Routines (III163) - This file contains routines to read 7- or 9-track tape in binary or character data. The data on tape can also be accessed in bit or word formats. This file also contains routines to reposition the tape for processing nested repeats of data.

# Chapter 2

## USE OF MONITOR ROUTINES

2.1 Basically, applications programs consist of:

1. Assembly parameters
2. Macro Definitions
3. MONITOR
4. Functional subroutines as required by specific programs (vectors, mag tape I/O, etc.)
5. Coding required for the specific application
6. Buffers, (disk, tape) - may include expendable code.

2.2 The sample program which follows demonstrates the use of MONITOR routines as well as the structure of a typical application program. The use of INSERT files and allocation of buffer area is also demonstrated.

2.3 The program reads a 9-track core dump tape containing vector end points (XHD,YHD,XTL,YTL) and frame advance information. A record consists of 256. words. If the first 4 bits of an 18 bit data word are all zeros, the low order 14 bits are XHD. The next 3 words contain YHD,XTL, and YTL in the same format. If the first 4 bits are all ones, then the word is a frame advance command.

```
/SAMPLE                     27 MAR 1974

MUMBLE==1                   /ENABLES TTY OUTPUT OF HARDWARE
                            /AND SOFTWARE CONFIGURATIONS
                            /DURING ASSEMBLY
FONT==0                     /USE FILM FONT
IIISET==1                   /USE III CHARACTER SET
BIGBUF==0                   /BIGBUF==1 GIVES LARGER BUFFER BUT FEWER
                            /FEATURES
.INSERT III109              /INSERT STANDARD SUBROUTINE PARAMETERS

.IFD DISC,DSKMON==1
.IFND DISC,TAPMON==1
MTPTR==16                   /USE AUTOINDEX REGISTER IN MAG TAPE
                            /ROUTINES
.INSERT III166              /MONITOR ROUTINES
MDSEND/                     /SETS LOCATION COUNTER TO THE END
                            /OF THE MONITOR COMMAND TABLE
.ASCII  /SET SIZE?/          JMP I SCALIN
.ASCII  /X,Y OFFSET?/        JMP I SETOFS
MDSEND==.       0           /END OF MONITOR DISPATCH TABLE
.INSERT III164              /CHARACTER SET AND FONT
.INSERT III162              /VECTOR ROUTINES
.INSERT III163              /MAG TAPE ROUTINES
.INSERT III147              /CHARACTER PLOTTER

HEREIS==$.                  /$ IS EXTENDED ADDRESS INDICATOR
                            /SAVES PROGRAM COUNTER
DATCOM/160200               /MAG TAPE COMMAND WORD (TAPE
                            /TYPE ETC)
.TEXT 306                   /SETS UP MODE FOR .ASCII
PGNAME/.ASCII /SAMPLE?/     /PROGRAM NAME
MTTBUF/
.TEXT 106
.ASCII "/SAMPLE"            /NAME TO BE USED ON SYSTEM
                            /TAPE INDEX
0
HEREIS/

BEGIN,
.IFNZ BIGBUF,DZM PDSTRT LAC ($PROGND-1) /MAKE EXPENDABLE
                            /OPERATOR COMMANDS INACCESSABLE
.IFZ BIGBUF,LAC ($EXPND-1)
.IFND TWOBUF,DAC MTAREA     /SINGLE BUFFER MONITOR CELL
.IFD TWOBUF,DAC CURBUF#     /DOUBLE BUFFER MONITOR CELL
          CMA
          DAC PBUFSZ        /SET UP DATA BUFFER SIZE
.IFD TWOBUF,CMA ADD CURBUF      DAC NEXBUF#  /SET
                            /UP BUFFER POINTERS
          JMS MTRINI        /INITIALIZE MAG TAPE ROUTINES (III163)
          CLC
          DAC DRWVEC        /FORCE FIRST PAGE EJECT (III162)
```

```
GET1,     JMS KYBLIS        /LISTEN FOR KEYBOARD INTERRUPT (III166)
          JMS MTLAC         /RETURN 18-BIT WORD IN AC (III163)
          DAC XHD#          /# ASSIGNS STORAGE LOCATION
          LMQ
          LLS 1004
          SNA               /HEAD & TAIL INFORMATION?
          JMP DOHDS
          SAD (17)          /FRAME ADVANCE?
          JMP NXADV
          JMP FORMAT        /FORMAT ERROR ROUTINE (III166)

DOHDS,
          LAC XHD
          MAPX
          DAC XHD
          JMS MTLAC         /GET Y COORD
          MAPY
          DAC YHD#          /STORE Y COORD
          JMS SETHDS        /SUBROUTINE TO SET HEAD AND TAIL
                            /COORDINATES (III162)
          LAC XHD
          LAC YHD

GET2,     JMS MTLAC
          MAPX
          DAC XTL#
          JMS MTLAC
          MAPY
          DAC YTL#
          SETTL XTL,YTL     /SET TAIL COORDINATES
          JMS DRWVEC        /PRODUCE SOLID VECTOR
          JMP GET1

NXADV,    LAC DRWVEC
          SZA
          NEXPIC            /ADVANCE CAMERA TO NEXT PAGE
          DZM DRWVEC
          JMP GET1
ENTRY MAPX                  /CONVERT TO III SCOPE COORDS
          MAP
          ADD XOFF
          EXIT MAPX
XOFF,     0                 /SET INITIAL VALUE TO ZERO

ENTRY MAPY
          MAP
          ADD YOFF
          EXIT MAPY

YOFF,     0

ENTRY MAP                    /SCALE X & Y
MAPSW,    NOP
```

```
          EXIT MAP
          MUL I
MAPMUL,   16384.          /SET SIZE
          DIV
MAPDIV,   16384.
          LACQ
          ADD MAPOFF
          EXIT MAP
MAPOFF,   0

.IFNZ BIGBUF,CONSTANTS    VARIABLES /ASSIGNS CURRENTLY
                          /UNASSIGNED CONSTANTS AND VARIABLES
PROGND,                   /REQUIRED LABEL - "END OF PROGRAM"
.INSERT III161 GO

.IFNZ BIGBUF,CONSTANTS    VARIABLES
GOND,                     /REQUIRED LABEL - "END OF GO
                          /EXPENDABLES"
.INSERT III161

SCALIN,   DAC TYPSCL      /POINT TO PARAMETER TYPER
          JMS GETNUM      /CONVERT DECIMAL NUMBER FROM
                          /TELETYPE TO BINARY (III166)
          LAC DECNUM      /DECIMAL NUMBER IN AC
          DAC MAPMUL
          CLL
          SAD (16384.)
          STL
          LAC SKPCON
          SZL
          LAC OPRCON
          DAC MAPSW
          LAC (1-16384.)
          TAD MAPMUL
          TCA
          LRSS 1
          DAC MAPOFF
          JMP MDONE       /WAIT FOR NEXT OPERATOR COMMAND
                          /(III166)
TYPSCL,   XX
          LAC MAPMUL
          JMS MDOUT       /OUTPUT AC IN DECIMAL
          JMP I TYPSCL
SETOFS,   DAC TYPOFS      /POINT TO PARAMETER TYPER
          JMS GETANM      /GETS NUMBERS SEPARATED BY
                          /COMMAS
          DAC XOFF
          JMS GET1NM      /MONITOR ROUTINE (III161)
          DAC YOFF
          JMP MDONE

TYPOFS,   XX
          LAC OPRCON
```

```
        JMS TYPOUT      /MONITOR ROUTINE (III166)
        LAC XOFF
        LAC YOFF
        JMP I TYPOFS

CONSTANTS    VARIABLES
EXPND,                  /REQUIRED LABEL - "END OF ALL
                        /EXPENDABLES"
.IFM 8192.-$.,.ERROR /TOO MUCH CODE FOR FIRST 8K
/

IFZ BIGBUF,HEREIS==BUFFND-EXPND-1 /BUFFND IS DEFINED
                        /BY MONITOR TO BE THE END OF
                        /AVAILABLE CORE
.IFNZ BIGBUF,HEREIS==BUFFND-PROGND-1
.IFD TWOBUF,HEREIS==HEREIS:2
PBUFSZ/-HEREIS+1
.IFD MUMBLE,[
.PRINT /BUFFER SIZE IS /
.VALD HEREIS.           /HEREIS IS EQUAL TO COMPUTED
                        /BUFFER SIZE
.PRINT / WORDS (/       /(SEE ASSEMBLER DOCUMENTATION
                        /FOR .PRINT PSEUDO-OP)
.VALD HEREIS*3
.PRINT / OR /
.VALD HEREIS*2
.PRINT / BYTES)
/
]

START
```

