DSPs and the DSP Interface

1. DSPS AND THEIR ENVIRONMENT

Device Support Programs (DSPs) are a collections of routines with a standard interface designed to make access to a variety of devices as uniform as possible.

A program called the DSP Interface is provided to do parameter and operation sequence checking for the DSPs. The DSP Interface was originally designed by John Hogg, and Alan Ballard. The calling sequence for some of the entry points was later revised to make it easier to to fit the DSP Interface in with the requirements of the MTS job program.

a. Environment

The DSP environment is built on the environment of the subtasking monitor. DSPs should request services via the routines provided by the subtasking monitor, or routines described in this document. MTS system subroutines should be used with caution, since DSPs in general should be able to run in non-MTS tasks. Some MTS system services do task level synchronous waits, which might conflict with a subtasking environment.

All DSP Interface procedures, and DSP entry points, use an S-type calling sequence. All DSP Interface procedures are called according to the RM conventions, and require R11 to contain the address of a pseudo-register vector (as initialized by DSPPRVI) which contains the addresses of various service routines, and tables. The DSPs are called with the same pseudo-register vector address in R11.

b. Entry Points

Each DSP begins with a transfer vector defining the entry points. This is defined by the macro DSPTV in RMGR:RMGR*M. The first 3 characters from the label on the DSPTV macro replace the "DSP" in the routine names described below. The routine names and order defined by this are:

DSPINIT - responsible for acquiring the device.
DSPOPEN - prepares to do I/O in the manner requested.
DSPREAD - reads a record from the device.
DSPWRIT - writes a record to the device.
DSPCONT - controls the device, or action of the DSP.
DSPSENS - returns state of device, and/or DSP.
DSPCLOS - returns to state ready for new OPEN.
DSPRELE - releases the device and cleans up.

The corresponding RSECTs defined by the macro are INIRCT, OPERCT, REARCT, etc.

2. THE DSP INTERFACE

Besides parameter and sequence checking, the DSP Interface is responsible for selecting which DSP to use (based on device type). The DSP interface creates a control block called a File, Device or Intertask Block (FDIB) to keep track of individual invocations of the DSP Interface.

a. Entry Points

The entry points in the DSP Interface are:

RMGFDIB - checks parameters, creates FDIB, calls DSPINIT.
RMOPEN - checks parameters, calls DSPOPEN.
RMREAD - checks parameters, calls DSPREAD.
RMWRITE - checks parameters, calls DSPREAD.
RMCNTRL - checks parameters, calls DSPREAD.
RMSENSE - checks parameters, calls DSPREAD.
RMCLOSE - checks parameter, calls DSPREAD.
RMFFDIB - checks parameter, calls DSPRELE, frees FDIB.

Each of these DSP Interface procedure calls its corresponding DSP entry with the original parameter list, except for RMGFDIB, which constructs an FDIB and passes it to the DSPINIT entry point.

Additionally the following are provided.

FSTREAD - calls DSPREAD with no checking.
FSTWRITE - calls DSPWRIT with no checking.
RMCKFDIB - verifies that an FDIB is valid.

b. DSP Selection

The DSP to use for a particular invocation of the DSP Interface is selected by RMGFDIB based on the device type given as a parameter. The type is matched to a particular DSP through the use of DSP type tables. RMGFDIB scans two tables to try and match the given type to a DSP. The address of the first table scanned is found in the pseudo-register UDSPTAB. The address of this table is provided by applications where applicable to define special or alternate DSPs. The address of the second table is found in the pseudo-register DSPTAB. This table contains all the default DSPs.

The format of the table is a fullword containing the number of table entries, followed by the entries. Each entry is sixteen bytes long, and consists of an eight byte type, followed by a halfword maximum I/O length, followed by a halfword maximum name length, followed by the address of the DSP transfer vector.

Table entries may be generated by the DSPTAB macro in

RMGR:RMGR*M.

The first DSP type matching the requested type is selected.

c. Parameter Checking

A DSP may assume that all parameter addresses passed to it are valid except that for DSPREAD, only the first byte of the buffer has been checked as addressable, and for DSPWRIT, only the first and last bytes have been checked. It may also assume that the FDIB pointer is valid, and points to a valid FDIB.

d. Sequence Checking

The DSP Interface will check for correct operation sequence; e.g., calls to DSPREAD will not be permitted unless the FDIB is open, calls to DSPOPEN will not be permitted unless it is closed. The DSP will not be called if a sequence error is detected.

Fields are set in the FDIB on each call to the DSP interface indicating the last operation performed, the return address of the last caller (these will be set before calling the DSP), and the last return code.

e. Error Handling

A message indicating clearly the flavour of the error, the entry point called, and the callers return address is issued for every error detected by the DSP Interface. The messages conform to the Resource Manager message scheme, and are emitted to the on-unit of the subtask making the erroneous call. If the DSP Interface detects any errors it will not call the DSP routine.

For errors detected by the DSP, a message should be issued, by the DSP, and return made with the appropriate return code. The return code received from a DSP will be passed on the the caller of the DSP Interface routine unchanged.

f. Return Codes

The following return codes are used for all DSP Interface procedures and DSPs:

0 O.K.

4 Exception (full on write, eof on read etc.)

8 Parameter or sequence error. This will normally be reported by the DSP Interface routine.

12 External error (hardware, external software, bad data format on OMR card, etc.)

16 Internal Error. (Software error in DSP.)

20 Wait Interrupted.

3. DSP INTERFACE EXTERNAL PROCEDURES

Routine: RMGFDIB

Location: DSPLCS

Call Type: S(4)

Purpose: To create an invocation the DSP Interface, and initialize the DSP.

Input parameters:

1	FDINAME, (File, Device, or Intertask name).
2	halfword length of FDINAME.
3	eight character type.
4	location in which to return FDIB pointer.

Return values: none

Return codes:

0-20	standard meanings
24	file, device etc. is busy.
28	file, device etc. is not operational.
32	an FDIB already exits for this FDINAME.

Description:

RMGFDIB acquires and initializes a FDIB, looks for the DSP in the type tables and calls the DSPINIT entry of the DSP matched. The DSPINIT routine should acquire the device and do what is necessary for its initialization.

The FDIB contains a pointer to the FDINAME and length. The pointer points to a halfword length followed by the FDINAME. The name will always be followed by a terminating blank (which may be useful in messages etc.) The blank is not included in the length.

A field is provided in the FDIB for the DSP to chain any space it needs.

If the return code from RMGFDIB is > 0, the FDIB will not have been created. If the DSPINIT procedure returns a non-zero return code, the FDIB created by RMGFDIB, and the space for the FDINAME will be released.

Routine: RMFFDIB

Location: DSPLCS

Call Type: S(1)

Purpose: To destroy an invocation the DSP Interface.

Input parameters: 1 FDIB pointer.

Return values: none

Return codes: 0-20 standard meanings

Description:

RMFFDIB checks that the FDIB pointer is valid, and calls the DSPRELE entry point in the DSP. When DSPRELE returns, the FDIB is released.

The DSP should clean up, and release the device.

Routine: RMOPEN

Location: DSPLCS

Call Type: S(2)

Purpose: To prepare the DSP for I/O.

Input parameters:	1	FDIB pointer.
	2	access code.
Return values:	none	
Return codes:	0-20	standard meanings

Description:

RMOPEN checks that the FDIB pointer is valid, that the access code is addressable, and calls the DSPOPEN entry point in the DSP.

The DSP should check the validity of the access code, and prepare for subsequent read, write, control and sense calls. The meaning of the access code is DSP dependent.

Routine: RMCLOSE

Location: DSPLCS

Call Type: S(1)

Purpose: To cause the DSP complete all pending operations.

Input parameters: 1 FDIB pointer.

Return values: none

Return codes: 0-20 standard meanings

Description:

RMCLOSE checks that the FDIB pointer is valid, and calls the DSPCLOS entry point in the DSP.

The DSP should flush all buffers, terminate all I/O operations etc.

Routine: RMREAD/FSTREAD

Location: DSPLCS

Call Type: S(8)

Purpose: To read a record from the file or device.

Input parameters:

1	FDIB pointer.
2	buffer to read into.
3	halfword length of the buffer.
4	8 bytes of modifiers, MTS compatible
5	halfword length of data placed in buffer by the DSP.
6	record/line number of record read.
7	notify code.
8	actual length of data record.

