MIMER/SH
REFERENCE MANUAL

Version 3.2
March 1984

MIMER/SH REFERENCE MANUAL

Page

1. INTRODUCTION ---------------------------- 1.01

2. INFORMATION HANDLING ------------------- 2.01

2.1 Output buffering ------------------------ 2.03
2.2 Picture specification ------------------- 2.03
2.3 Field specification ------------------------ 2.04
2.4 Current picture ---------------------------2.05

3. MIMER/SH ROUTINES ------------------------ 3.01

3.1 Routines summary in alphabetical order
with parameters ------------------------ 3.01
3.2 Routines summary grouped by function ------ 3.02
3.3 Description of routines --------------- 3.03

3.3.1 CLFSH2 --------------------- 3.03
3.3.2 CLPSH2 --------------------- 3.04
3.3.3 CLSSH2 --------------------- 3.05
3.3.4 CONSH2 --------------------- 3.06
3.3.5 DEFSH2 --------------------- 3.07
3.3.6 DIASH2 --------------------- 3.09
3.3.7 ENDSH2 --------------------- 3.10
3.3.8 GETSH2 --------------------- 3.11
3.3.9 GINSH2 --------------------- 3.12
3.3.10 GSTSH2 --------------------- 3.14
3.3.11 INASH2 --------------------- 3.15
3.3.12 INISH2 --------------------- 3.16
3.3.13 INPSH2 --------------------- 3.17
3.3.14 JMPSH2 --------------------- 3.18
3.3.15 MSGSH2 --------------------- 3.19
3.3.16 PMTSH2 --------------------- 3.20
3.3.17 PROSH2 --------------------- 3.21
3.3.18 PUTSH2 --------------------- 3.22
3.3.19 RDFSH2 --------------------- 3.23
3.3.20 RMPSH2 --------------------- 3.24
3.3.21 SELSH2 --------------------- 3.25
3.3.22 SETSH2 --------------------- 3.26
3.3.23 WRFSH2 --------------------- 3.28
3.3.24 WRPSH2 --------------------- 3.29

3.4 Description of the status parameter ------- 3.30

3.4.1 STATUS parameter from routine
DIASH2 --------------------- 3.30
3.4.2 STATUS parameter from routine
RDFSH2 -------------------- 3.30
3.4.3 STATUS parameter in user written
check routines -------------------3.31
3.4.4 Using the STATUS parameter for
application variables ------------------3.31
3.4.5 STATUS parameter in unit check
routines - 3.32
3.4 6 STATUS parameter in exit routines -------3.33
3.5 Logging facility -------------------------3.34

Page

4. STRUCTURE DESCRIPTION ---------------------- 4.01

4.1 Information checking ------------------ 4.01

4.1.1 Check codes -----------------------------4.01
4.1.2 Validation ----------------------------4.02
4.1.3 Check routines -----------------4.03

4.1.4 Unit check routines ---------- 4.03

4.2 Field display ------------------------- 4.05

4.2.1 Display editing pattern --------- 4.05
4.2.2 Display editing routine -------- 4.06

4.3 Exit functions ------------------------ 4.07

4.3.1 Exit routines ------------------- 4.07

4.4 Read control and picture modes ---------- 4.08
4.5 Repositioning the cursor among the fields of the picture ---------------------------- 4.10
4.6 Field enter status ------------------------ 4.10
4.7 Editing when entering fields ---------------- 4.10
4.8 Message areas ---------------------------- 4.11
4.9 Prompters -------------------------------- 4.12
4.10 Helptexts ------------------------------- 4.12

APPENDICES

A Glossary

B Terminal type table

C Picture definition language

D Limitations

E Error messages from MIMER/SH compiler

F MIMER/SH editor error codes

G MIMER/SH routine error messages

H Restart of internal space area

I Terminal independence and terminal read modes

1. INTRODUCTION

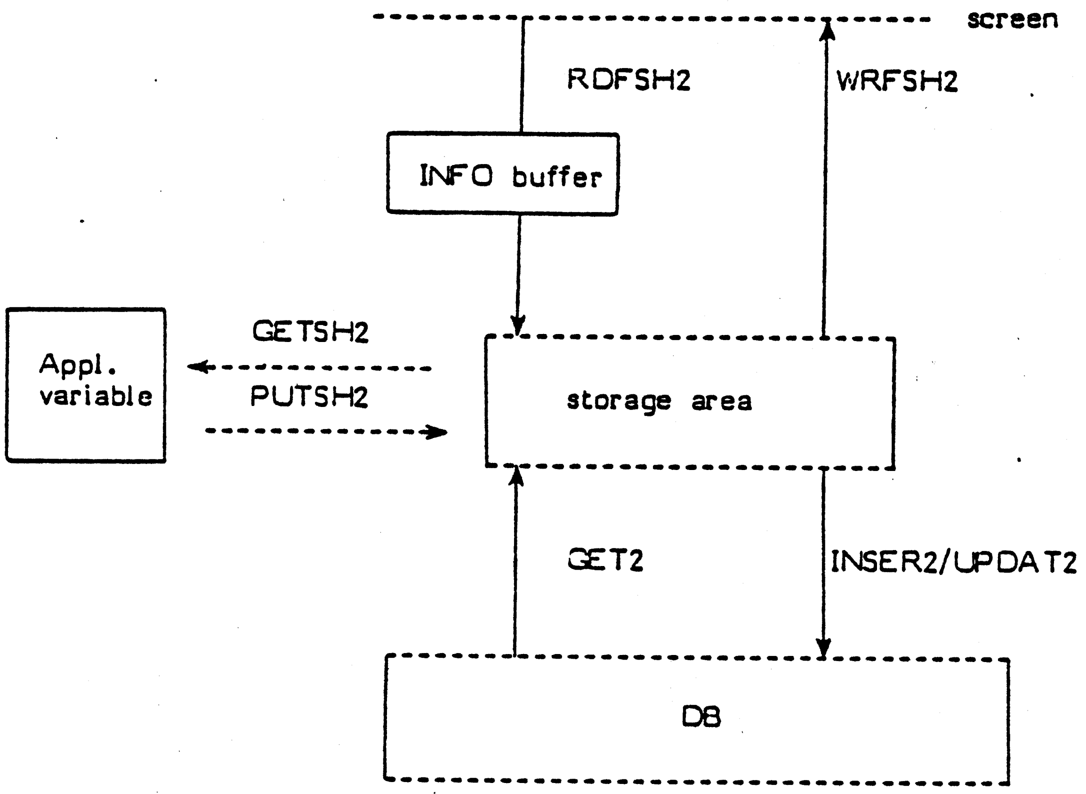
This section of the manual provides the detailed information about checking, editing and validation necessary for users to write their own comprehensive routines for information handling and screen manipulation.