# Chapter 3

## USE OF STANDARD SUBROUTINES ON A NON-DISK MACHINE

3.1 The use of standard subroutines greatly simplifies the writing of programs for non-disk machines. However, in symbolic form, these routines are too lengthy for paper tape. The usual practice therefore, is to use the host computer to build a magnetic tape for the paper tape assembler - the binary tape which results is not unreasonably long.

3.2 SYSTEM TAPE DEBUG has been assembled incorporating MONITOR subroutines. The symbols for these routines are available and may be utilized to create new System Tape programs. This Debug and Monitor system has a dump command to enable the user to dump his program on the System Tape after debugging it.

3.3 In order to write the sample program which follows, it is first necessary to determine the value of the MONITOR symbols required by the program. These values are determined by loading DEBUG from the System Tape and typing the symbol name followed by a colon (See DEBUG documentation). The value is returned on the teletype and a list of symbols is thereby created which may be incorporated in the user program through use of the Editor.

3.4 Upon completion of editing, a paper tape of the symbolic program is punched (See EDITOR documentation). The program may now be assembled by loading the Assembler from the System tape and following the instructions for Paper Tape Assembly (See ASSEMBLER documentation). Errors detected during the assembly process may be corrected by returning to the Editor and revising the symbolic program (this necessitates the punching of a new symbolic tape) or by patching the program directly using the commands available to DEBUG.

3.5 The binary produced by the Assembler may be loaded and run by DEBUG or the program dumped on the end of the System Tape and run in the usual manner.

3.6 Since DEBUG can process symbolic instructions (including calls to MONITOR routines), it is feasible to type small programs directly into core. The first location available for use by a user program is AVAIL. Use may be made of DEBUG's symbol definition capability to handle the forward reference problem.

```
/SAMPLE                                          APRIL 2, 1974

MUMBLE==1

/MONITOR SYMBOLS
BIGBUF==1
MONTOR==40        /STARTING ADDRESS FOR MONITOR
MDONE==57         /RETURN POINT FOR CORRECT PARAMETER ENTRY
SKPCON==346
OPRCON==743
MTAREA==1030      /CONTAINS POINTER TO TAPE BUFFER AREA
PBUFSZ==1032      /CONTAINS TWO'S COMPLEMENT OF BUFFER LENGTH
DATCOM==1033      /CONTAINS MAG TAPE SELECT WORD
PGNAME==1037      /STARTING ADDRESS OF ASCII PROGRAM NAME
MTTBUF==1065      /STARTING ADDRESS OF ASCII PROGRAM NAME ON TAPE
MDOUT==1356       /NUMBER OUTPUT ROUTINE
GETNUM==1433      /PARAMETER NUMBER INPUT ROUTINE
FORMAT==1517      /RETURN POINT FOR BAD PARAMETER ENTRY
TYPOUT==1526
ADVAN=JMS 1672    /FILM ADVANCE ROUTINE
KYBLIS==2043      /KEYBOARD LISTEN ROUTINE
PSTART==2234      /WHERE CONTROL GOES WHEN "GO/" IS TYPED TO MONITOR
DECNUM==2304
PDSTRT==2460
MDSEND==2511      /30. CELLS ARE PROVIDED FOR USER DISPATCHES
GET1NM==2550      /ROUTINE TO GET EXACTLY ONE PARAMETER
GETANM==2555      /ROUTINE TO GET A PARAMETER IF IT EXISTS
AVAIL==10042      /LOWEST LOCATION FOR USER PROGRAM

DEBUG==20000-1-100      /ENTRY POINT FOR DEBUG
MEMEND==DEBUG-4000      /ALLOW ROOM FOR SOME SYMBOLS

/MAG TAPE ROUTINE SYMBOLS
MTRINI==3573      /INITIALIZE MAG TAPE INPUT
MTLAC==4013       /GET A WORD FROM MAG TAPE

/VECTOR ROUTINE SYMBOLS
DRWVEC==3536      /ROUTINE TO DRAW A VECTOR
SETHDS==3464
SETTLS==3511

/VECTOR MACROS

.DEF SETHD X,Y
JMS SETHDS
LAC X
LAC Y
.TERM

.DEF SETTL X,Y
JMS SETTLS
LAC X
LAC Y
```

```
.TERM


MDSEND/                 /SETS LOCATION COUNTER TO THE
                        /END OF THE MONITOR COMMAND TABLE

.TEXT 306                /SETS UP MODE FOR .ASCII
.ASCII  /SET SIZE?/     JMP I SCALIN
.ASCII  /X,Y OFFSET?/   JMP I SETOFS

MDSEND==.       0       /END OF MONITOR DISPATCH TABLE

PSTART/JMP BEGIN
DATCOM/160200           /MAG TAPE COMMAND WORD (TAPE
                        /TYPE ETC)

PGNAME/.ASCII /SAMPLE?/ /PROGRAM NAME
MTTBUF/
.TEXT 106
.ASCII "/SAMPLE"        /NAME TO BE USED ON SYSTEM
                        /TAPE INDEX
0
AVAIL/                  /FIRST LOCATION USED BY DEBUG
                        /FOR USER PROGRAM

BEGIN,
.IFNZ BIGBUF,DZM PDSTRT LAC ($PROGND-1)  /MAKE EXPENDABLE
                        /OPERATOR COMMANDDS INACCESSABLE
.IFZ BIGBUF,LAC ($EXPND-1)
        DAC MTAREA      /SINGLE BUFFER MONITOR CELL
        CMA
        DAC PBUFSZ      /SET UP DATA BUFFER SIZE
        JMS MTRINI      /INITIALIZE MAG TAPE ROUTINES
        ISZ DRWVEC      /FIRST PAGE EJECT
GET1,   JMS KYBLIS      /LISTEN FOR KEYBOARD INTERRUPT
        JMS MTLAC       /RETURN 18-BIT WORD IN AC
        DAC XHD#        /# ASSIGNS STORAGE LOCATION
        LMQ
        LLS 1004
        SNA             /HEAD AND TAIL INFORMATION?
        JMP DOHDS
        SAD (17)        /FRAME ADVANCE?
        JMP NXADV
        JMP FORMAT      /FORMAT ERROR ROUTINE

DOHDS,
        LAC XHD
        MAPX
        DAC XHD
        JMS MTLAC       /GET Y COORD
        MAPY
        DAC YHD#        /STORE Y COORD
        JMS SETHDS      /SUBROUTINE TO SET HEAD AND TAIL
```

```
                        /COORDINATES
        LAC XHD
        LAC YHD

GET2,   JMS MTLAC
        MAPX
        DAC XTL#
        JMS MTLAC
        MAPY
        DAC YTL#
        SETTL XTL,YTL   /SET TAIL COORDINATES
        JMS DRWVEC      /PRODUCE SOLID VECTOR
        JMP GET1

NXADV,  LAC DRWVEC
        SZA
        ADVAN           /ADVANCE CAMERA TO NEXT PAGE
        DZM DRWVEC
        JMP GET1
ENTRY MAPX              /CONVERT TO III SCOPE COORDS
        MAP
        ADD XOFF
        EXIT MAPX
XOFF,   0               /SET INITIAL VALUE TO ZERO

ENTRY MAPY              /SCALE X AND Y
        MAP
        ADD YOFF
        EXIT MAPY
YOFF,   0

ENTRY MAP
MAPSW,  NOP
        EXIT MAP
        MUL I
MAPMUL, 16384.          /SET SIZE
        DIV
MAPDIV, 16384.
        LACQ
        ADD MAPOFF
        EXIT MAP
MAPOFF, 0

.IFNZ BIGBUF,CONSTANTS VARIABLES /ASSIGNS CURRENTLY
                        /UNASSIGNED CONSTANTS AND VARIABLES
PROGND,                 /REQUIRED LABEL - "END OF PROGRAM"

.IFNZ BIGBUF,CONSTANTS VARIABLES
GOND,                   /REQUIRED LABEL - "END OF GO
                        /EXPENDABLES"
```

```
SCALIN, DAC TYPSCL      /POINT TO PARAMETER TYPER
        JMS GETNUM      /CONVERT DECIMAL NUMBER FROM
                        /TELETYPE TO BINARY (III166)
        LAC DECNUM      /DECIMAL NUMBER IN AC
        DAC MAPMUL
        CLL
        SAD (16384.)
        STL
        LAC SKPCON
        SZL
        LAC OPRCON
        DAC MAPSW
        LAC (1-16384.)
        TAD MAPMUL
        CMA
        TAD (1)
        LRSS 1
        DAC MAPOFF
        JMP MDONE       /WAIT FOR NEXT OPERATOR COMMAND (III166)
TYPSCL, XX
        LAC MAPMUL
        JMS MDOUT       /OUTPUT AC IN DECIMAL
        JMP I TYPSCL
SETOFS, DAC TYPOFS      /POINT TO PARAMETER TYPER
        JMS GETANM      /GETS NUMBERS SEPARATED BY COMMAS
        DAC XOFF
        JMS GET1NM      /MONITOR ROUTINE (III161)
        DAC YOFF
        JMP MDONE

TYPOFS, XX
        LAC OPRCON
        JMS TYPOUT      /MONITOR ROUTINE (III166)
        LAC XOFF
        LAC YOFF
        JMP I TYPOFS

CONSTANTS       VARIABLES
EXPND,                  /REQUIRED LABEL - "END OF ALL
                        /EXPENDABLES"
.IFM MEMEND-.$,.ERROR /TOO MUCH CODE FOR FIRST 8K
/
BUFFND==MEMEND

.IFZ BIGBUF,HEREIS==BUFFND-EXPND-1
.IFNZ BIGBUF,HEREIS==BUFFND-PROGND-1
PBUFSZ/-HEREIS+1

START MONTOR
```