Return values: none

Return codes: 0-20 standard meanings

Description:

RMREAD checks that the FDIB pointer is valid, that the other parameters are addressable in the required mode, and calls the DSPREAD entry point in the DSP. Parameters 2, and 5 through 8 are modified by the DSP. The record/line number is a parameter to the DSP, if an indexed operation is indicated by the modifiers.

A notify code should always be returned by the DSP, and if the code is non-zero, and notify is specified in the modifiers, a return code of 4 should be given. Return code 4 is given by the DSP on end-of-file regardless of the setting of the notify modifier.

The standard notify codes for RMREAD are:

4 - end-of-file
8 - record truncated.

The routine FSTREAD does no parameter checking, but just calls the DSPREAD entry point. It is useful in applications where one is sure of the parameters, and/or the cost of the parameter checking is too high a price to pay.

Routine: RMWRITE/FSTWRITE

Location: DSPLCS

Call Type: S(7)

Purpose: To write a record to the file or device.

Input parameters:

1 FDIB pointer.
2 buffer containing data to write.
3 halfword length of the data in the buffer.
4 8 bytes of modifiers, MTS compatible
5 halfword length of data written by the DSP.
6 record/line number of record written.
7 notify code.

Return values: none

Return codes: 0-20 standard meanings

Description:

RMWRITE checks that the FDIB pointer is valid, that the other parameters are addressable in the required mode, and calls the DSPWRIT entry point in the DSP. Parameters 5 through 7 are modified by the DSP. The record/line number is a parameter to the DSP, if an indexed operation is indicated by the modifiers.

A notify code should always be returned by the DSP, and if the code is non-zero, and notify is specified in the modifiers, a return code of 4 should be given. Return code 4 is given by the DSP for a device full condition regardless of the setting of the notify modifier.

The standard notify codes for RMWRITE are:

4 - device full.
8 - record truncated.

The routine FSTWRITE does no parameter checking, but just calls the DSPWRIT entry point. It is useful in applications where one is sure of the parameters, and/or the cost of the parameter checking is too high a price to pay.

Routine: RMCNTRL

Location: DSPLCS

Call Type: S(3)

Purpose: To control the operation of the file, device or DSP.

Input parameters:	1	FDIB pointer.
	2	control command.
	3	halfword length of control command.

Return values: none

Return codes: 0-20 standard meanings

Description:

RMCNTRL checks that the FDIB pointer is valid, that the other parameters are addressable, that the length of the control command is reasonable (currently <= 512 bytes) and calls the DSPCONT entry point in the DSP.

Control commands should be passed one at a time to RMCNTRL.

Routine: RMSENSE

Location: DSPLCS

Call Type: S(6,1)

Purpose: To sense the state of the file, device, DSP etc.

Input parameters:	1	FDIB pointer.
	2	sense command.
	3	halfword length of sense command.
	4	sense information returned by DSP.
	5	halfword length of information returned.
	6	halfword maximum length of return area.
Return values:	1	- code for type of information returned.
Return codes:	0-20	standard meanings

Description:

RMSENSE checks that the FDIB pointer is valid, that the other parameters are addressable, and calls DSPSENS entry point in the DSP.

The DSP should put the information in the buffer if it fits, update the length and return with a code in R0 indicating what kind of information is present. The currently defined codes are:

1 - information is a binary number.
2 - information is a character string.

Sense commands should be passed one at a time to RMSENSE.

Routine: RMCKFDIB

Location: DSPLCS

Call Type: S(1)

Purpose: To verify the validity of an FDIB pointer.

Input parameters:	1	FDIB pointer.
Return values:	none	
Return codes:	0	O.K.
	4	pointer does not point to an FDIB.

Description:

RMCKFDIB scans the FDIB chain to verify that the pointer points to one of the entries.

4. DSP INTERFACE INTERNAL PROCEEDURES

These are described here for completeness.

Routine: CPLIST

Location: internal

Call Type: R(4,1)

Purpose: To verify the addressabilty of the parameter list, and the validity of the FDIB pointer.

Input parameters:	1	number of parameter addresses in parameter list.
	2	parameter list pointer.
	3	DSP interface callers address (for error messages).
	4	address of the Interface routine name (for error messages).
Return values:	1	FDIB pointer.
Return codes:	0	O.K.
	4	FDIB pointer is invalid (no message issued in this case).
	8	parameter list is not addressable in whole or in part.

Description:

CPLIST just verifies that the parameter list is addressable, and that the first parameter is indeed an FDIB pointer.

Routine: CRWP

Location: internal

Call Type: R(2,0)

Purpose: To verify the addressabilty of the parameters for RMREAD or RMWRITE.

Input parameters:	1	FDIB pointer.
	2	parameter list pointer.
Return values:	none	
Return codes:	0	O.K.
	8	some parameter is not addressable, or is invalid (a message has been issued.)

Description:

CRWP makes sure that all parameters are addressable in the mode required, and also checks that the buffer length is not negative.

Routine: CCSP

Location: internal

Call Type: R(2,0)

Purpose: To verify the addressabilty of the parameters for RMCNTRL or RMSENSE.

Input parameters:	1	FDIB pointer.
	2	parameter list pointer.
Return values:	none	
Return codes:	0	O.K.
	8	some parameter is not addressable, or is invalid (a message has been issued.)

Description:

CCSP makes sure that all parameters are addressable in the mode required, and checks the validity of the command length.

5. DSP SERVICE ROUTINES

The following set of routines are provided as a set of service procedures for some of the common DSP required services not provided by the subtasking monitor.

Routine: DSPPRVI

Location: DSPLCS

Call Type: R(1,0)

Purpose: To initialize the pseudo-register vector whose address is in R11.

Input parameters:	1	total size of the pseudo-register vector.
Return values:	none	
Return codes:	0	OK

Description:

This routine assumes (as do all others) that the pseudo-register vector address is in R11. The pseudo-register vector is initialized by calling MNTPRVI, and the address of the default DSP type and DSP External form message tables are filled in.

Routine: MTSDSPRV

Location: DSPLCS

Call Type: R(0,0)

Purpose: To initialize certain pseudo-registers in the pseudo-register vector for an MTS environment.

Input parameters: none

Return values: none

Return codes: 0 OK

Description:

This routine calls MTSMNPRV to set up the subtasking monitor for for an MTS environment. The addresses of of RMTKINFO and RMTKCINF in the pseudo-register vector are replaced by MTTKINFO and MTTKCINF respectively. These use MTS's GUINFO and CUINFO routines.

Routine: GETPAR

Location: DSPLCS

Call Type: R(1,2)

Purpose: A simple token scanning routine.

Input parameters:	1	address of a two word block, containing the current length and address of the string to be scanned. These are updated by GETPAR.
Return values:	1	length of token.
	2	address of token.
Return codes:	0	O.K.
	4	end of string, no more tokens.

Description:

The address and length of the next token in the string are returned. A token is defined as a sequence of non-blank characters.

Routine: KEYWORD

Location: DSPLCS

Call Type: R(4,4)

Purpose: A simple keyword routine.

Input parameters:	1	length of string containing potential keyword.
	2	address of the string.
	3	address of keyword table.
	4	address of where to put LHS keyword, or zero.
Return values:	1	R0 value from table.
	2	R1 value from table.
	3	length of right hand side.
	4	address of right hand side.
Return codes:	0	O.K.
	4	Left hand side is not in table.

Description:

This routine only processes keyword expressions of the form LHS=RHS, or LHS. Each table entry has the following format:

AL1(L'keyword),AL1(minlen),C'keyword',AL4(R1 value),AL4(R0 value)

The table is terminated by X'FF'.

Table entries may be generated by the TBLENT macro in RMGR:RMGR*M.

Routine: CVTDB

Location: DSPLCS

Call Type: R(2,1)

Purpose: To convert an EBCDIC string representing a decimal number to binary.

Input parameters:	1	length of string.
	2	address of string.
Return values:	1	binary value.
Return codes:	0	O.K.
	4	string does not represent a valid decimal number, or is too long.

Routine: CVTHB

Location: DSPLCS

Call Type: R(2,1)

Purpose: To convert an EBCDIC string representing a hexadecimal number to binary.