Several appendices are provided, including a description of the picture definition code which the Editor uses for picture development. Error messages are also described in appendices.

2. INFORMATION HANDLING

MIMER/SH stores all information read from the screen in a storage area. When writing to the screen information is written from this storage area.

The storage area is normally hidden from the user i.e. it is internal to MIMER/SH.



Each field has a storage area connected to it. When fields are read from the screen using RDFSH2 the information is stored in the storage area for each field. If check routines are used to check entered information, the INFO buffer will contain the read information until the check routine is exited. On exit from the check routine you can specify whether to store the information or not. When writing fields to the screen using WRFSH2 the information written will be the information currently in the storage area for each field.

Information can be fetched directly from a database table into the storage area for a field by a call to the MIMER/DB routine GET2. Information can also be stored into a table by a call to INSER2 or UPDAT2 routines. To set up a path between the fields of a picture and the columns of a data base table the routine PROSH2 is used.

To move information between a field and an application variable the routines GETSH2 and PUTSH2 are used.

It is possible to handle the storage area directly from the application program. Here the user declares the storage area as a ordinary variable in the application program and uses the picture initiation call CONSH2 to connect the field to the user defined storage variable.

2.1 Output buffering

MIMER/SH buffers everything that is to be written on the screen in an output buffer. The output buffer is written to the screen under the following conditions:

1. The output buffer is full.
2. Input from the user is requested.
3. A call to one of the routines DIASH2, MSGSH2 or ENDSH2.

2.2 Picture specification

A picture specification is a character string variable or constant containing the picture name padded with blanks to 8 characters length. The picture name may only contain capital letters and digits.

EXAMPLE:**

```
        WRPSH2 ("CUST    ")
        Refers to the picture CUST
```

*************************************** EXAMPLE END

2.3 Field specification

Fields are referenced by their names. A field name must be unique within a picture. The specification is a character string variable or constant ending with a blank character.

Two types of field specifications are used. These are referred to as field specification type 1 and type 2.

Type 1 field specification may only refer to one field and may be preceded by a picture declaration consisting of a picture name followed by a colon.

Type 2 field specification may refer to several fields specifying:

```
     o    a range,
     o    a list of single fields
or   o    a start field an number of fields.
```

However, a mixture e.g. of range and single fields cannot be specified.

Four system defined field names may be used:

$FIRST	denoting the first field of a picture.
$LAST	denoting the last field of a picture.
*	denoting all fields of a picture (only as type 2).
$INFO	denoting field information buffer on input to check routine. This is not an 'ordinary' field and can only be used with the routine SELSH2.

EXAMPLE: **

Type 1 "CUSTREQ:CITYCODE "
This specification refers to the field CITYCODE in the picture CUSTREQ.

Type 2 "ARRIVDAT;3 "
This specification refers to the field ARRIVDAT and the two fields following it, in read order.

"ARRIVDAT-$LAST "
This specification refers to all fields from and including ARRIVDAT to the last field of the picture, in read order.

"TODAYTE,ARRIVDAT,DEPARDAT "
This specification refers to the three fields given.

*** - EXAMPLE END

2.4 Current Picture

When specifying fields of a picture, a current (or default) picture may be used. This default is set by the routines (INPSH2, RDFSH2 and RESSH2, i.e. after a call to one of these routines the current picture will be the picture referred to in the call to the routine.
The current picture may also be set by the routine SETSH2.

```
EXAMPLE **************************************************

                    INPSH2 ("CUST     ");
                    INASH2 ("CUST     ");

--                  The current picture is now set to 'CUST----'
--                  If fields are referred to (type 2 field spec)
--                  without giving an explicit picture
--                  specification, the fields are assumed to be in
--                  the picture 'CUST'.

                    GETSH2 ("NAME ",NAME, "I"); -- Will look for
--                  the field CUST:NAME

************************************************** EXAMPLE END
```