STANDARD SUBROUTINE PARAMETERS (III109)

March 27, 1974

Published by
Information International
12435 West Olympic Boulevard
Los Angeles, California 90064

# STANDARD SUBROUTINE PARAMETERS (III109)

## ABSTRACT

III109 contains subroutine parameters that specify the hardware and software configuration at assembly time.

## USAGE

The first page of III109 contains symbol definitions for all hardware configurations, e.g., processor type, memory size, peripheral I/O devices, vector and character generators, and cameras. A desired configuration is selected by choosing which symbols to define and their value.

The remainder of III109 consists of software configuration definitions, e.g., location of DEBUG swapper, macro definitions, definitions for IOT instructions, most of which are dependent upon the hardware configuration selected on the first page.

The set of symbol and macro definitions that result from III109 are used in other standard Information International subroutines to make them independent of configuration. Applications programs then should insert III109 before any other subroutine inserts; common practice is to insert it before the first executable coding, to allow full use of the macros.

The symbol definitions on the first page of III109 are organized into groups according to the feature they describe. All symbols for machine types, e.g., PDP15, III15, are grouped together with only one definition allowed; the others are preceded by a slash (/), causing the assembler to ignore the line. Changing the machine type, then, can be accomplished by inserting a slash before the previously selected option, and removing a slash from the definition for the required option.

Some symbols are always defined, and their value is selected according to configuration. The BANKS parameter, for example, is used to select memory size.

## HARDWARE CONFIGURATION

### MACHINE

| | |
|---|---|
| FR80 | If defined, machine is an FR 80 |
| COMp80 | If defined, machine is a COMp 80 |
| PFR | If defined, machine is a PFR |
| EYE | If defined, machine is a COMPUTER EYE |
| PFS | If defined, machine is a PFS |

STANDARD SUBROUTINE PARAMETERS (III109)
Hardware Configuration

CPU TYPE

| | |
|---|---|
| PDP7 | If defined, assemble for PDP-7 |
| PDP9 | If defined, assemble for PDP-9 |
| PDP9L | If defined, assemble for PDP-9L |
| PDP15 | If defined, assemble for PDP-15 |
| III15 | If defined, assemble for III-15 |

MEMORY SIZE

| | |
|---|---|
| BANKS | Number of 4K memory banks present. |

PERIPHERALS

| | |
|---|---|
| 7TRACK | A value of zero means a 7-track tape drive is not available; a value of one means 7-track is available. |
| 9TRACK | A value of zero means 9-track drive is not available; one means available. |
| DECTAP | If defined, micro tapes available. |
| DISC | If defined, disc available; set equal to number of discs (1 or 2). |
| LPRNTR | If defined, line printer available. |
| CARDRD | If defined, card reader available. |

CHARACTER GENERATORS

| | |
|---|---|
| HSCHAR | If defined, high-speed character option present |
| GACHAR | If defined, graphic arts character generator present |
| HRSPAC | If defined, high-resolution spacing option present |

CAMERAS

| | |
|---|---|
| CAMNUM | Indicates type of camera to be used: |

= 1 35UN
= 2 16UN
= 3 35SP
= 4 16SP
= 5 16mm Strip Fiche
= 7 Fiche (Mod 1 Controller)
= 8 16mm Fiche (Mod 1 Controller)
= 9 Fiche (Mod 2 Controller)
= 10 16mm Fiche (Mod 2 Controller)
= 11 Hardcopy
= 13 Hardcopy Without Vacuum Control
= 14 5 Inch
= 15 70mm
= 16 Full Size Hardcopy

NUMCAM — Indicates number of cameras to be used. If zero, no camera will be used; if greater than one, define alternate cameras with CAM macro:

CAM n,m,o,...
when n, m, o, etc., are alternate camera numbers

## SOFTWARE CONFIGURATION

III109 defines a set of symbols for software configuration based on the hardware configuration chosen on the first page. These symbols can be used by programs to make them machine independent.

### DISK DEBUG

| | |
|---|---|
| MEMSIZ | Memory size in 18-bit words. |
| DEBUG | Memory location of entry point for DEBUG swapper. |
| SWAPSZ | Number of words reserved for the DEBUG swapper (at top of memory). |
| MEMEND | Memory location of the last free word before the beginning of the DEBUG swapper. |
| DBGINT | Defined as a jump to the appropriate location to enter the DEBUG swapper. |

### MULTIPLE CORE CONVENTIONS

Standard subroutines use bank mode addressing, with a 13-bit address for memory reference instructions. This allows direct reference to a maximum of 8192 locations; locations beyond this range must be referenced indirectly. For programs written to assemble for both 16K and 8K memory sizes, any indirect addressing used for the 16K version is unnecessary for 8K. A set of symbols are permanently defined in the assembler which use indirect addressing only if the instruction and the memory cell are referenced are in different core banks.

| | | | |
|---|---|---|---|
| LACI | XORI | TADI | JMPI |
| ADDI | DACI | ISZI | JMSI |
| ANDI | DZMI | SADI | XCTI |

Each of the above represents a memory reference instruction code with an "I" appended. For memory sizes of 8K or less, they are defined identically to their counterparts; otherwise they are defined as pseudo-ops that accomplish the same result through indirect addressing. For example:

```
LACI X becomes LAC X for 8K or less
               LAC I (if X in different core bank
```

Two other instructions are supplied similarly in two versions:

```
LACA X = LAW X 8K or less
       = LAC (X more than 8K

SADA X = SAD (LAW X 8K or less
       = SAD (X more than 8K
```

## NUMBER PRINTING MACROS

The following macros will print the value of an expression on the Teletype during program assembly:

```
.VAL X  - Print the value of X in octal
.VALD X - Print the value of X in decimal,
          followed by a period (.)
```

## ASSEMBLY TYPEOUTS

If the symbol "MUMBLE" is defined, III109 will type to the Teletype, during program assembly, a description of the hardware and software configuration chosen.