Input parameters:	1	length of string.
	2	address of string.
Return values:	1	binary value.
Return codes:	0	O.K.
	4	string does not represent a valid hexadecimal number, or is too long.

Routine: CVTLNRB

Location: DSPLCS

Call Type: R(2,1)

Purpose: To convert an external form line number to internal form.

Input parameters:	1	length of string.
	2	address of string.
Return values:	1	binary value.
Return codes:	0	O.K.
	4	string does not represent a valid line number.

Routine: DSPCLEAN

Location: DSPLCS

Call Type: R(0,0)

Purpose: To clean up the DSP world in case of fatal error.

Input parameters: none

Return values: none

Return codes: 0 O.K.

Description:

This routine will try to release all devices, intertask nodes, and files that are currently active. It attempts to close open files so that data will not be lost.

Routine: RMTKINFO/MTTKINFO

Location: pseudo-register vector

Call Type: R(3,0)

Purpose: To return information about the task or environment.

Input parameters:	1	address of 8 character information item keyword.
	2	address of returns area.
	3	length of returns area.
Return values:	none	
Return codes:	0	O.K.
	4	invalid keyword.
	20	wait aborted.

Description:

This routine returns various items of information about the task and environment. It is similar to the MTS routine GUINFO. The routine MTSDSPRV replaces the address of RMTKINFO in the pseudo-register vector with MTTKINFO.

Routine: RMTKCINF/MTTKCINF

Location: pseudo-register vector

Call Type: R(3,0)

Purpose: To change information about the task or environment.

Input parameters:	1	address of 8 character information item keyword.
	2	address new information.
	3	length new information.
Return values:	none	
Return codes:	0	O.K.
	4	invalid keyword, or change not allowed.
	20	wait aborted.

Description:

This routine changes various items of information about the task and environment. It is similar to the MTS routine CUINFO. The routine MTSDSPRV replaces the address of RMTKCINF in the pseudo-register vector with MTTKCINF.

6. THE MESSAGE EXPANSION SYSTEM

This is a brief overview of the Resource Manager message system. It is included here, because the message system routines are used by the DSPs and the DSP interface, and are therefore DSP services.

The message expansion system is independent of the Resource Manager at the low-level routine interface. It is described here, however, in terms of the Resource Manager environment.

a. Concept

The Message Expansion System is a general purpose scheme that allows for a standardization of system messages. Programs will emit a Canonical Form Message (CFM) which contains values identified by keywords, and in general no message text. The CFM is in a form that is fairly easily handled by programs. Each different message is given a unique message number which is used in expanding the message data into human readable form. Each CFM is associated with one or more Expanded Form Messages (EFMs). An EFM is a message in which keywords identify the data items (values) which are to be supplied by the CFM.

b. Definition Of Terms

The internals of the Message Expansion System are relatively complicated, but the external interfaces are fairly easy to use. First, though, a few definitions.

®® MSGCC Packet

A new concept and command for MSGCC. Instead of generating an S-constant for an operand, MSGCC will produce a "packet". A packet consists of a keyword, preceded by its length and a value, preceded by its length.

An example of a packet in assembler notation:

```
AL1(03),C'CSW',AL(8),X'0160350000000100'
```

®® Canonical Form Message - CFM

Instead of emitting a text message, a program, using the message expansion system, causes a CFM to be issued. The CFM consists of a header, followed by a set (maybe empty) of packets. The header has the following format:

i - globally unique message number.

ii - A 4 character ID of the module that is sending the message.

iii - An invocation ID for the calling module.

iv - Store clock value of the time that the message was sent.

v - A message reference (sequence) number.

vi - Length of the optional packet list.

®® CFM Prototype

CFM prototypes are input to the system message routine MSGCC, and are normally produced via the CFM, or PCFM macros.

An example of a CFM prototype is:

```
CFM     #DSPACER,MODULE=FDSP,*
        FDINAME((0(R6),END=C' ',LEN=#RMFNLEN))
        CEND
```

#DSPACER is the unique number and "FDSP" is the module ID. There is one keyword value pair with keyword "FDINAME" and value pointed to by register 6, which is #RMFNLEN bytes long, or if shorter is terminated by a blank.

The macros allow continuation by specifying * as the last operand on the CFM line. The list is then terminated with a CEND statement.

®® Expanded Form Message - EFM

An EFM is a message fit for human consumption, and is obtained by the substitution of the CFM packets into an EFM prototype.

®® EFM Prototype

EFM prototypes are also input to the system message routine MSGCC, and are normally produced via the EFM macro.

An example of an EFM prototype is:

```
EFM    #DSPACER,SEVERITY=8,VERBOSITY=1
PHRASE 'DSP detected access code error for '
PHRASE (FDINAME)
PHRASE '.',END
```

#DSPACER is the unique message number for this message. The severity and verbosity are filters that can be used to control whether the message gets seen at all, and what form of verbosity it takes. In order for the verbosity to be meaningful, several EFM prototypes of different verbosities have to exist.

c. EFM Tables

For the Resource Manager, there are three levels of messages defined. The first is called the DSP level. DSP level messages have numbers in the range 1,000,000 to 1,999,999. The second level is the Resource Manager level, with message numbers in the range 2,000,000 to 2,999,999. The last level is the so called spooling system level with message numbers in the range 3,000,000 to 3,999,999.

The address of the default EFM prototype table for a given level is placed in the pseudo-register vector by the pseudo-register vector initialization routine for that level. The table addresses are found in the pseudo-registers RMEFMT1, RMEFMT2, and RMEFMT3. These pseudo-registers are 8 bytes long. The fist 4 bytes is for the address of a user supplied EFM prototype table, and the second 4 bytes is for the address of the default table. The user supplied table is scanned first, and then if no EFM prototype for the message is found there, the default table is searched.

7. MESSAGE EXPANSION SYSTEM ROUTINES

There are two basic routines at the user interface level for using the message expansion system. The first, RMMESSG, is used from within the program emitting a message to produce a CFM from a CFM prototype and the data referenced by it. The second is RMMSGEXP, which is used to produce an EFM from an EFM prototype and a CFM.

Routine: RMMESSG

Location: pseudo-register vector

Call Type: S(5)

Purpose: To generate a CFM.

Input parameters:	1	address of CFM prototype.
	2	4 character module ID.
	3	message route descriptor.
	4	message reference number.
	5	module invocation ID.
Return values:	none	
Return codes:	0	O.K.

Description:

A message header is generated by this routine from the parameters and a store clock (STCK) value. Then the CFM prototype is expanded into packets via the system message routine MSGCC, and the subtasking monitor service RMMSG is invoked to pass the message to the current message on-unit.

Routine: RMMSGEXP

Location: pseudo-register vector

Call Type: R(4,0)

Purpose: To generate and output an Expanded Form Message.

Input parameters:

1	address of CFM.
2	address of output subroutine.
3	severity filter.
4	verbosity filter.

Return values: none

Return codes:

0	O.K.
4	message was and end-of-file message.
8	EFM prototype not defined for the message number.

Description:

This routine will expand the CFM and EFM prototype from one of the EFM prototype tables into an EFM, and output it to the given subroutine, if the severity of the message from the EFM prototype table is greater than the severity filter.

This routine just selects the appropriate EFM prototype tables and calls MSGEXP.

Routine: MSGEXP

Location: DSPLCS

Call Type: s(5)

Purpose: Low level routine to generate and output an Expanded Form Message.

Input parameters:	1	address of CFM.
	2	address of output subroutine.
	3	message severity filter.
	4	message verbosity filter.
	5	address of optional and default EFM prototype tables.
Return values:	none	
Return codes:	0	O.K.
	4	no message found for given message number.
	8	internal error.

Description:

This routine will expand the CFM and and EFM prototype from either the optional or default table, and output it to the given subroutine, if the severity of the message from the EFM prototype table is greater than the severity filter.

Routine: CFMVALUE

Location: DSPLCS

Call Type: R(2,2)

Purpose: To return a particular value from a CFM.

Input parameters:	1	address of the keyword with trailing blank.
	2	address of the CFM.
Return values:	1	length of the value.
	2	address of the value.
Return codes:	0	O.K.
	4	keyword not found.
	8	invalid parameter.