3. MIMER/SH ROUTINES

3.1 Routines summary, in alphabetic order with parameters

CLFSH2	picture, fields
CLPSH2	type, picture, fields valid for type: "T ","TF" or "F "
CLSSH2	type, row1, row2, col1, col2 valid for type: "WH","R " or "RC"
CONSH2	field1, field2, type valid for type: "V" or "F"
DEFSH2	type, string valid for type: "MESS" or "PMT "
DIASH2	row, col, text, length, answer, length, status
ENDSH2	
GETSH2	field, variable, type valid for type: "C" or "I"
GINSH2	type, par1, par2, par3 valid for type: "CP","CF","FI","EA","EP", or "EF".
GSTSH2	no, string, length
INASH2	picture
INISH2	term.type, file
INPSH2	picture
JMPSH2	field, offset
MSGSH2	area, string, lengtth
PMTSH2	name, code
PROSH2	tid, lop, col, acc, ba, field, len
PUTSH2	field, variable, type valid for type: "C" or "I"
RDFSH2	picture, fields, status
RMPSH2	picture
SELSH2	tid, lop, col, rop, ba, field, len
SETSH2	type, str.par1, str.par2, int.par3, int.par4 valid for type: "PO","PE","PN","CP","MO","FX",
WRFSH2	picture, fields
WRPSH2	picture

3.2 Routines summary, grouped by function

Initiation routines

INISH2	--	initiates MIMER/SH
INPSH2	--	initiation of picture, start
INASH2	--	initiation of picture, end
DEFSH2	--	additional definitions for the session

Termination routines

RMPSH2	--	terminates picture
ENDSH2	--	ends MIMER/SH

Screen manipulation

Reading from screen:

DIASH2	--	reads string from specified position
PMTSH2	--	reads with prompter and returns number for predefined answer
RDFSH2	--	reads specified fields of picture, checks information and stores it

Writing to the screen:

DIASH2	--	writes string to specified position
MSGSH2	--	writes message string to a message area
PMTSH2	--	writes prompter to message area (and waits for answer)
WRPSH2	--	writes picture texts to the screen
WRFSH2	--	writes specified fields from storage to the screen

Clearing the screen:

CLFSH2	--	clears specified fields from the screen and in storage
CLPSH2	--	clears specified text of picture and/or fields on the screen
CLSSH2	--	clears specified positions on the screen

Controlling reading/writing:

GINSH2	--	gets status information
SETSH2	--	sets read/write parameters
JMPSH2	--	supports "jumps" in field read sequence order

Information handling

CONSH2	--	redefines storage of a field to coincide with storage of another field or program variable (at initiation only)
GETSH2	--	fetch contents of a field to a program variable
PUTSH2	--	store from program variable to field storage
PROSH2	--	set up information transfer path between field storage and DB columns, for DB storage/retrieval
SELSH2	--	set constraints on columns of DB table v.s. contents of a field, for DB retrieval
GSTSH2	--	get a predefined string from definition file into program variable.

3.3 Description of routines

3.3.1 CLFSH2

Routine:	CLFSH2
Purpose:	Clear specified fields of the screen.
Format:	CLFSH2 pic, fields

Arguments:	PIC	C8	in	Picture specification
	FIELDS	Cx	in	Field specification type 2

Description: The specified fields of the picture are cleared from the screen. Clearing means writing field positions with the general fill character. The storage area associated with each field is filled with the fill character defined for the field.

Example: CLFSH2 ("CUSTREQ ","* ");

All fields of the customer requirement picture will be cleared on the screen, i.ê. if dot is defined as general fill character, dots will be written in the positions of each field. The storage areas for the fields of the picture are cleared with their respective fillers.

Note: To clear fields at the correct positions on the screen picture offset should be the same as when fields were written.

3.3.2 CLPSH2

Routine:	CLPSH2			
Purpose:	Clear picture texts and/or fields of the picture on the screen.			
Format:	CLPSH2 code , pic , fields			
Arguments:	CODE	C2	in	Mnemonic for clear action valid values 'T ','F ' or 'TF'.
	PIC	C8	in	Picture specification
	FIELDS	Cx	in	Field specification type 2

Description: If CODE is specified as T the texts of the picture specified by PIC are cleared, i.e. blanked from the screen.
If CODE is specified as F the fields specified by FIELDS are cleared from the screen.
If CODE is specified as TF the two above actions are performed, clearing texts first.

Example:

```
CLPSH2 ("F ","CUSTREQ ","* ");
```

All fields of the CUSTREQ picture are cleared from the screen, i.e. overwritten with blanks. The storage areas of the fields are not altered.

Note: To clear the correct positions on the screen picture offset should be the same as when texts or fields were written.

3.3.3 CLSSH2

Routine:	CLSSH2
Purpose:	Clear specified parts of the screen.
Format:	CLSSH2 code , row1 , row2 , col1 , col2

Arguments:

CODE	C2	in	Mnemonic code defining parts of screen to clear. Valid values 'WH','R ', 'RC'.
ROW1	Iw	in	First row of range when 'R ' or 'RC'.
ROW2	Iw	in	Last row of range when 'R ' or 'RC'.
COL1	Iw	in	First column of range when 'RC'.
COL2	Iw	in	Last column of range when 'RC'.

Description: The screen is cleared according to the clear code:
'WH' denotes the whole screen
'R' denotes the range of rows given by ROW1 and ROW2.
'RC' denotes the area given by ROW1,ROW2,COL1 and COL2. Row an column are in range 1 to 24 and 1 to 79, respectively.

Example:

```
CLSSH2 ("WH",0,0,0,0);
```

Clears the whole screen.

```
CLSSH2 ("RC",14,16,20,40);
```

Clears the part of the screen inbetween and including 14 and 16 and column 20 and 40.