## MACHINE IOTS

III109 defines symbols for all those Input-Output Transfer (IOT) instructions that are not machine independent.

INFORMATION INTERNATIONAL MONITOR SYSTEM (III166)

August 7, 1973

Published by

Informational International
12435 West Olympic Boulevard
Los Angeles, California 90064

# MONITOR SYSTEM

## INTRODUCTION

The MONITOR is normally a part of application programs. It creates the monitor display of commands, communicates with the operator, and executes routines after interpreting valid operator commands. Also included in the Monitor are general purpose routines that simplify FR 80 application programming and handle special functions.

The files III161, III161 GO, III166, III166 INVAR, III166 SPEC, III166 ADVAN, and III166 TABLE make up the total MONITOR. The only files that require insertion by the application program are III161 and III161 GO. III161 inserts III166 and III166 inserts the other files. These files may be inserted at other points in the application program if assembly parameters are to be changed during assembly or any of the files are expendable at the time of program execution.

The user should verify the first file to be inserted is III109 and the second is III166. The first cell of executable user code should be at location BEGIN as the MONITOR jumps to this location after recognizing a "GO" command.

USAGE

ADDITION OF MONITOR OPERATOR COMMANDS

1. Set the location counter to the address of cell MDSEND (located in III166).

2. Pack the characters to appear in the command three to a word, terminated by a question mark. Typically this can be accomplished with the use of either the A macro or the .ASCII pseudo-op.

3. Follow the text with a JMP to the appropriate command handler. The indirect bit in this JMP, if on, is a flag to MONITOR to also process the subroutine whose address is contained in the first cell of the command handler. Typically this second subroutine is used to display the command parameters on the monitor screen. Each command handler routine is terminated by a "JMP MDONEX". If the JMP to the Command Handler was flagged by double "I"s (i.e., JMP II---) then the command parameters will appear only if the assembly parameter BIGBUF is defined as being equal to zero.

4. Terminate the new command list with a zero word and set the parameter MDSEND equal to the current location.

The following are subroutines included in the monitor and are available to the user to simplify programming:

GETNUM:

Converts octal and decimal numbers from the teletype buffer to binary. Address 11(octal) contains the address of the teletype buffer -1. The character stream up to a non-numeric is converted and the subroutine exits with the non-numeric that terminated conversion in the AC and MQ; the binary representation of the octal number is in cell OCTNUM, the decimal number in cell DECNUM.
Calling sequence - JMS GETNUM

GETANM:

GETANM can be used for MONITOR command handling routines. It uses GETNUM to retrieve decimal numbers separated by commas. After the last number is retrieved, the instruction following the "JMS GETANM" is executed unless nothing was typed for the last entry.
Calling sequence - JMS GETANM  leaves number in AC
Any instruction

MTTOUT:

Displays the character whose teletype code is contained in the AC on either the monitor screen or the teletype depending on the mode.
Calling sequence - JMS MTTOUT

MCRLF:

Outputs a carriage return and line feed to either the monitor screen or the teletype depending on the mode.
Calling sequence -- JMS MCRLF

MMESSG:

Outputs a message to either the monitor screen or teletype depending on the mode. The message should be packed three 6-bit ASCII characters to a word and terminated by a question mark. The subroutine is entered with the address of the message -1 in the AC.
Calling sequence - JMS MMESSG

PSTLL:

Remains in a two instruction loop until the character, vector and point plotting generators are not busy.
Calling sequence - PSTLL

LROTAT:

Executes an LROT instruction using the value contained in the AC plus the value of FONTYP.
Calling sequence - LROTAT

MNLSIZ:

Executes an LSIZ instruction using the character size value contained in the AC.
Calling sequence - MNLSIZ

MNBRIT:

Executes an LBRT instruction using the value contained in the AC.
Calling sequence - MNBRIT

MNLSPS:

Loads the spot size register by executing an LSPS instruction using the value contained in the low order three bits of the AC.
Calling sequence - MNLSPS

MNSPOT:

Calls subroutine MNLSPS and if the PLS is on and the spot size is changed, a fifty millisecond delay is initiated.
Calling sequence - MNSPOT

MNSCOL:

This subroutine is used to select a color filter. Bits 14 - 17 of the AC specify the filter code:

| BIT | ADDITIVE SYSTEM | SUBTRACTIVE SYSTEM |
|---|---|---|
| 17 | CLEAR | CLEAR |
| 16 | RED | MAGENTA |
| 15 | GREEN | YELLOW |
| 14 | BLUE | CYAN |

Any "on" bits cause the corresponding filters to be inserted; "off" bits remove the filters.
Calling sequence - LACC n (filter code)
MNSCOL

MNLCDY:

Loads the character delta-Y by executing a LCDY instruction using the value contained in the AC.
Calling sequence - MNLCDY

MNLCDX:

Loads the character delta-X by executing an LCDX instruction using the value contained in the AC.
Calling sequence - MNLCDX

ROTTST:

Executes the instruction immediately following the call to ROTTST if the recording mode is comic (upright). If the recording mode is cine the second instruction following the call to ROTTST is executed. Processing is continued with the third instruction following the call to ROTTST.
Calling sequence - JMS ROTTST
Any Instruction
Any Instruction

```
EXAMPLE:  JMS ROTTST
          LAC XDELTA (Execute if in comic mode)
          LAC YDELTA (Execute if in cine mode)
          DAC DELTA  (Processing continues here)
```

ROTATE:

Rotates the image depending on the value (0-3=counterclockwise - 0=uprght) contained in the AC. The corresponding value (20=comic, 12=cine) is stored in cell CHRROT (for character rotation).
Calling sequence - LACC n
JMS ROTATE

SETPLS:

This subroutine calls SETUMU and unblanks the high precision CRT if PLSON contains a NOP (MAKE FILM = 1).
Calling sequence - SETPLS

SETOMU:

Calls subroutine PSTLL, executes a RST instruction and using the values contained in the following cells either calls the corresponding subroutine or executes the corresponding instruction to load a particular FR 80 register.
Calling sequence - SETOMU

| CELL | - INSTRUCTION/SUBROUTINE |
|---|---|
| LRAWRD | LRA |
| HSLBWD | HSLB |
| RECSPT | MNSPOT |
| RECPIN | MNBRIT |
| CHRROT | LROTAT |
| CHRSIZ | MNLSIZ |
| CHDELX | MNLCDX |
| CHDELY | MNLCDY |
| RECCOL | MNSCOL |

SETXY, SETXYF, SETXYS:

These routines load the X and Y DACs using the values obtained by executing the instruction immediately following the subroutine call for the X value and the next instruction for the Y value. SETXY issues no delay after setting the DACs, SETXYF issues a 30 microsecond delay while SETXYS allows a 120 microsecond delay. The routines perform the coordinate transformation for the selected rotation so the programmer is only required to supply the values for comic mode regardless of actual rotation.
Calling sequence - JMS SETXY or SETXYF or SETXYS
Any instruction
Any instruction
EXAMPLE: JMS SETXY

```
LAC XVALUE
LAC YVALUE
```

INTENS:

Calls subroutine PSTLL and executes an INTS instruction causing intensification of the current point on the monitor and on the PLS if enabled. The current values for spot size and intensity are used and the ones complement of the number of hits is stored in cell PTHITS.
Calling sequence - INTENS

MNOUT:

Converts the contents of the AC to the specified radix and outputs the number to the teletype. If BIGBUF = 0, the next instruction must be a LACC n where n is the radix number.