Description:

This routine searchs the given CFM for the packet containing the given keyword, and returns the address and length of the associated value.

Routine: DISPLAY_CFM

Location: internal

Call Type: R(2,0)

Purpose: To output a CFM for RMMSGEXP in readable format when
the corresponding EFM prototype does not exist.

Input parameters:	1	address of CFM.
	2	address of output subroutine.
Return values:	none	
Return codes:	0	O.K.

Description:

This routine is used to get the message out in some form when the EFM prototype is not defined.

Routine: PRINT

Location: internal

Call Type: R(2,1)

Purpose: To make sure MSG routine's buffer does not overflow for DISPLAY_CFM.

Input parameters:	1	address null message.
	2	address MSG routine buffer control block.
Return values:	1	address of null message.
Return codes:	0	O.K.

Routine: MSGINIT

Location: internal

Call Type: R(3,0)

Purpose: To build a message operand table for MSGEXP.

Input parameters:	1	address of operand table.
	2	length of opernad table.
	3	address of the CFM.
Return values:	none - operand table filled in.	
Return codes:	0	O.K.
	4	table overflow.
	8	invalid CFM.

Description:

This routine builds the operand table which is used by MSGSCON to get the value for a given keyword.

Routine: EFMFIND

Location: internal

Call Type: R(3,2)

Purpose: To find the EFM prototype in the EFM prototype table.

Input parameters:	1	address table directory.
	2	message number.
	3	verbosity filter.
Return values:	1	address of the EFM prototype.
	2	message severity.
Return codes:	0	O.K.
	4	message number is valid, but verbosity too high.
	8	no EFM prototype for the message number.

Description:

When the directory contains multiple messages with the same number, the address returned will be that of the message with the highest verbosity less than or equal to the given filter value. When no message is found whose verbosity is acceptable to the filter, EFMFIND fails with a return code of 4.

Routine: MSGSCON

Location: internal

Call Type: R(2,1)

Purpose: To evaluate EFM operands.

Input parameters:	1	address of EFM operand.
	2	address of CFM operand table.
Return values:	1	address of the operand value.
Return codes:	0	O.K.
	4	operand not in table.

Description:

MSGCC calls the routine instead of its standard routine to evaluated S-type constants.

Routine: MSGOPND

Location: internal

Call Type: R(3,2)

Purpose: To find the value for a given name in the CFM operand table.

Input parameters:	1	address of CFM operand table.
	2	address of opernad name.
	3	length of opernad name.
Return values:	1	address of value.
	2	length of value.
Return codes:	0	O.K.
	4	opernamd is not in the table.

Description:

This routine is called by MSGSCON to search the operand table after it has extracted the operand name.

8. THE PRINTER CARRIAGE CONTROL ROUTINES

The printer carriage control routines are a collection of routines designed to handle output to a line printer device, with either machine or logical carriage control or a mixture of both. The routines produce an optimal list of CCW operations (for speed) for a given carriage control tape definition, and do page and line accounting.

These routines are going to require more work in order to make them more flexible. Currently they are table driven, but the tables are selected solely by device type. Things like carriage control tape, lines per page, page width etc. are not easily changable, or not changable at all.

Routine: CCINIT

Location: DSPLCS

Call Type: S(5,1)

Purpose: To initialize an invocation of the CC routines.

Input parameters:	1	address of 8 character device type.
	2	address of output subroutine.
	3	a parameter for the output subroutine.
	4	address of an accounting routine.
	5	a parameter for accounting routine.
Return values:	1	address of Printer Control Block (PCB).
Return codes:	0	O.K.
	8	invalid type.
	12	external error (get space).

Description:

This routine allocates and initializes a PCB for a printer of the given type.

Routine: CCOPEN

Location: DSPLCS

Call Type: R(1,0)

Purpose: To open an invocation of the CC routines.

Input parameters:	1	address of the PCB.
Return values:	none	
Return codes:	0	O.K.
	8	parameter error.
	12	external error (unit check).
	16	internal error (accounting).
	20	wait interrupted.

Description:

This resets various fields to the initial values and outputs a page skip so that things start out at a known place.

Routine: CCWRITE

Location: DSPLCS

Call Type: R(4,0)

Purpose: To write a line on the printer.

Input parameters:	1	address of PCB.
	2	address of line with carriage control.
	3	length of the line.
	4	address of modifier word. Only @MCC, @CC, @NOCC are checked.
Return values:	none	
Return codes:	0	O.K.
	4	page limit exceeded.
	8	parameter error.
	12	external error (unit check).
	16	internal error (accounting).
	20	wait interrupted.

Description:

This routine interprets the carriage control, and calls the output routine if necessary. There may not be a one to one correspondence between calls to CCWRITE and calls to the output subroutine. CCWRITE will in general buffer a line so that machine carriage control on the next line may be interpreted correctly.

When the output subroutine is called, it is called as an S-type routine as follows:

Input parameters:	1	line to print.
	2	halfword length of line.
	3	output routine parameter given to CCINIT.
	4	CCW operation code to use.
	5	flag, 0 - normal call, 1 - flush buffers.

When the accounting routine is called, it is called as an R-type routine as follows:

Input parameters:	1	accounting routine parameter given to CCINIT.
	2	accounting item. 2 - pages, 3 - lines.
	3	value of increment.
	4	flag, #TRUE - increment, #FALSE to check maximum.

Routine: CCCONT

Location: DSPLCS

Call Type: R(3,0)

Purpose: To control some things.

Input parameters:	1	address of PCB.
	2	address of control command.
	3	length of control command.
Return values:	none	
Return codes:	0	O.K.
	8	parameter error.
	12	external error (unit check).

Description:

Currently the following control commands are accepted:

FLUSH - force CCRTNS to flush buffers.

CCTAPE=hexaddress - change the carriage control tape definition to that at hexaddress. See the PTRTBLS assembly for details.

DEVICETYPE=Ý1403|PTRX¨ - change the device type. This currently only used to change between the different modes of the printronix printers. These printers can emulate a 1403, or can work in plot mode where the lines per page increases by a factor of 12.

Routine: CCCLOSE

Location: DSPLCS

Call Type: R(1,0)

Purpose: To close an invocation of the CC routines.

Input parameters:	1	address of PCB.
Return values:	none	
Return codes:	0	O.K.
	8	parameter error.
	12	external error (unit check).
	16	internal error (accounting).
	20	wait interrupted.

Description:

This will cause all buffered lines to be emitted from the CC routines and will cause the output subroutine to be called with a "flush" indication.

Routine: CCRELE

Location: DSPLCS

Call Type: R(1,0)

Purpose: To terminate an invocation of the CC routines.

Input parameters:	1	address of PCB.
Return values:	none	
Return codes:	0	O.K.
Return code:	8	parameter error.
	12	external error (free space).

Description:

This routine releases the PCB.

Routine: MCCRTN

Location: internal

Call Type: R(3,0)

Purpose: To handle lines with machine carriage control.

Input parameters:	1	address of PCB.
	2	address of line with carriage control.
	3	length of the line.
Return values:	none	
Return codes:	0	O.K.
	4	page limit exceeded.
	8	parameter error.
	12	external error (unit check).
	16	internal error (accounting).
	20	wait interrupted.

Description:

This is an internal routine to handle lines with machine carriage control.

Routine: LCCRTN

Location: internal

Call Type: R(3,0)

Purpose: To handle lines with logical carriage control.

Input parameters:	1	address of PCB.
	2	address of line with carriage control.
	3	length of the line.
	4	address of modifier word. Only @CC and @NOCC are checked.
Return values:	none	
Return codes:	0	O.K.
	4	page limit exceeded.
	8	parameter error.
	12	external error (unit check).
	16	internal error (accounting).
	20	wait interrupted.

Description:

This is an internal routine to handle lines with logical carriage control.

9. THE PLOT ACCOUNTING ROUTINES

The Plot Accounting (PAC) routines are a collection of routines that decode the data lines from a plotfile encoded using the UBC-extended University of Michigan plotfile format. Although the name of these routines implies that their primary purpose is to perform accounting, they are in fact the only routines that actually understand the format used in MTS plotfiles, decoding it into plotter moves, draws, and pen changes.