3.3.4 CONSH2

Routine:	CONSH2
Purpose:	Connect the storage area of a field to the storage area of another field or a program variable.
Format:	CONSH2 field , storage , type

Arguments:

FIELD	Cx	in	Field specification type 1. This is the field for which the storage is to be connected.
STORAGE	Cx/ref	in	Either a field specification of type 1 or a program variable depending on type specified.
TYPE	C1	in	Type of connection. Valid values 'F' or 'V'.

Description: If type is 'F' the field specified by FIELD gets its storage area connected to the storage area of the field specified in STORAGE. I.e. the field specified by FIELD will share storage area with the field specified by STORAGE. If type is 'V' the program variable STORAGE will be used as storage area for the field information.

Note:

1) Connection is only a convenience and is not needed for standard applications.
2) A connection for a field can only be made in between a call to the picture initiation routine INPSH2 and the picture storage allocation routine INASH2.
3) When connecting a field to a field of another picture this picture must already be initiated through INPSH2 and INASH2 calls.
4) When connecting to a program variable make sure to define your variable to sufficient size.
5) Connection to a program variable does not work on all machine brands; see machine dependent information.

Example:

```
CONSH2 ("CUSTREQ :TODAYTE ","CUSTREQ:DEPARTIM ","F");
```

Here the storage area of the field TODAYTE is the same as the storage area of the field DEPARTIM. This means that if you write the field DEPARTIM the same information will be displayed as if you write the field TODAYTE.

```
CONSH2 ("CUSTREQ:CITYCODE ",CITY,"V");
```

Here the storage area of the field CITYCODE is the program variable CITY. The storage area of the field CITYCODE is no longer internal to MIMER/SH but the program variable CITY. One should note that the information in the storage area (and thus in the variable CITY) will always be stored as characters (of hollerith type).

3.3.5 DEFSH2

Routine:	DEFSH2
Purpose:	Make definitions at runtime.
Format:	DEFSH2 type , String

Arguments:

TYPE	C4	in	Decides type of definition to be made. valid values 'MESS' or 'PMT'.
STRING	Cx	in	Definition string, the definition syntax is determined by the type parameter.

Description: The routine is used to set up definitions for message areas and prompters. See reference manual definitions and routines MSGSH2 and PMTSH2. The syntax of the definition strings is as follows:

when type = 'MESS'

area , row , col , length , sep fixtext sep

where	area	is the area number for area to define.
	row	is the row on the screen for the area.
	length	is the length of the message area
	sep	separator delimiting the fixtext. This should immediately follow the comma after length.
	fixtext	is the fixtext for the message area.
	sep	same character as sep above, this separator terminates the fixtext definition.

when type = 'PMT '

name,area,sep text sep answ sep answ... sep sep

where	name	is the name used when referring to the prompter from prompt routine PMTSH2.
	area	is the message area number on which the prompter is to be displayed.
	sep	is a separator used to delimit the prompt text and answers. The separator should immediately follow the comma after area.
	text	is the prompter text.
	sep	same character as sep above, used to end the prompter text.
	answ	is a definition of a valid answer for the prompting. Many valid answers can be defined. Each answer ended by the separator character. The first defined answer determines the maximum answer length.
	sep sep	is the end of definition mark.

Example:

```
DEFSH2 ("MESS"," 1, 24, 1,40,//");
```

Here message area number 1 of length 40 is defined on row 24 and column 1. No fixtext is defined for the message area.

```
DEFSH2 ("PMT ","OK ,1,-Record ok? (Y/N) -Y-N--");
```

Here the prompter OK has been defined on message area 1. Two answers have been defined, Y and N. The separator in this case is the minus sign.

Note:

Answer definitions should always be made in capital letters. A convenient way to define prompters is to use the get string facility (GSTSH2) to fetch the definition string into the main program.

This may be done as:

```
I:=10
Loop
          GSTSH2 (I,STRING, LEN);
          DEFSH2 ("PMT", STRING);
          exit when I:=14;

end loop:
```

Now you can use the editor to define strings 10 - 14 according to the prompter definition above.

3.3.6 DIASH2

Routine:	DIASH2
Purpose:	Dialogue with user on specified position of the screen.
Format:	DIASH2 row , col , mess , mlen , answ , alen , status

Arguments:

ROW	Iw	in	Row on screen.
COL	Iw	in	Column on screen.
MESS	Cx	in	Message to write.
MLEN	Iw	in	Length of message.
ANSW	Cx	out	Answer buffer.
ALEN	Iw	in	Maximum answer length.
STATUS	4Iw	out	Read status.

Description: The message is written to the screen at the row and column specified by ROW and COL. Then prompting is made directly after the last character in the message text. If the message length is zero prompting is done at the position given by ROW and COL. The end user terminates prompting giving end of field (i.e. carriage return). If the answer length is zero the message is written to the screen without any prompting. Although no reading is done the output buffer is written to the screen. See section 2.1 on output buffering.

The length actually read will be returned in status(3) and the read answer will be left justified and padded with blanks to the length given in ALEN.

If a negative answer length is specified, the answer will be read invisible (currently restricted to machines with character read).

Dialogue on a message area is possible by setting the column (COL) to zero and giving message area number in the row (ROW) specification.

Example: DIASH2 (10,12,"TEXT>",5,ANSW_AREA,3,STATUS);

The user is prompted by the text "TEXT>" at row 10 column 12 and an answer of max length 3 is read from column 17 into the ANSW_AREA. If an exit function is given as answer this is given by the status variable. See section 3.4 on status parameter.