The maximum number of digits that may be output is 6 and if leading zeros are to be printed, MDOUT4+3 should be changed to NOP. Further, if an unsigned number is to be output, MNOUT+1 should be changed to CLL.
Calling sequence - JMS MNOUT
LAC XXXX (this instruction should load the accumulator with the radix if BIGBUF = 0)

MOOUT:

Calls MNOUT to output the contents of the AC in octal.
Calling sequence - JMS MOOUT

MDOUT:

Calls MNOUT to output the contents of the AC in decimal.
Calling sequence - JMS MDOUT

TYPOUT:

This subroutine is used to type a list of parameters, separated by commas, to either the monitor or teletype depending on the mode. The routine is entered with an instruction in the AC to "massage" each number of the parameter list. A LAC "parameter" instruction should follow the call to TYPOUT; one LAC instruction for each parameter to be typed out. The list should be terminated with an exit instruction.
Calling sequence - LAC (instruction)

```
JMS TYPOUT
LAC "parameter"
.
.
.
Exit instruction
```

KYBLIS:

This routine is normally inserted within a program loop as it listens for a teletype interrupt. If there is no interrupt, processing continues, otherwise if the following characters are typed, the corresponding action takes place - any other character causes processing to continue with the character code in the AC and the link set. The routine also checks for clock overflow and if it has occurred, will increment location six.

| CODE | ACTION |
|---|---|
| Control D(204-octal) | Exit to Debug |
| Control I(211-octal) | Exit to MONITOR |
| Control A(201-octal) | Exit to MONITOR next frame advance |

Calling sequence - JMS KYBLIS

ADVSYS:

Advances the camera the number of increments contained in the AC. If the number of increments is larger than the number of increments allowable for one camera advance for a specified camera (PULMAX), the camera is advanced PULMAX number of increments per advance until the desired number of increments has been reached. The last advance may or may not be PULMAX increments.
Calling sequence: LACC n
ADVSYS

ADVANN:

Advances the camera the number of increments contained in the AC by calling ADVSYS. Prior to advancing, the routine verifies that more than ten feet of film is in the supply magazine; if not, the routine executes a JMP to MONITOR to wait for further operator instructions.
Calling sequence: LACC n
ADVANN

ADVANF:

Keeps track of the number of frames recorded; also increments cell NUMFRM (negative number of frames to be

done) and if zero, outputs the message *FRAMES DONE and executes a JMP to MONITOR. Further, if a <CNTRL>A(2011 octal) had been recognized, a JMP to MONITOR is executed.
Calling sequence: ADVANF

ADVAN:

Calls ADVANN using the value contained in cell PULLNO; cell PICNUM is incremented if MANYUP is not defined and ADVANF is called.
Calling sequence: ADVAN

CLEAR:

Calls ADVSYS to advance all exposed film into the take-up magazine. The number of increments to be advanced is stored in cell MCMCLV.
Calling sequence: CLEAR

MVDATA:

Performs basic mag tape operations. The cell DATCOM contains the complete mag tape controller command word with the exception of the operation field. The mag tape operation must be in the AC when MVDATA is called. The data channel word count (32) and current address (33) cells must be set before calling MVDATA. Symbols for the mag tape operations are:

```
     REWCOM - rewind
     RDCOM  - read
     WRCOM  - write
     EOFCOM - write EOF
     SKPCOM - skip forward
     BCKCOM - skip backward
Calling sequence:  LAC (operation)
                   JMS MVDATA
```

ASSEMBLY PARAMETERS

BIGBUF:

This parameter must always be defined. A value of zero causes the MONITOR to be assembled with a maximum of features, none of which are expendable. A value of one causes fewer features to be assembled and allows for a program organization with expendable code.

DSKMON:

The definition of this parameter causes the assembly of the "LOAD" command and subroutines for form loading from disk.

TAPMON:

The definition of this parameter causes the assembly of code for a mag tape MONITOR system.

NODISP:

The definition of this parameter eliminates the display of operator commands. A BIGBUG=1 system automatically defines NODISP.

NOFOCS:

The definition of this parameter eliminates the code for generating the focus pattern.

NOHITS:

The definition of this parameter eliminates the code for specifying multiple hits.

# VECTOR ROUTINES (III162)

August 23,1973

Published by

Information International
12435 West Olympic Boulevard
Los Angeles, California 90064

# VECTOR ROUTINES (III162)

## ABSTRACT

III162 provides subroutines for drawing solid vectors, dashed vectors, and dotted vectors.

## REQUIREMENTS

The insert file III162 MACRO contains macro definitions and must be inserted before the SETHD and SETTL macros are used.

## USAGE

The starting point of a vector is referred to as the "head"; the end point is called the "tail". The macro SETHD is used to set the coordinates of the head; the macro SETTL sets the coordinates of the tail. Specification of a null (zero-length) vector results in the intensification of a single point.

For a vector from (XHD, YHD) to (XTL, YTL) the macro coding should be:

```
SETHD XHD, YHD
SETTL XTL, YTL
```

The following subroutines assume that the head and tail coordinates have been set with the SETHD, SETTL macros:

### SOLID VECTORS

Subroutine DRWVEC uses the hardware vector generator to produce solid vectors. Location VECHIT contains the 1's complement of the number of hits for each vector.
Calling sequence: JMS DRWVEC

### DASHED VECTORS

Subroutine DRWVDS uses the hardware vector generator to produce dashed vectors. Dashes are made by blanking and unblanking the PLS while the vector generator is drawing. The dashed line format is specified by a call to COMDSH before calling DRWVDS.
Calling sequence: STL or CLL

```
JMS COMDSH (compile dashed code)
LAC "SMALLER LENGTH" (arg. 1)
LAC "LARGER LENGTH" (arg. 2)
```

If the Link is zero, then argument 1 refers to the length of the dash and argument 2 refers to the length of the space.

If the Link is one, then argument 1 refers to the length of the space and argument 2 refers to the length of the dash.

It is necessary to call COMDSH only once for a particular dashed line format.

Any change in format requires another call to COMDSH.

A call to DRWVDS utilizes the output of COMDSH and draws the dashed vector. Location VECHIT contains the 1's complement of the number of hits for each vector.
Calling sequence: JMS DRWVDS

DOTTED VECTORS

Subroutine DRWDOT produces dotted vectors without the use of the hardware vector generator. The assembly parameter DSH determines the spacing between dots. The number of scope points between dots, as measured on the axis of the larger component, is 2**DSH. Location DOTHIT contains the 1's complement of the number of hits for each vector.
CALLING sequence: JMS DRWDOT

ASSEMBLY PARAMETERS

DASHED - When DASHED is defined, routines will be assembled to produce dashed vectors.

DOTVEC- When DOTVEC is defined, routines will be assembled to produced dotted vectors.

DSH - When DOTVEC is defined, DSH determines the number of scope points, X, between dots, as measured on the axis of the larger component.
X = 2**DSH

CHARACTER ROUTINES (III147)
March 27, 1974

Published by
Information International
12435 West Olympic Boulevard
Los Angeles, California 90064

# CHARACTER ROUTINES (III147)

## ABSTRACT

III147 provides subroutines for drawing characters, with mono or proportional spacing, adding extra character/word space for justification, measuring the width of characters, and computing character size from scope points of height.