The PAC routines are called both by the spooling and unspooling plot DSPs to interpret plot data lines. The spooling DSP calls the PAC routines to decode the plotfile for accounting purposes only - the lines that are spooled are the original lines, intact. The unspooling DSP then has to use the PAC routines to decode the plotfile again, to drive the plotter.

The PAC initialization routine expects a coordinate output routine, a pen change routine, an accounting routine, and associated control blocks to be passed to it. For the spooling phase, a null coordinate output routine, pen change routine, and associated control block are given; for the unspooling phase, a null accounting routine and control block are passed. When proper RM accounting is implemented (i.e. charges are made at unspooling time, rather than during the spooling phase), the unspooler will give the PAC routines an accounting routine and control block as well.

During the processing of a plotfile, the PAC routines will call the supplied routines, if available, with each pen change and coordinate point that is moved or drawn to; the associated control block that is given will be passed along in the call. Currently, the accouting routine called during spooling is the Ctlacc routine in the "shared" control routines (CTL routines) module, whereas the coordinate output and pen change routines are entry points to drive the plotter in the PLTDSP.

Note that the alphabet file accessed by the PAC routines is permitted READ PKEY=*RMGR. This means that the spooling DSP must perform a Pushpkey/Poppkey operation for each call to a PAC routine.

Routine: PACINIT

Location: DSPLCS

Call Type: S(6,1)

Purpose: To initialize an invocation of the PAC routines.

Input parameters:	1	address of 8 character device type
	2	address of coordinate output routine
	3	address of pen change routine
	4	address of control block for coordinate/pen change routines
	5	address of an accounting routine
	6	address of control block for the accounting routine
Return values:	1	address of PAC control block
Return codes:	0	O.K.
	12	space acquisition or alphabet file error.

Description:

This routine allocates and initializes a control block for the PAC routines. It also gets and sets up the alphabet file used to read font data from, and initializes an invocation of the symbol/dashed line expansion module.

Routine: PACOPEN

Location: DSPLCS

Call Type: R(1,0)

Purpose: To open an invocation of the PAC routines.

Input parameters: 1 address of PAC control block

Return values: none

Return codes: 0 O.K.

Description:

This routine resets various fields to initial values.

Routine: PACWRITE

Location: DSPLCS

Call Type: S(5,1)

Purpose: To process a single plotfile data line.

Input parameters:	1	address of PAC control block
	2	address of data
	3	address of data length
	4	address of sequence check switch
	5	address of account usage switch
Return values:	1	result code
Return codes:	0	O.K.
	4	notify code (see result code)
	8	error (see result code)
	12	external error (see result code)
	20	wait aborted

Description:

This routine ensures that the length of the data line is valid, and then calls one of several subsidiary routines to decode the data, depending on the record type contained in the line.

The sequence check switch indicates whether or not a check should be performed to ensure that the record being written is not out of sequence, e.g. to make sure that a PBGN record is passed before a PCOD record. Currently such checks are done only during the spooling phase with the user's plotfile proper, as there are many extra records generated by the plot processor during unspooling.

The accounting switch indicates whether or not the current record is to be accounted for; things like headers, trailers, and frame numbers are not charged to the user.

The result code returned is described by the following Plus type, contained in Copy:Rmgr*Sql:

```
type Plotacc_Result_Type is (
   /* rc 0 */
      Plotacc_Ok,
   /* rc 4 */
      Plotacc_Start_Frame, Plotacc_End_Of_Frame,
      Plotacc_Pen_Change, Plotacc_Pen_Position,
   /* rc 8 */
      Plotacc_Missing_Pbgn, Plotacc_Missing_Pend,
      Plotacc_Illegal_Scale_Factor, Plotacc_Invalid_Coordinate,
      Plotacc_Invalid_Pcod_Length, Plotacc_Unsupported_Colour,
      Plotacc_Psalph_Parm_Error, Plotacc_Bad_Length,
      Plotacc_Unknown_Record,
   /* rc 12 */
      Plotacc_Plottime_Exceeded, Plotacc_Plotpaper_Exceeded,
      Plotacc_Plotpens_Exceeded, Plotacc_Other_Error,
   /* rc 20 */
      Plotacc_Wait_Aborted);
```

Routine: PACCNTRL

Location: DSPLCS

Call Type: R(3,0)

Purpose: To perform control operations.

Input parameters:	1	address of PAC control block
	2	address of control command
	3	control command's length
Return values:	none	
Return codes:	0	O.K.
	4	invalid control command

Description:

The following control commands are currently accepted:

SCALE - Gives the scale factor to be applied to subsequent coordinates, i.e. all coordinates will be multiplied by this number.

RESET_ALPHABET - Gives the name of the alphabet that is to become the current alphabet. If not already defined, the given alphabet will be read in from the alphabet file.

SUPRESS_PENS - This causes all further pen changes to be ignored. This option is used when monochrome jobs are being plotted.

Routine: PACSENSE

Location: DSPLCS

Call Type: S(6,0)

Purpose: To have the PAC routines return information.

Input parameters:	1	address of PAC control block
	2	address of sense command line
	3	address of sense command's length
	4	address of region to return information in
	5	address of length of information returned
	6	address of maximum length of returned information allowed
Return values:	none	
Return codes:	0	O.K.
	4	requested information not found.

Description:

The following sense requests are currently accepted:

SCALE - Returns the current scaling factor being applied to plot coordinates.

CLOSING_VALUES - Returns some useful values that the spooling DSP needs to set the papersize, pentype, and frames attributes of a job. The pen down distance is also returned to determine if the the job is an "empty" one, i.e. to determine if the job should be cancelled.

RIGHT_MARGIN - Returns the right margin value off of the last PEND plot record processed.

Routine: PACCLOSE

Location: DSPLCS

Call Type: R(2,0)

Purpose: To close an invocation of the PAC routines.

Input parameters:	1	address of PAC control block
	2	address of returned closing message string
Return values:	none	
Return codes:	0	O.K.
	4	error - missing PEND record

Description:

This will release most alphabet-related storage, and possibly build a "closing message" suitable for sending to the user after spooling a plot.

A return code of 4 occurs only during spooling, to indicate that the last record processed was not a PEND record. If necessary, the calling routine may generate one, calling PACWRITE with it, and then calling this routine again.

Routine: PACREL

Location: DSPLCS

Call Type: R(1,0)

Purpose: To terminate an invocation of the PAC routines.

Input parameters: 1 address of PAC control block

Return values: none

Return codes: 0 O.K.

Description:

This routine releases any remaining alphabet storage, shuts down the alphabet file, closes the symbol/dashed line expansion module, and deallocates the PAC control block.

Routine: PAC_xxxx

Location: internal

Call Type: R(3,1)

Purpose: To interpret plotfile records.

Input parameters:	1	address of plot record data
	2	address of PAC control block
	3	sequence check switch
Return values:	1	result code (see PACWRITE description for details)
Return codes:	0	not used.

Description:

These routines are the actual record decoding routines, called by the PACWRITE entry. Each routine interprets a single plotfile record type, and takes care of calling the coordinate/pen/accounting routines with each pen change or coordinate point, when necessary.

The entry points are PAC_PBGN, PAC_PEND, PAC_PCOD, PAC_PSYM, PAC_PPEN, PAC_PALP, PAC_PDH1, and PAC_PDH2.

The PAC_PPEN routine may call the pen selection routine and/or the accounting routine when a different pen from the current one is requested. The pen selection routine is called as an R-type routine as follows:

Input parameters:	1	pen number
	2	address of the output routine control block

When the accounting routine is called, it is called as an Rmacctng_Type, described in the Plus library Copy:Rmgr*Sql.

Routine: PAC_ACCI, PAC_ACC, PAC_ACCT

Location: internal

Call Type: R(1,0), R(4,0), R(1,1)

Purpose: To perform coordinate accounting for one record.

Description:

PAC_ACCI, PAC_ACC, and PAC_ACCT are initialization, accounting, and termination routines for record-level coordinate accounting. During PCOD, PSYM, and PDH2 record processing, these accounting routines are called for each coordinate processed, to maintain distance travelled measurements, pen up and down motion counts, and maximum X and Y values for that one record. When processing for that record is finished, the termination entry PAC_ACCT updates the global values with the values from the record just processed.