Note: 1) This routine provides a means of reading simple information from the screen from check routines which is useful since RDFSH2 cannot be used here.
2) The routine can be used to force the output buffer to be written to the screen.

3.3.7 ENDSH2

Routine:	ENDSH2
Purpose:	End MIMER/SH session.
Format:	ENDSH2
Arguments:	no arguments.
Description:	The output buffer is emptied, the definition file closed and terminal communication reset.

3.3.8 GETSH2

Routine:	GETSH2
Purpose:	Fetch information for a field into a program variable.
Format:	GETSH2 field , var , type

Arguments:

FIELD	Cx	in	Field specification type 1
VAR	ref	out	Program variable in which to store information
TYPE	C1	in	Type specification for output value. Valid values 'I' or 'C'.

Description: The information in the storage area associated with the specified field is moved into a program variable. Since the information in the storage area is in character format also for numeric fields, conversion to integer program variables can be made setting TYPE to 'I'.

Example:

```
GETSH2 ("CUSTREQ:CITYCODE ",CITY,'I');
```

The information of the field CITYCODE is stored in the program variable CITY in integer format. The information may have been entered using a read fields call. The conversion to integer suggests that the field CITYCODE is defined as a numeric field to avoid conversion errors.

Note:

1) When getting character information (Type='C') be sure to define your variable with sufficient size.

2) Conversion errors due to non-numeric information or too large values cause program error exit.

3.3.9 GINSH2

Routine: GINSH2

Purpose: Get status information.

Format: GINSH2 type , par1 , par2 , par3

Arguments:

type	C2	in	Type of information to be retrieved. Valid values 'CP','CF','FI', 'VE','EA','EP' and 'EF'.
when type = 'CP'			Fetch current picture info.
PAR1	Cx	out	Current picture name.
PAR2	Iw	out	Current mode number of picture.
PAR3	Iw	n/a	Dummy.
when type = 'CF'			Fetch current field name. Valid only from check routine.
PAR1	Cx	out	Current field name.
PAR2	Iw	out	Field length
PAR3	Iw	out	Read status, 0 = not given 1 = given
when type = 'FI'			Fetch field information.
PAR1	C8	in	Field name to get info for.
PAR2	Iw	out	Field length
PAR3	Iw	out	Read status, 0 = not given 1 = given
when type = 'EA'			Existence check, application.
PAR1	C8	in	Name of appl. definition file.
PAR2	Iw	n/a	Dummy.
PAR3	Iw	out	-1 = does not exist, 0 = exists
when type = 'EP'			Existence check, picture.
PAR1	C8	in	Picture specification.
PAR2	Iw	n/a	Dummy.
PAR3	Iw	out	-1 = does not exist, 0 = exists
when type = 'EF'			Existence check, field.
PAR1	Cx	in	Field specification type 1.
PAR2	Iw	out	-1 = picture not initiated, 0 = picture initiated.
PAR3	Iw	out	-1 = does not exist, 0 = exists
when type = 'VE'			Get MIMER/SH version number.
PAR1	C8	out	Version number XX.XX.XX
PAR2	Iw	n/a	Dummy.
PAR3	Iw	n/a	Dummy.

Description:

The routine is used to retrieve status information. The desired information is retrieved into the respective arguments. The current picture and field information are useful for check routines that do slightly different checks depending on which field they are called from. The field information retrieval, may be used to check if a certain field has been entered. The application file existence check, checks if the application definition file with specified name exists. The picture existence check, checks if the specified picture exists in the currently initiated application definition. The field existence check, checks if the specified field exists. The picture in which the field is searched for must have been initiated. The existence check is useful when making applications where the user may specify program actions, e.g. a program where the user starts by specifying in which columns of a table field contents will be stored.

Example:

```
integer FIELD(2);          -- program storage definition
integer LEN,STAT;
.
.
GINSH2 ("CF",FIELD,LEN,STAT);
```

The variable FIELD will not contain the character string for the current field name; LEN contains the defined length of the field and status as to whether the field has been entered or not. (This code is only valid for check routines.)

3.3.10 GSTSH2

Routine:	GSTSH2
Purpose:	Get defined string into application variable.
Format:	GSTSH2 string_no , string , length

Arguments:

STRING_NO	Iw	in	String number. Valid values 1 - 1000. This is the identification given when defining the string.
STRING	Cx	out	Variable to get string into.
LENGTH	Iw	in/out	The defined length for the string.

Description: The string defined for the string number given is retrieved into the application variable STRING. The actual length defined for the string is returned in LENGTH and the string itself is returned in STRING.

Example: GSTSH2 (10,STRING,LENGTH);

Note: Be sure to define the string receiving area large enough.

3.3.11 INASH2

Routine:	INASH2
Purpose:	Allocate space for a picture.
Format:	INASH2 pic
Arguments:	PIC C8 in Picture specification.
Description:	The routine completes the initiation of a picture. Initiation of a picture is started by a call to INPSH2. In between the calls to INPSH2 and INASH2 one may connect fields using CONSH2.
Note:	All pictures must be initiated by a call to INPSH2 followed by a call to INASH2 before they can be used.

Example:

```
INPSH2 ("CUSTREQ");
INASH2 ("CUSTREQ");
```

Initiation of a picture making no connection of fields.

```
INPSH2 ("CUSTREQ ");
CONSH2 ("CUSTREQ:TODAYTE ","CUSTREQ:DEPARTIM ","F");
INASH2 ("CUSTREQ ");
```

Initiation of a picture making a connection of one field to another field.