## REQUIREMENTS

The insert file III147 SPEC contains the subroutines for proportional spacing, justification, and character width measuring.

## USAGE

The character plotting subroutine VCHAR draws a single character whose character code is in the accumulator. The character will be plotted with its lower left-hand corner at the current beam position.
Calling sequence:

```
LAC "Character Code"
JMS VCHAR
```

Subroutine VCHAR uses a pointer table at VCHTAB to find the stroking information for the requested character code. The word at VCHTAB + Character Code is used as the address of the first word of stroking data.

## CHARACTER SPACING

After drawing the character, VCHAR will move the beam one character space. The direction of movement is dependent upon the current character rotation; upright characters are spaced from left to right.

### Monospaced Characters

Monospaced characters are drawn as is, without any initial beam movement. The beam position returns to the origin after completing the character, whereupon the beam is moved one character space. This distance is determined by the last value loaded with LCDX or LCDY instruction, depending upon rotation.

### Proportional Spaced Characters

Before beginning a proportional spaced character, the beam position is moved forward or back to ensure that the leftmost light of the character will begin at the origin. After the character is completed, beam position is moved to the rightmost light of the character, then further by one character space. This

distance is specified in scope points by the cell VCHSPC.

If proportional spacing is required of VCHAR, the cell VCPROP must contain an OPR instruction. For monospacing, VCPROP must contain a SKP (default).

JUSTIFICATION

Justification allows the addition of extra (positive or negative) character spacing, treating a space character (40 octal) differently from other characters.

The subroutine JUSTIN sets up VCHAR to add spacing for subsequent character plotting as follows:

JUSTC1 - Additional spacing (positive or negative) for non-space characters.

JUSTW1 - Number of non-spaced characters to receive one additional scope point of character space after JUSTC1.

JUSTC2 - Additional spacing (positive or negative) for space characters (40 octal).

JUSTW2 - Number of space characters to receive one additional scope point of space after JUSTC2.

Once these values have been initialized with JUSTIN, placing a SKP instruction at JUSTSW will cause spacing to be added for subsequent calls to VCHAR. Placing an OPR instruction at JUSTSW will plot characters normally.

MEASURING

VCHAR may also be used to measure a character without plotting. To accomplish this, place an OPR instruction at VCMEAS before calling VCHAR. The width, including character spacing, will be added to the value of CHWID, and the result placed in CHWID. Repeated calls without clearing CHWID will thus sum up the width of a character string.

If justification is being used and JUSTSW contains a SKP instruction, justify spacing will be added into the width calculation. In this case, JUSTIN should be used to initialize the justification routines before beginning the measure.

For character plotting, VCMEAS should contain a SKP instruction.

## CHARACTER SIZE

The subroutine VCHSIZ will convert a scope point character height from the character baseline, to a character size number between Ø and 63. This computation assumes the characters are designed with a cap height equal to box height, and chooses the next highest size available. The box height in strokes should be stored in CHBUXH.
Calling Sequence:

```
LAC "Character Height"
JMS VCHSIZ      /size number returned in AC
```

The subroutine VCINIT and VCIN scale the character width and offset values, used by VCHAR, for a given character height.
Calling sequence:

```
LAC "Character Height"
JMS VCINIT
JMS VCIN
```

The subroutine VCINIT stores the character height and initializes VCIN to scale when called. VCIN does the measuring and sets itself to ignore further calls until VCINIT is called again. Thus, if VCINIT is called when character size is changed, and VCIN is called before plotting characters, scaling will occur only when necessary.

## ASSEMBLY PARAMETERS

COMPOZ - When defined, width measuring routines will be assembled.

COMPRP - When defined, proportional spacing routines will be assembled.

COMJUS - When defined, justification routines will be assembled.

CHARACTER DISPATCH TABLES (III164)
March 27, 1974

Published by
Information International
12435 West Olympic Boulevard
Los Angeles, California 90064

# CHARACTER DISPATCH TABLES (III164)

## ABSTRACT

III164 provides standard character dispatch tables to relate character codes to character descriptors. Dispatch tables are supplied for Information International standard (IIISET), BCD, and EBCDIC, and others tailored to specific applications. III164 defines the ZZ macro used in optionally reserving extra words between character descriptors in a font.

## REQUIREMENTS

The insert file III164 SPEC contains dispatch tables tailored for specific applications. The insert file III164 MACRO defines the ZZ macro.

## USAGE

A character dispatch table is a series of 18-bit word addresses pointing to the character descriptors to be used for plotting characters. These addresses are in order of the character code; that is, the first address points to the desired descriptor for character code zero, the second address for code one, and so forth. This table can be used by the hardware to plot characters from character codes.

The symbolic font format for the FR 80 and CUMp 80 includes a unique symbol assigned to the starting address of each character descriptor. This symbol is "CCnnnn", where "nnnn" is a four-digit octal character identification code. All characters are assigned a unique character ID code; those characters having an Information International standard character code are assigned that character code as their character ID number; for example, the Information International character code for an upper case "A" is 101 (octal), so the symbol defined as the starting address for the "A" in a font is CC0101. The symbol "CC7777" is reserved for a "null" or non-printing character.

The file III164, then, defines a series of these symbolic addresses as a table beginning at location VCHTAB. For IIISET, this table is simply in order of character ID code:

```
VCHTAB, CC7777
        CC0001
        CC0002
        .
        .
        .
```

All dispatch table positions for unused character codes are filled with the dispatch for the null character, "CC7777."

## CHARACTER DISPATCH TABLES (III164)

To provide a dispatch table for other character sets, III164 defines the symbolic addresses in a different order. For example, the EBCDIC code for the letter "A" is a 301 octal; for an EBCDIC dispatch table, then, III164 would define a CC0101 at VCHTAB+301 octal, rather than VCHTAB+101 octal for a IIISET dispatch table.

The file III164 may thus be assembled with a font in a program, and it will provide a dispatch table, according to assembly parameters, that references the absolute addresses of character descriptions in the fonts. The symbolic font files are set up with conditional assembly statements around each character description directing it to be assembled if the character ID symbol for it has been referenced. Thus, III164 must be inserted in the user program prior to the font; the assembly parameter FONT is provided for this purpose, and may direct automatic insertion of the required font at the end of III164.

The length of the dispatch table depends upon the number of bits to be used for a character code. BCD is a 6-bit code, therefore the BCD dispatch table is 64 words long; the dispatch table for 8-bit EBCDIC is 256 words long, etc.

SPCTAB

A table is included in III164 at location SPCTAB for conversion of character sizes between scope points of height and character size number (0-63). The table consists of 64 words representing the scope points of height for a 14-stroke-high character, for character sizes 0-63.

ASSEMBLY PARAMETERS

IIISET - If defined, assemble a dispatch table for Information International standard character codes.

EBCDIC - If defined, assemble an EBCDIC dispatch table.

BCD - If defined, assemble a BCD dispatch table.

COMPRP - If defined, causes ZZ macro to reserve two words between character descriptors for character width and offset. Necessary for plotting characters in proportional-spaced mode.

LOCASE - If defined, include lower case characters in the character set; otherwise upper case only.

FONT - Must be defined to direct the inclusion of a font at the end of III164.

```
FONT =  0   FILM
     =  1   MICRO
     =  2   OCRB
     =  3   BELL
     =  4   BELL2
     =  5   MICRO2
     =  6   MICRO1
     =  7   FILM1
     =  8   COUR
     =  9   OCRBZ
     = 10   BELLX
     = 11   BELL2S
     = 12   IBMALD
     = 13   NDFILM
     = 14   OCRB3
     = 15   NEW
```

# MAGNETIC TAPE BUFFERING ROUTINES (III163)

March 26, 1974

Published by
Information International
12435 West Olympic Boulevard
Los Angeles, California 90064

# MAGNETIC TAPE BUFFERING ROUTINES (III163)

## ABSTRACT

III163 provides subroutines for initialization, reading, and repositioning of single or multiple tape units.