A by-product of the PAC_ACC routine is to call the output subroutine and/or the accounting routine with the coordinate pair it just received. When the coordinate output routine is called, it is called as an R-type routine as follows:

Input parameters:	1	X coordinate
	2	Y coordinate
	3	pen motion (move or draw)
	4	address of the output routine control block

When the accounting routine is called, it is called as an Rmacctng_Type, described in the Plus library Copy:Rmgr*Sql.

Routine: PACAINIT, PACAFREE, PACSYM, PACDSHPR, PACDSH

Location: internal

Call Type:

Purpose: To generate dashed lines and symbols.

Description:

This set of routines comprise the dashed line and symbol expansion module; the routines have been written in Assembler due to their high floating-point instruction content.

PACSYM will, given a character string, generate the moves and draws necessary to plot the string in whatever font is passed to it; PACDSH will generate a dashed line given two endpoints, and a description of what the dashed line should look like (in the form dash1, space1, dash2, space2, sent via the PACDSHPR routine).

Both routines will call an output routine with each coordinate they generate; this output routine, together with a control block, is passed to the expansion module during initialization with the PACAINIT routine. As it turns out, the output routine is the PAC_ACC entry point in the Plus portion of the plot accounting routines.

PACAFREE is called to release storage acquired for expansion purposes. The general calling sequence is therefore of the form:

```
                PACAINIT
                   .
      PACSYM       or        PACDSHPR
                             PACDSH
                   .
                   .
           (possibly more calls)
                   .
                   .
                PACAFREE
```

10. CHARACTERISTICS OF SOME DSPS

DSP name: File DSP

Transfer vector name: FILEDSP

DSP type(s): FILE, SPOOL

Maximum I/O length: 32767

Description:

The file DSP is an interface to the file system. It supports both line and sequential files. The file DSP supports indexed operations on both types of files, but for sequential files, the application must keep track to the "line number" for the indexed operations. the "line number" for sequential files is the "NOTE" information for the record.

The FDINAME accepted by the file DSP, is either a normal file name in internal form (between 5 and 16 characters long) or an extended file name. The extended file name is a 16 character file name (padded with blanks if necessary) followed by an 8 character catalog pointer. The total name length in this case must be exactly 24 characters.

The file DSP accepts the following access codes:

X'00000001' - read access.
X'00000002' - write expand access.
X'00000004' - write change access.
X'00000006' - read and write change access.
X'0000003F' - unlimited access

Additionally, the following may be specified with the above:

X'00000010' - destroy access.
X'00000020' - permit access.

The access code is checked against the permitted access to the file, is used to determine how the file should be locked, and is used for checking on the validity of subsequent operations.

The file DSP accepts the following control commands:

EMPTY - causes the file to be emptied, the line pointer is set back to the beginning.

FLUSH - causes all current buffers to be written.

REWIND - positions the line pointer back at the beginning of the file.

POINT=nnnn - causes the current line pointer to be set to nnnn.

CURRENTLINE=nnnn - same as POINT.

SIZE=ppp - sets the size of the file to ppp pages.

MAXSIZE=ppp - sets the maximum size of the file to ppp pages.

INCREMENT=nnn - sets the line number increment to nnn. The default is 1.

APPEND - set the line pointer just beyond last line in file.

The file DSP accepts the following sense commands:

CURRENTLINE - returns the current line pointer as a four byte binary number.

LASTREAD - returns the internal line number of the last line read as a four byte binary number.

LASTWRITE - returns the internal line number of the last line written as a four byte binary number.

SIZE - returns the size of the file in pages as a four byte binary number.

MAXSIZE - returns the current maximum size of the file in pages as a four byte binary number.

LINES - returns the current number of lines in the file as a four byte binary number.

FIRSTLINE - returns the internal line number of the first line in the file as a four byte binary number.

LASTLINE - returns the internal line number of the last line in the file as a four byte binary number.

MAXLINE - returns the length of the longest line in the file as a four byte binary number.

INCREMENT - returns the current line number increment as a four byte binary number.

The I/O modifiers that have meaning to the file DSP are @INDEXED, @SEQUENTIAL, @FWD, @BKWD, @NOTIFY.

DSP name: Printer DSP

Transfer vector name: PTRDSP

DSP type(s): 1403PTR,3203PTR,1443PTR,9700PTR

Maximum I/O length: 133

Description:

The printer DSP supports several types of line printers. Current the 1403, 1443, and 3203 are supported. The unit check routines do not support 3211 printers at the moment.

The FDINAME accepted by the printer DSP is a 4 character UMMPS device name.

The access code is not used for the printer DSP, and should be X'00000000'.

The printer DSP accepts the following control commands:

FLUSH - causes all buffered lines to be written.

PN - causes all lines to be translated for a PN print train.

TN - causes all lines to be translated for a TN print train.

ALA - causes all lines to be translated for an ALA print train.

UCB - causes all lines to be translated for the UBC 9700 character set.

DEFTRAN - cause all lines to be translated according to the default (which is the UBC 9700 character set at UBC).

NOTRAN - causes no translation to be done on the data lines.

TRUNCATE=nn - causes lines to be truncated at nn characters. This us useful for 9700s and devices that choke or complain about long lines.

- CCTAPE=cctapename - specifies the name of the carriage control tape to use for carriage control optimization (and accounting).

CANCEL - cause the current CCW chain if ant to be terminated, and all buffer lines to be discarded.

SKIP - causes output to be supressed, but accounting etc to be continued. All actions except output are continued.

NOSKIP - enables output again after a SKIP command had disabled it.

The printer DSP accepts the following sense commands:

POSITION - returns position of the last line written on the page as a four byte binary number.

The I/O modifiers that have meaning to printer DSP are @CC, @NOCC, @MCC, @BINARY, @NOTIFY.

Lines written @BINARY are not translated.

A notify code of 12 is returned for the first line on a new page. If the @NOTIFY modifier is set, this also causes a return code of 4 (the notify return code). This allows page counting to be done easily by the caller of the DSR. The psuedo register vector accounting routine RMACCTNG is also called for page and line accounting.

DSP name: Plotter DSP

Transfer vector name: PLTDSP

DSP type(s): PLOTTER

Maximum I/O length: 32767

Description:

The plotter DSP supports the two Houston Instruments plotters at UBC. The plotter DSP uses the FECPDSP to send data through the UBC Message Multiplexor to the front end device; since the intertask DSP is used in the communication path, this DSP must run under the subtasking monitor.

The FDINAME accepted by the plotter DSP is a 4 character Front End Processor device name.

The access code is not used for the plotter DSP, and should be X'00000000'.

Because the plotter DSP accepts many control commands that include binary and floating-point data, a variant Plus record type has been defined to make things easier, as follows (see the relevant library members in Copy:Rmgr*Sql for more detail):

```
type Pltdsp_Control_Type is
      record
      variant Pct_Id of (Pen_Id, Reset_Id, Scale_Id,
         Reset_Alphabet_Id, Suppress_Pens_Id, Accounting_Id,
         Setup_Id, Speed_Id, Transmit_Id, Flush_Id, Abort_Id,
         Cancel_Id) from
      case Pen_Id:
         Pct_Pen_Position is Positive_Integer,
         Pct_Pen_Colour is Plot_Pentype_Type
      case Reset_Id:
         Pct_Blankspace is Non_Negative_Short_Integer
      case Scale_Id:
         Pct_Scale_Factor is Short_Real
      case Reset_Alphabet_Id:
         Pct_Alphabet_Name is Plot_Alphabet_Name_Type
      case Accounting_Id:
         Pct_Accounting_Switch is Boolean
      case Setup_Id:
         Pct_Papertype is Plot_Papertype_Type,
         Pct_Papersize is Plot_Papersize_Type,
         Pct_Maxpens is Positive_Short_Integer
      case Speed_Id:
         Pct_Speed is Positive_Integer
      case Transmit_Id:
         Pct_Onoff is Boolean
      case Suppress_Pens_Id, Flush_Id, Abort_Id, Cancel_Id:
         /* No parameters for these options */
      end;
```

Plot processor or Pltling (task RM.PLT) functions that require plotter DSP control requests are all mapped from their external form into the above variant record, by the Pltling.