3.3.12 INISH2

Routine:	INISH2			
Purpose:	Initiate a MIMER/SH session.			
Format:	INISH2 termtype , applfile			
Arguments:	TERMTYPE	Iw	in/out	Terminal type
	APPLFILE	Cx	in	Application file name, ended with a space character.

Description:

The MIMER/SH terminal handling is initiated and, if the file specification is nonblank, an application definition file is initiated. The application file contains definitions used through calls to MIMER/SH routines. This is the file name given as output for the MIMER/SH compiler.

The terminal type is initiated through the following steps:

i) To determine terminal type the initiation file 'SHINIT' is used if found. The first record of this field shall contain the text TTYPE:nn, where nn is the MIMER/SH terminal no.

   See machine dependent information for name of initiation file. See terminal type appendix for terminal type information.

ii) If no initiation file is found, the operating system is asked for terminal type if it is possible. See machine dependent information.

iii) If no init file is found the argument TERMTYPE is used if a nonzero value is given.

iv) The user is prompted for terminal type.

The derived terminal type is returned in parameter TERMTYPE.

Example:

```
TTYPE:=0;
INISH2 (TTYPE,"application ");
```

Since a zero is specified as terminal type the user will be prompted if operating system cannot provide a terminal type and no initiation file is found. The terminal type derived will be returned in TTYPE.

3.3.13 INPSH2

Routine:	INPSH2			
Purpose:	Initiate a picture			
Format:	INPSH2 pic			
Arguments:	PIC	C8	in	Picture specification
Description:	This routine initiates internal picture tables to handle writing of pictures and fields and reading of fields. The initiation must be completed with a call to INASH2.			

3.3.14 JMPSH2

Routine:	JMPSH2			
Purpose:	To support re-direction of reading from one field to another field.			
Format:	JMPSH2 field , offset			
Arguments:	FIELD	Cx	in	Field specification type 1
	OFFSET	Iw	out	Offset in field read sequence number

Description: This routine is only useful from check, unit check or exit routines when reading with RDFSH2. It returns the offset from current field (the field for which the check, unit or exit routine was called) to the field specified by FIELD. Both fields must be defined in the same picture. When the returned offset is put in STATUS(2) on exit from the check routine, the next field read will be the field specified by FIELD.

Example:

```
procedure check (fldbuf,len,status)
.
begin
.
.
JMPSH2 ("READTHIS ",STATUS(2))
STATUS(1):=1;
end check;
```

Here the field READTHIS will be the next field to read when current field (field for which check routine check has been defined) has been read.

Note: Jumps may pass required but not given fields. If a jump is made to a field that is currently excluded from the picture the jump will be made to the first following (in read order) non excluded field.

3.3.15 MSGSH2

Routine:	MSGSH2			
Purpose:	To display a message on a message area.			
Format:	MSGSH2 area , text , length			
Arguments:	AREA	Iw	in	Message area number for the message area in which to display the text.
	TEXT	Cx	in	Message text to be displayed.
	LENGTH	Iw	in	Length of message.
Description:	The message is displayed on the message area. See routine DEFSH2 and sections 3.3.6 of reference manual.			
Example:	MSGSH2 (1,"Please reenter citycode",23);			
Note	This routine also writes the output buffer to the screen.			

3.3.16 PMTSH2

Routine:	PMTSH2
Purpose:	To prompt predefined answer from the terminal, using a predefined prompter text.
Format:	PMTSH2 name , no

Arguments:

NAME	C4	in	Name of prompter.
NO	Iw	out	Number for given answer

Description: The user is prompted with the predefined prompter text. The prompting is done on the position of the message area, on which the prompter has been defined. Only one of the answers defined for the prompter can be entered by the end user. The number returned for a correct answer, is the order number for the answer, among the defined answers. The matching is done in upper case regardless of the case the answer was entered in. See routine DEFSH2 and sections 3.3.6 of reference manual.

Example:

```
PMTSH2 ("OK ",NUMBER);
```

If two answers have been defined, e.g. 'Y' and 'N' and the user answers 'N', NUMBER will return two.

3.3.17 PROSH2

Routine:	PROSH2
Purpose:	Project field to a column of a MIMER/DB table or project a whole picture to a table.
Format:	PROSH2 tid , lop , col , acc , ba , field , len

Arguments:

TID	4Iw	in/out	Table identifier
LOP	C1	in	Logical operator
COL	C8	in	Column name
ACC	C2	in	Access specification
BA	ref	in	Base address
FIELD	Cx	in	Field specification type 1
LEN	Iw/C1	in	Length

Description: A data transmission path is set up between the storage area of the field specified by FIELD, and the column COL in the table given by TID. If '*' is specified as fieldname in the field specification, projections are made for successive fields of the picture, until no more fields in the picture or no more columns in the table remain. If '*' is specified as projection length the field length is used.

See MIMER/DB reference manual description of PROJE2 routine.

Example:

```
PROSH2 (TID,"F","CITYNAME","RO",BA,"CITYNAME ","*");
```

Here the column CITYNAME of the table CITIES is projected to the storage area of the field CITYNAME.

Note: The table (TID) must have been initiated through calls to MIMER/DB.