## REQUIREMENTS

The Information International MONITOR system is required.

## USAGE

### READ ROUTINES

Subroutine MTRINI performs buffer initialization. The two's complement buffer size in FR 80 words must be in the MONITOR cell PBUFSZ. For single-buffered input, the buffer address minus one must be in the MONITOR cell MTAREA; for double-buffered input, CURBUF and NEXBUF must contain buffer pointers. Each pointer is a buffer address minus one.
Calling sequence for single buffered:

```
     LAC (BUFBEG-1     /buffer address -1
     DAC MTAREA
     LAC (1-BUFSIZ     /two's complement buffer size
     DAC PBUFSZ
     JMS MTRINI
```

Calling sequence for double buffered:

```
     LAC (BUFBEG-1
     DAC CURBUF#
     LAC (1-BUFSIZ
     DAC PBUFSZ
     TCA
     ADD CURBUF
     DAC NEXBUF#
     JMS MTRINI
```

If multiple tape units are allowed (MTMANY>0), calling sequence is as follows:

```
     LAC UNIT
     JMS MTRINI        /unit number in AC
     LAC (1-BUFSIZ     /two's complement buffer size
     LAC (BUFONE-1     /buffer address -1
     LAC (BUFTWO-1     /2nd buffer (only necessary
                       /if double buffered)
```

Subroutine MTLAC will return (in the AC) 18-bit words sequentially from the buffer. Each buffer is filled automatically as it becomes empty.
Calling sequence: JMS MTLAC

Subroutine MTBYTE will return (in the AC) a requested number of data bits. The number of bits requested is in the AC when MTBYTE is called. If data is known to cross record boundaries, MTBYSW should contain a SKP. If not, MTBYSW should contain a NOP and the partial word at the end of the record will be ignored. NOTE: calls to the MTBYTE and MTLAC routines cannot be intermixed.

```
Calling sequence:  LAW n
                   JMS MTBYTE
                   AND (mask appropriate to n bits)
Alternate calling sequence:  GETT n
```

WRITE ROUTINES

Subroutine MTWINI performs buffer initialization, record size is specified by the buffer size.
Calling sequence: Same as MTRINI

Subroutine MTDAC will put the 18 bit AC into the next position in the output buffer.
Calling sequence: JMS MTDAC

Subroutine MTPUT will write a number of bits (the number specified by the cell BYTE) from the AC. The number specified refers to the low order AC bits.

MULTIPLE TAPE UNITS (MTMANY>0)

Subroutine MTSEL will select a tape unit for reading or writing. The AC must contain a unit number previously used with a MTRINI or MTWINI.

```
Calling sequence:  LAC UNIT /tape unit #
                   JMS MTSEL
```

REPOSITIONING ROUTINES

These routines consist of a pushdown list and backup routines to process nested repeats of data from mag tape.

1. Save current mag tape pointers.
   Calling sequence: JMS MTPUSH

2. Reposition the mag tape to the previous "push" position.
   Calling sequence: JMS MTREPO

3. Remove the top mag tape pointers from the pushdown list.
   Calling sequence: JMS MTPOP

### END OF FILE PROCESSING

A jump to a user-supplied EOF processor can be stored at MTEOF. Default EOF processing types *END OF FILE and returns to the MONITOR.

## ASSEMBLY PARAMETERS

TWOBUF - Definition of TWOBUF causes assembly of double-buffered read routines; if it is not defined, the read routines will be single-buffered.

MTPTR - If the parameter MTPTR is not defined, the read routines will allocate a core location for indexing through the data buffer. If the user desires a faster access time and can afford the dedicated use of an autoindex register, define MTPTR=n where n is 10 through 17.

MTWRDS - If data is to be accessed a word at a time (no calls to MTBYTE), then a core savings is realized if MTWRDS is defined.

MTRPT - If the repositioning routines are needed, define MTRPT==n where n is the maximum depth of nesting.

MTMANY - If defined and non-zero, allows handling of more than 1 tape unit; a value of 1 allows units 0 and 1, 2 allows 0, 1 and 2, etc.

MTWRIT - If defined, mag tape write routines (MTDAC, MTPUT) are included.

DISK I/O SYSTEM (III183)

March 5, 1974

Published By
Information International
12435 West Olympic Boulevard
Los Angeles, California 90064

# DISK I/O ROUTINES

## ABSTRACT

III-183 is a set of subroutines which provide capability for Disk Input/Output. A push down stack is provided for nested reads.

## USAGE

The disk organization is described completely in the "Disk Operating System" documentation. The disk has 1024 blocks of 256 18-bit words. Resident on the disk is a master directory which indexes up to 15 user directories, and a track usage table (TUT) which indicates the state (used or unused) of all blocks on the disk.

Data is read from the disk or written to the disk in blocks of 256 18 bit words; however, the disk buffering routines make the block structure invisible to the user.

A disk file is referenced by a directory name and a file name. the subroutines in III183 assume that the directory name is in location DKFILE and that the file name is in locations DKFILE+1 through DKFILE+4.

## SUBROUTINES

### DKRINI (READ INITIALIZE):

This subroutine initializes the system to read from the file specified by DKFILE through DKFILE+4

Calling sequence - DKRINI

| | |
|---|---|
| RETURN | DKRINI returns here if the file doesn't exist. |
| RETURN+1 | normal return for DKRINI |

### DKWINI (WRITE INITIALIZE):

This subroutine initializes the system to write on the disk by finding an available block, setting up the buffer, and storing the default directory and file name into locations DKFILE through DKFILE+4. Control is returned to DEBUG if the disk is full.

Calling sequence - DKWINI

DKREAD:

This subroutine returns to the next 9 bit byte in the low order AC from the file opened by DKRINI.

```
Calling Sequence:   DKREAD
                    SAD (EOFCHR /check for EOF char.
```

DKRDWD:

This subrotine returns the next 18 bit word in the AC from the file opened by DKRINI.

```
Calling Sequence -  DKRDWD
```

DKWRIT:

This subroutine writes the 9 low order bits of the AC to the disk.

```
Calling Sequence -  DKWRIT
```

DKWRWD:

This subroutine writes the contents of the AC to the disk.

```
Calling Sequence -  DKWRWD
```

DKNAME:

This subroutine names all disk output since the last DKWINI. Locations DKFILE through DKFILE+4 specify the directory name and file name.

```
Calling Sequence -  DKNAME
                    RETURN      /DKNAME returns here if
                                /there is no such
                                /directory defined,
                                /or if the master
                                /directory is full.
                    RETURN+1    /DKNAME returns here if
                                /the user directory is
                                /full.
                    RETURN+2    /normal return for
                                /DKNAME
```

DKPUSH:

This subroutine saves all relevant information about the last file opened by DKRINI. This allows reading from a new file and later reopening th old file by doing a DKPOP. Extra core locations of intrest to the

user program may be pushed onto the stack by defining DKPNUM to be the number of extra words desired. THe extra core locations are DkPBLK to DKPBLK+DKPNUM, for non-zero DKPNUM.

```
Calling Sequence -  DKPUSH
                    RETURN      /push down stack is
                                /full.
                    RETURN+1    /normal return for
                                /DKPUSH
                       .
                       .
                       .
                    DKRINI      /open a new file.
```

DKPOP:

This subroutine reopens the last file processed by DKPUSH. The read routines will continue where they left off in the file.

```
Calling Sequence -  DKPOP
                    RETURN      /push down stack is
                                /empty.
                    RETURN +1   /normal return for DKPOP
```

The following subroutines are called by those listed above:

DKRTUT - This subroutine reads the track usage table into core and stores the default directory name and file name in DKFILE through DKFILE+4.