The Pltdsp accepts the following control options:

Pen_Id - Informs the DSP that a certain pen type is mounted in a given position of the plotter's pen holder.

Reset_Id - Causes the DSP to reorigin the pen to Y=0, and an X value that is an even number of inches or centimetres beyond the value that is passed to the DSP.

Scale_Id - Gives the scale factor to be applied to the current plot. The DSP just passes this information along to the plot accounting routines.

Reset_Alphabet_Id - Tells the DSP to make the specified alphabet the current one. The DSP just passes this command on to the plot accounting routines.

Accounting_Id - Causes the DSP to turn accounting on or off for further processing.

Setup_Id - Gives the DSP its initial operating parameters: what paper type, paper size, and number of pens its plotter has.

Speed_Id - Tells the DSP to drive the plotter at the specified speed, 1<=n<=9, 9=default speed=fastest speed.

Transmit_Id - Tells the DSP to turn coordinate output on or off for further processing. Transmission of coordinate data is turned off when the Pltling is reading through a plot in search of a given frame.

Suppress_Pens_Id - Tells the DSP to ignore any pen changes following this command; the DSP just passes this command along to the plot accounting routines. This option is used for plots which may originally have been multicoloured jobs, but were forced to be plotted as monochrome jobs by the user.

Flush_Id - Causes the DSP to flush any buffered plot codes it has.

Abort_Id - Causes the DSP to throw away any buffered plot codes it has.

Cancel_Id - Informs the DSP that the job it is processing has been cancelled. Any buffered plot codes are *not* thrown away, since the position of the plotter pen would be lost.

The plotter DSP accepts the following sense commands:

PEN_CONFIGURATION - Requests that the DSP return the current contents of the plotter's pen holder, as well as the current pen (or rather, its position), and a pen type that the DSP is waiting for, if any.

SCALE - Requests the DSP to return the current scaling factor being applied to plots. This information is retrieved from the plot accounting routines.

The plotter DSP does not recognize any I/O modifiers, as all records are written @BINARY to the FECPDSP.

The plot accounting routines, described elsewhere in this document, are used to decode the plot lines read from a spoolfile.

DSP name: Intertask DSP

Transfer vector name: ITSKDSP

DSP type(s): INTRTASK

Maximum I/O length: 4074

Description:

The intertask DSP is an interface to the UMMPS intertask facility.

The FDINAME accepted by the intertask DSP must be 12 characters long, and is composed of a 4 character arbitrary identifier for the network node to be created, followed by an 8 character network name. The intertask DSP creates the specified node on the specified network, creating the network if it does not already exist.

The intertask DSP uses subtasking, and must therefore run under the subtasking monitor.

The intertask DSP accepts the following access codes:

X'00000080' - receive access.
X'00000002' - send access access.
X'00000004' - notify access.
Or any combination of the above.

In addition, X'00000000', is taken as send and receive access.

The intertask DSP takes and returns an intertask header as the first 20 characters of each buffer. The application must fill in values for the arbid, and task number when sending a message. These may be set to zero to indicate that all arbids and/or all tasks on the network are to receive the message. The format of the header is as follows:

item	length	use
LINK	4	not used, for use by the caller.
SEQ#	4	sequence number of message received.
ARBID	4	arbitrary id of sender or receiver.
TIMEOUT	4	message time out value (not used).
TASK#	2	task number of sender or receiver.
TEXTLEN	2	length of text of message received.

The intertask DSP accepts the following control commands:

NOTIFICATION - causes notify access to be added to the node.

RECEIVE - causes receive access to be added to the node.

SEND - causes send access to be added to the node.

NOTIFICATIONONLY - causes the node access to be changed to just notification.

RECEIVEONLY - causes the node access to be changed to just receive.

SENDONLY - causes the node access to be changed to just send.

CLEAR - causes any buffered input messages to be discarded.

The intertask DSP accepts no sense commands.

The I/O modifier that has meaning to the intertask DSP is @NOTIFY.

DSP name: Intertask Line DSP

Transfer vector name: ITLNDSP

DSP type(s): ITLINE

Maximum I/O length: 4074

Description:

The intertask line DSP is another interface to the UMMPS intertask facility. This DSP makes a connection between 2 nodes.

The FDINAME accepted by the intertask DSP must be 18 characters long, and is composed of a 4 character arbitrary identifier for the network node to be created, followed by an 8 character network name, followed by a 4 character arbid of the node on the other end of the connection, followed by a 2 character task number for that node. The DSP creates the node on the specified network, creating the network if it does not already exist. Messages received from nodes other than the one specified are ignored, and all messages are sent to the node specified. The buffer in this case does not contain an intertask header.

The intertask line DSP uses subtasking, and must therefore run under the subtasking monitor.

The intertask line DSP accepts the following access codes:

X'00000080' - receive access.
X'00000002' - send access access.
Or a combination of the above.

In addition, X'00000000', is taken as send and receive access.

The intertask line DSP accepts the following control commands:

RECEIVE - causes receive access to be added to the node.

SEND - causes send access to be added to the node.

RECEIVEONLY - causes the node access to be changed to just receive.

SENDONLY - causes the node access to be changed to just send.

CLEAR - causes any buffered input messages to be discarded.

The intertask line DSP accepts no sense commands.

The I/O modifier that has meaning to the intertask line DSP is

@NOTIFY.

DSP name: Printronix DSP

Transfer vector name: PTXDSP

DSP type(s): PTRXPTR

Maximum I/O length: 133

Description:

The Printronix DSP supports the Ascii Printronix printers at UBC. The Printronix DSP calls on the Front End Communication Protocol DSP to send data through the UBC Message Multiplexor to the front end device.

The FDINAME accepted by the Printronix DSP is a 4 character Front End Processor device name.

Since the intertask DSP is used in the communication path, this DSP must run under the subtasking monitor.

The access code is not used for the Printronix DSP, and should be X'00000000'.

The printronix DSP accepts the following control commands:

FLUSH - causes all buffered lines to be written.

PX - causes all lines to be translated for the printronix printer.

DEFTRAN - cause all lines to be translated according to the default (which is the same a PX).

NOTRAN - causes no translation to be done on the data lines.

- CCTAPE=cctapename - specifies the name of the carriage control tape to use for carriage control optimization (and accounting).

CANCEL - cause the current CCW chain if ant to be terminated, and all buffer lines to be discarded.

SKIP - causes output to be supressed, but accounting etc to be continued. All actions except output are continued.

NOSKIP - enables output again after a SKIP command had disabled it.

The printronix DSP accepts the following sense commands:

POSITION - returns position of the last line written on the page as a four byte binary number.

The I/O modifiers that have meaning to Printronix DSP are @CC, @NOCC, @MCC, @BINARY, @NOTIFY.

Lines written @BINARY to the Printronix DSP are taken as lines to be written in plot mode. No translation is done on lines written @BINARY.

A notify code of 12 is returned for the first line on a new page. If the @NOTIFY modifier is set, this also causes a return code of 4 (the notify return code). This allows page counting to be done easily by the caller of the DSR. The psuedo register vector accounting routine RMACCTNG is also called for page and line accounting.

DSP name: Front End Communication Protocol DSP

Transfer vector name: FECPDSP

DSP type(s): FECP

Maximum I/O length: 254

Description:

The FECP DSP is used to communicate to devices attached to the UCB Front End Processors (FEPs). The Intertask DSP is used to send and received data to the device through the UBC message multiplexor. It must therefore run under the subtasking monitor.

The FDINAME accepted by the FECP DSP is a 4 character FEP device name followed by ".FECP"

The access code is not used for the FECP DSP, and should be X'00000000'.

The FECP DSP accepts the following control commands:

CANCEL - causes all buffered lines to be discarded.

BLOCKING=ON_OFF - controls whether or not output lines are blocked before they are sent to the Message Multiplexor.

The FECP DSP accepts no sense commands.

The I/O modifiers that have meaning to FECP DSP are @NOTIFY.

DSP name: VideoPrint DSP

Transfer vector name: VPRTDSP

DSP type(s): VIDEOPRT

Maximum I/O length: 240

Description:

The VideoPrint DSP supports two devices: a standard videotex decoder and a VideoPrint 5000 camera device. Both physical devices are attached to a UBCnet NIM and so are accessed through the VTP DSP.