3.3.18 PUTSH2

Routine:	PUTSH2
Purpose:	Store information into the storage area of a field.
Format:	PUTSH2 field , var , type

Arguments:

FIELD	Cx	in	Field specification type 1
VAR	Cx/Iw	in	Program variable/constant to store.
TYPE	C1	in	Type of variable/constant. Valid values 'C' and 'I'.

Description: The information in the program variable VAR is stored in the storage area associated with the specified field. Since the information of the storage area is in character format also for numeric fields, conversion from an integer program variable should be made setting TYPE to 'I'.

Example:

```
PUTSH2 ("CUSTREQ:CITYCODE ",CITY,'I');
```

The program variable CITY is converted from integer format and stored in the storage area of the field CITYCODE.

Note: Conversion errors cause program error exit.

3.3.19 RDFSH2

Routine:	RDFSH2
Purpose:	Read fields of a picture.
Format:	RDFSH2 pic , fields , status

Arguments:

PIC	C8	in	Picture specification
FIELDS	Cx	in	Field specification type 2
STATUS	4Iw	in/out	Read status

Description: The routine reads specified fields of a picture. See Chapter 4 for information about read control, field information checking and repositioning the cursor.

Example:

```
RDFSH2 ("CUSTREQ ","* ",STATUS);
```

All fields of the picture CUSTREQ are read.

Note: The routine cannot be called recursively i.e. you cannot call the routine from a check, unit, exit or edit routine (which is called by RDFSH2).

3.3.20 RMPSH2

Routine:	RMPSH2
Purpose:	To remove picture initiated with INPSH2 and INASH2
Format:	RMPSH2 pic
Arguments:	PIC C8 in Picture specification
Description:	The picture is deactivated. Thus the storage area is lost and the picture can no longer be referenced in any call to MIMER/SH. The routine is used to free internal areas.
Example:	RMPSH2 ("CUSTREQ ");
Note:	If you remove a picture that has a field of another picture connected to one of its fields, you may not use the field of the other picture. If you do the result will be unpredictable.

3.3.21 SELSH2

Routine: SELSH2

Purpose: Make selections on a MIMER/DB table against a field.

Format: SELSH2 tid , lop , col , rop , ba , field , len

Arguments:

TID	4Iw	in/out	Table identifier
LOP	C1	in	Logical operator
COL	C8	in	Column specification
ROP	C1	in	Relational operator
FIELD	Cx	in	Field specification
LEN	Iw/C1	in	Length of selection

Description: The information of the field storage area associated with the field FIELD is set up as a selection on column COL in the table defined by TID. If the length of selection is set to '*' the field length will be used for selection.

See MIMER/DB reference manual description of SELEC2 routine.

Example:

```
SELSH2 (TID,"F","CITYCODE","EQ",BA,"CITYCODE ","*");
.
.
SET2 (TID,BA);
GET2 (TID,BA);
```

The row retrieved from the table given by TID will be selected to have the value of column CITYCODE equal to the value stored in the field storage area of the field CITYCODE at the time the SET2 call is executed.

Note: The table must have been initiated through MIMER/DB calls. A special field specification, $INFO denoting the field information buffer on input to check routines, may be used here. This can be useful since information read from the screen, is not yet stored into the field when the check routine is entered. See Chapter 4 on check routines.

3.3.22 SETSH2

Routine:	SETSH2
Purpose:	To set MIMER/SH parameters.
Format:	SETSH2 type , par1 , par2 , par3

Arguments:

TYPE	C2	in	Parameter type to set Valid values 'PO', 'PE', 'PN', 'CP', and 'MO'
when type = 'PO':			
PAR1	C1	in	dummy parameter
PAR2	Iw	in	row offset for pictures
PAR3	Iw	in	column offset for pictures
when type = 'PE'			
PAR1	Cx	in	Exit specification, valid values: 'NOARR', 'ARRCHK' or 'ARR'
PAR2	Iw	n/a	dummy
PAR3	Iw	n/a	dummy
when type = 'PN'			
PAR1	Cx	n/a	dummy
PAR2	Iw	n/a	dummy
PAR3	Iw	n/a	dummy
when type = 'CP'			
PAR1	Cx	in	picture specification
PAR2	Iw	n/a	dummy
PAR3	Iw	n/a	dummy
when type = 'MO':			
PAR1	C8	in	picture specification
PAR2	Iw	in	mode number
PAR3	Iw	in	dummy parameter
when type = 'FX',x = G,N,E,W,O,R or F:			
PAR1	Cx	in	field specification type 2
PAR2	Iw	n/a	dummy parameter
PAR3	Iw	n/a	dummy parameter

Description: When TYPE equals 'PO' the picture offset is set to the row and column given in PAR2 and PAR3, respectively. This changes the location on the screen for all pictures and their fields in all following treatment of pictures. Default value 0.0.

When TYPE equals 'PE' the picture exit using arrows may be set,
'NOARR' = arrows cannot exit from picture (default)
'ARRCHK' = arrows will exit from picture after information check
'ARR ' = arrows will exit from picture
This setting will hold until a new setting of arrow exit is done.

When TYPE equals 'PN' a temporary no-initiation of the picture is set up, I.e. when entering a picture read call (RDFSH2) no status initiations will be done. This makes it possible to reenter reading the just exited picture with the same status as before no-initiation exit.
The no-initiation will only hold for the next call to RDFSH2.

When TYPE equals 'CP' the current picture is set, i.e. field specifications without explicit picture name will assume that the current picture is referred to.

When TYPE equals 'MO' the picture mode number for picture specified in PAR1 is changed to the mode number given in PAR2. Default value 0.
See section 4.4 on read control.