DKCRUS - This subroutine creates a new drectory with the name in DKFILE. A new master directoy is written on the disk. The routine skips on a normal return.

DKRDMD - This subroutine reads the master directory into core.

DKDRFN - This subroutine searches the master directory for the name in DKFILE. The routine skips on a normal return.

DKFFIL - This subroutine searches the user directory for the name in DKFILE+1 through DKFILE+4. THe routine skips on a normal return with a pointer in the AC to the first block of the file.

DKDLET - This subroutine deletes the file named in DKFILE through DKFILE+4 from the disk.

DISK I/O ROUTINES
USAGE

DKGET - This subroutine searches the track usage table for an available block. Control is returned to DEBUG if the disk is full.

DKINIT - This subroutine initializes to read into core the block whose number is in the AC.

DKRDSK - This subroutine handles the actual data transfer from disk to core.

DKWRBK - This subroutine handles the actual data transfer from core to disk.

ASSEMBLY PARAMETERS

NODKWT - When NODKWT is defined, it is not possible to write on the disk.

DKNWSR - When DKNWSR is defined, it allows new directories to be created.

NODKRD - When NODKRK is defined, it is not possible to read files from the disk.

DKREPL - When DKREPL is defined, it allows file replacement.

DKPNUM - When DKPNUM is defined, it allows "push-reads." Define it as the number of extra words (usually zero) to be pushed.

DKCHAN - When DKCHAN is defined, it allows a new TUT to be built when using subroutine DKNAME. Store a LAM (-0) in location DKCHAN when calling DKNAME to cause a new TUT to be built.

ADDITIONAL NOTES: FILE NAMES LOWER CASE 6 bit ASCII !

III182 reads a file name and formats it correctly for use with III183. A useful macro for producing file names in the proper format directly is:

N X, Y, Z - macro to pack characters X, Y, Z into name format.

```
.DEF N X, Y, Z
"X+40&77+6!<"Y+40&77>+6!<"Z+40&77>
.TERM
```

MASS STORAGE I/O (III188)

Published by
Information International
12435 West Olympic Boulevard
Los Angeles, California 90064

MASS STORAGE I/O

ABSTRACT

This program contains subroutines to read or write disk blocks.

USAGE

The following routines will operate either with interrupts enabled or not.

WDSK:

This subroutine is used to write a block of data.
Calling sequence - LAC (block #)
WDSK
LAC (starting core address)
NORMAL EXIT
BLKSIZ (400octal) words will be copied from the specified core address to the specified data block on the disk. Any error detected will cause retry until no error is detected.

RDSK:

This subroutine reads a block of data.
Calling sequence - LAC (block #)
RDSK
LAC (starting core address)
ERROR EXIT
NORMAL EXIT
BLKSIZ (400octal) words will be copied from the specified data block into specified consecutive core locations. The error exit will be taken if an error is detected trying to read the block more than a specified number of times. The number of read attempts to be tried before exiting is stored as a two's complement number in RDSKMX.

SYSTEM TAPE DEBUG (III126 P)

March 13, 1974

Published By
Information International
12435 West Olympic Boulevard
Los Angeles California 90064

# SYSTEM TAPE DEBUG (III126 P)

## ABSTRACT

DEBUG is a general purpose symbolic debugging program. Controls are the non-alphanumeric characters and letters when preceded by $ or <ALT MODE> (The <ALT MODE> key prints as $.) This version of DEBUG is meant for use on an 8K machine with no mass storage. It is coded very efficiently in its own interpretive language which packs two commands to a word, and hence, saves space at the expense of time. Since the program is usually waiting for a teletype response, there is no noticeable loss of speed in operation.

## REQUIREMENTS

Storage = approximately 2000(octal) locations, plus space for program's symbol table. (When large programs are read in, symbols are deleted.)

## USAGE

This program may be assembled to reside anywhere in memory. Normally it resides at the end of memory and its starting address (DEBUG) is 17677. Load DEBUG from the SYSTEM tape and type GO/<CR>. To restart, START at -100. (All errors are indicated by the printing of a ? and ringing of the bell.)

## AVAILABLE COMMANDS

(In the following, a and b stand for any legal expression, and s stands for any legal symbol.)

### Initializing Commands

$K - Kill all but permanent symbols
a<$Z - Zero memory from a up to symbol table
a<b$Z - Zero memory from a to b-1 inclusive

### Program Loading

$D - Load program and symbols from paper tape

SYSTEM TAPE DEBUG (III126 P)
Available Commands

Mode Control

```
$S          - Print words as symbolic commands
$C          - Print words as constants
$R          - Print addresses in relative symbolic
$O          - Print addresses as constants
n$R         - Change output and input number base to n (n>1)
=           - Print last quantity in current mode
:           - Print last quantity in opposite mode
```

The initial modes are S and R, and the initial output base is 8.

Arithmetic

```
+ or space  - Plus
-           - Minus
*           - Times
!           - Inclusive OR
```

Examination

```
a/          - Open a in current mode
a\          - Open a in opposite mode
/           - Open addressed register in current mode
\           - Open addressed register in opposite mode
<CR>        - Modify and close
<LF>        - Modify and open next
↑           - Modify and open previous
;           - Modify and open addressed
TAB         - Modify, open addressed, and change sequence
```

Registers

```
$A          - Accumulator
$L          - Link
$Q          - MQ
$F          - Bottom of symbol table
$J          - Location at which program starts on '
$M          - Mask for searches
$M+1        - Upper limit
.           - Current location
#           - Last quantity typed (by either user or DEBUG)
```

Running

```
'           - Go to location contained in $J
a'          - Go to a
"           - Go to location contained in DEBUG-1
a"          - Go to a and put a in DEBUG-1
```

$J and DEBUG-1 initially contain halts. If ' or " are used without arguments before changing $J or DEBUG-1, an error is indicated.

When a paper tape is loaded, its start or pause address is put in $J.

A running program may JMS to DEBUG-1 and a " will return control to the program with the status restored.

NOTE: If it is desired that DEBUG type the return address for such an entry, the JMS DEBUG-1 should be preceded by a DZM DEBUG-2.

```
$B          - Remove breakpoint
a$B         - Put breakpoint at a
$P          - Proceed after breakpoint stop
n$P         - Proceed and do not break next n times
a$X         - Execute the command a
```

If the execution of the command results in a skip, the bell will ring.

Symbol Definition

```
a<s>        - Define the symbol s as the quantity a
s,          - Define the current location as s
```

Searching

```
a$W         - Search for and print all locations equal to a
a$N         - Search for locations not equal to a
```

SYSTEM TAPE DEBUG (III126 P)
Available Commands

## COMMAND SUMMARY - DEBUG

| | |
|---|---|
| $K | kill symbols |
| $Z | zero memory |
| a<$Z | zero from a on |
| a<b$Z | zero from a to b-1 |
| $D | load program and symbols |
| $S | symbolic |
| $C | constant |
| $R | relative |
| $O | octal |
| n$R | set base n |
| = | equals, current mode |
| : | equals, opposite mode |
| + | plus |
| - | minus |
| * | times |
| ! | or |
| / | open, current mode |
| \ | open, opposite mode |
| <CR> | close |
| <LF> | open next |
| ↑ | open previous |
| ; | open addressed |
| TAB | new sequence,addressed |

| | |
|---|---|
| $A | accumulator |
| $L | link |
| $Q | MQ |
| $F | symbol table |
| $J | start address |
| $M | search mask |
| $M+1 | upper limit |
| . | current address |
| # | last quantity typed |
| ' | start |
| " | return |
| $B | breakpoint |
| $P | proceed |
| $X | execute |
| a<s> | define a as s |
| s, | define current as s |
| $W | word search |
| $N | not-word search |