Note: the status of this description is currently incomplete and may not accurately reflect the implementation.

The FDINAME accepted by the VideoPrint DSP is an N character symbolic name found in the symbol table under <basdevt>. The VideoPrint DSP checks for this name in the symbol table to retrieve the network address of the two NIM attached devices.

The access code is not used for the VideoPrint DSP, and should be X'00000000'.

This DSP is used by the VTXLING (task RM.VTX) for despooling videotex jobs and has be designed to buffer the ling as much as possible from dealing with the actual devices. Input to the VideoPrint DSP consists of 'Videotex Records' which the ling has no knowledge of; it simple copies them from the spool files. Videotex Records are generated and spooled by the *VIDEOTEX* DSP (VTEXDSP).

Videotex Records are defined in the Resource Manager Plus Source Library. Additional definitions needed for Videotex Records may be found in Plus Source Libraries VTX:ROUTINES*SQL and VTX:INTERNAL*SQL.

In addition to controlling two UBCnet attached devices the VideoPrint DSP performs the following important funtions:

1) Interacts with the device operator over the physical aspects of the job: when to load film, number of frames to print, size of film roll loaded, etc.

2) Manages the splitting of large jobs over multiple rolls of film when necessary.

3) Generates an identifying 'header' frame for each roll of film.

In the future this DSP will also be do the despooling side of the MTS accounting for videotex jobs.

The VideoPrint DSP currently does not support text control, commands, rather it accepts binary command records. These control commands can come from two sources: directly from the VTXLING or indirectly from the *VIDEOTEX* DSP which may imbed them in Videotex Records. In either case command records are the same. The definitions for these commands may be found in the Resource Manager Plus Source Library under VPRTDSP_Control_Types.

Like control command records the VideoPrint DSP currently only supports sense command records and returns sense result records. The definitions for these commands and result may be found in the Resource Manager Plus Source Library under VPRTDS_Sense_Types.

This DSP currently does not recognise any of the I/O modifiers. All data passed along to the VTP DSP is written @BINARY.

A. APPENDIX - COPY SECTIONS/CONTROL BLOCKS

Copy Section/Control Block Name: FDIB

Macro Library: RMGR:RMGR*M

File, Device or Intertask block.

```
FDIB     RDSECT
*
FDIBCBID DC    CL4'FDIB'    control block id for dumps
FDILINK  DS    A            link to next FDIB or a(0)
FDISPACE DS    A            space chain for DSP
FDINAME  DS    A            pointer to name of file, device,
*                           or intertask node
*                           (halfword count & string)
FDITYPE  DS    CL8          DSP type
FDIDSP   DS    A            address of DSP transfer vector
FDIFDCB  DS    A            address of file/device control block
FDIMODS  DS    XL8          modifier bits (MTS compatible)
*
*  FDIB SWITCHES
*
FDIOPEN  DS    X            true/on -> FDIB is open
*
FDILSTOP DS    X            last operation
*                           is one of:
FDILINIT EQU   0            0  = initialize
FDILOPEN EQU   4            4  = open
FDILREAD EQU   8            8  = read
FDILWRIT EQU   12           12 = write
FDILCONT EQU   16           16 = control
FDILSENS EQU   20           20 = sense
FDILCLOS EQU   24           24 = close
FDILRELE EQU   28           28 = release
*
FDIMAXLN DS    H            max length for device
FDILSTCL DS    A            last caller (GR14 of caller)
FDILSTRC DS    F            last return code (GR15 from DSP)
FDIACB   DS    A            address of accounting control blk
FDILEN   EQU   *-FDIB       length of FDIB
*
         RDSECT END
```

Copy Section/Control Block Name: RMMSGFMT

Macro Library: RMGR:RMGR*M

Resource Manager message format.

```
RMMSGFMT RDSECT
*
RMSGID   DS    F            Globally unique message ID
RMSGMOD  DS    CL4          ID of module sending message
RMSGINV  DS    F            Some kind of invocation ID
RMSGTIME DS    XL8          STCK value at time message sent
RMSGREF# DS    F            Message reference (sequence) no.
RMSGLEN  DS    H            Length of following CFM.
RMSGFMTL EQU   *-RMMSGFMT   Length of message header
RMSGTEXT DS    0XL256       Start of message operands
*
* Each message operand is a name-value pair with the format:
*  1. AL1(length of name)
*  2. C'name'
*  3. AL1(length of value)
*  4. X'value'
*
* Following the last name-value pair
* is the canonical-form message terminator X'00' .
*
         RDSECT END         End of message
```

Copy Section/Control Block Name: ITHEADER

Macro Library: RMGR:RMGR*M

Intertask message header.

```
ITHEADER RDSECT
*
* This dsect describes an intertask message as received or sent
* via the intertask DSP. the SEQ# and LEN fields have meaning
* only for messages received. the timeout value has meaning only
* for messages being sent, and the arbid and task# fields have
* meaning in both cases.
*
ITHLINK  DS    A            user link field
ITHSEQ#  DS    F            sequence number of message
ITHARBID DS    CL4          arbid of message sender/receiver
ITHTIMEO DS    F            message timeout value
ITHTASK# DS    H            task # of sender/receiver
ITHLEN   DS    H            length of message text
ITHTEXT  DS    0XL256       message text
         SPACE 2
ITHDRLEN EQU   ITHTEXT-ITHEADER length of header
```

Copy Section/Control Block Name: DSPMSGEQU

Macro Library: RMGR:RMGR*M

Message equates for DSP level messages.

```
*
* General messages.
*
#MNSPCER EQU   1001001      space allocation error
#MNSBTFL EQU   1001002      subtask creation error
*
* Device support program interface messages.
*
#DIBADPL EQU   1002001      parameter list is unaddressable
#DIBADPA EQU   1002002      bad parameter address
#DIBADP  EQU   1002003      bad parameter
#DIDUPL  EQU   1002004      attempt to allocate duplicate FDIB
#DINODSP EQU   1002005      DSP is not loaded
#DIBADSQ EQU   1002006      invalid operation sequence
*
* DSP messages
*
#DSPNXER EQU   1003001      file/device non-existant/unavailable
#DSPLKER EQU   1003002      error locking file
#DSPLKOP EQU   1003003      error from icopen
#DSPULKR EQU   1003004      error unlocking file
#DSPFLSR EQU   1003005      file system error
#DSPCTLR EQU   1003006      invalid control command
#DSPSNSR EQU   1003007      invalid sense command
#DSPBSYR EQU   1003008      device is busy
#DSPPERR EQU   1003009      parameter error
#DSPACER EQU   1003010      access code error
*
* Standard unit check messages
*
#MSUCR   EQU   1004001      command reject
#MSUIR   EQU   1004002      intervention required
#MSUIREM EQU   1004003      situation unrecoverable after ireq
#MSUBOCK EQU   1004004      bus out check
#MSUEQCK EQU   1004005      equipment check
#MSUDCK  EQU   1004006      data check
#MSUUCS  EQU   1004007      unusual command sequence
#MSUOR   EQU   1004008      overrun
#MSUQER  EQU   1004009      too many i/o operations (>1)
#MSUSER  EQU   1004010      invalid sense after unit check
#MSURTY  EQU   1004011      retry count exhausted error
#MSUPCK  EQU   1004012      parity check
#MSUIREJ EQU   1004013      initial rejection of command
#MSUIOP  EQU   1004014      immediate operation error
#MSUCHER EQU   1004015      channel error
#MSUUC   EQU   1004016      general purpose UC message
```

```
*
* Unique error messages from the UBC FECPDSP.
*
#FEPGTMSG  EQU 1005001      unknown return code from GTFEPDEV
#FEPWRMSG  EQU 1005002      error message from FEP
#FEPCMDERR EQU 1005003      unknown/unsupported command from FEP
*
* Unique error messages for printers
*
#PTRLDCK EQU   1006001      load check on printer
#PTRLPCK EQU   1006002      line position check on printer
```

Device Support Programs and the Device Support Program Interface

by

Ken Bowler

Computing Centre

UNIVERSITY OF BRITISH COLUMBIA
6356 Agricultural Road
Vancouver, B.C., Canada V6T 1W5

April 1981

Revised July 1982

INDEX

Table of Contents