When TYPE equals 'FG' the specified fields are marked as given. I.e. as if the user had entered information into the fields. See section 4.6.

When TYPE equals 'FN' the specified fields are marked as not given. I.e. as if the user had not entered information into the fields.

When TYPE equals 'FR', 'FO', 'FE', or 'FW' the read control codes of the fields specified are set to R-,O-,E-, or W- respectively. (see chap. 4.4 read control codes). These settings will only hold during current picture read (thus only useful from check or exit routines) or until a new picture mode is chosen (see 'MO' above).

The field required setting only specifies that the field must be entered during a picture read. Once it has been entered it will be set to given So if you want to specify that a certain field must be entered anew, you must set that the field is not given ('FN') and that it is required ('FR'). This setting has a mnemonic of it's own 'FF' = forced field read.

Example: SETSH2 ("PO","DUMMY",4,0);

All following calls will now work on pictures shifted 4 rows down on whe screen.

SETSH2 ("MO","CUSTREQ ",1,DUMMY);

Now all the following treatment of the picture CUSTREQ will use mode number 1.

3.3.23 WRFSH2

Routine:	WRFSH2
Purpose:	Write specified fields to the screen
Format:	WRFSH2 pic , fields

Arguments:

PIC	C8	in	Picture specification
FIELDS	Cx	in	Field specification type 2

Description: The fields specified in FIELDS are written to the screen.

Example:

```
WRFSH2 ("CUSTREQ ","CITYCODE-CITYNAME ");
```

The fields in the read sequence between CITYCODE and CITYNAME are written to the screen.

3.3.24 WRPSH2

Routine:	WRPSH2			
Purpose:	Write the texts of a picture			
Format:	WRPSH2 pic			
Arguments:	PIC	C8	in	Picture specification
Description:	The texts of the pictureare written to the screen.			
Example:	WRPSH2 ("CUSTREQ ");			
	The texts of the picture CUSTREQ are written to the screen.			

3.4 Description of the STATUS parameter:

The status parameter is used with several routines. It consists of an integer array of four elements.

This array can be extended and used by the application programmer to pass parameters from main program to check, unit check, edit or exit routines when the programming language does not support global (common) variables.

3.4.1 STATUS parameter from routine DIASH2

STATUS(1): Exit status

<0 =	Error
0 =	The answer was read ok.
1 =	An arrow was given in the first prompting position
2 =	An exit function was given in the first prompt. position

STATUS(2): Exit information

1. when arrows the number for the arrow given (see RDFSH2) or 7 = Question mark given in first position.
2. when exit function, the exit code for the function.

STATUS(3) Actually read answer length

3.4.2 STATUS parameter from routine RDFSH2

On exit from RDFSH2, STATUS parameters have the following meanings:

STATUS(1): Exit status

<0 = Error (system or read error)
0 = Last field of picture read
1 = Arrow out of picture
2 = Exit function used to exit from picture

STATUS(2): Exit information

1. when Arrow
   1 = Backward arrow out of picture
   2 = Forward arrow out of picture
   3 = Up arrow out of picture
   4 = Down arrow out of picture

2. when Exit function

   Number of exit function

STATUS(3): Information status

0 = Information ok
1 = All required fields not given
2 = All required fields given but some unit check is not yet performed

3.4.3 STATUS parameter in user written check routines

On exit from user written check routines, STATUS parameters have the following meanings:

STATUS(1): Exit directive

-1 = Field not ok reread field
0 = Read next field in read order
1 = Reposition reading to field specified by status(2)
2 = exit from reading picture

STATUS(2): Exit specification

1. when repositioning (status(1)=1)
   Offset in read order to the next field to read. Then offset is given from next field to read. i.e. 1 is skip next field and reposition reading to the following field. See routine JMPSH2.

2. when exiting from picture (status(1)=2)

   Exit code that will be given in status(2) when returning from the call to RDFSH2.

STATUS(3): Information treatment

-1 = Discard information in info parameter. Otherwise the information is stored into the storage area for the field.

3.4.4 Using the STATUS parameter for application variables

STATUS(4+I): I > 0 Can be used to communicate information from main program to the check routine. This is useful when MIMER/SH is used together with a programming language that does not support global (common) variables.

3.4.5 STATUS parameter in unit check routines

The status variable is treated as with Check routines.

STATUS(1): Exit directive

input 0 = the unit check was called when repositioning in the picture.
2 = the unit check routine is called since an exit function has been given.

output

not altered =	the unit check is ok, proceed as if the routine had not been called and deactivate the unit.
-1 =	the unit check was not ok, let the unit stay active and reposition to field given by status(2)
1 =	the unit check was ok, deactivate unit and reposition to field given by status(2)
2 =	exit from reading picture

STATUS(2): Exit specification

1. when repositioning (Status(1) = -1 or 1) offset in read order to next field to read (from last field in unit).
e.g. status(2) = -1 will position to the last field of the unit.

2. when exiting from the picture (status(1) = 2).
Exit code that will be given in status(2) when returning from the call to RDFSH2.

3.4.6 STATUS parameter in exit routines

STATUS(1): Exit directive (out from exit routine)

- -1 = reread field
- 0 = exit with exit code specified for exit function that entered the exit routine
- 1 = reposition reading
- 2 = exit with exit code specified in status(2)

STATUS(2): Exit information (out from exit routine)

1. when (status(1)= -1 or 0
   N/A

2. when repositioning (status(1)=1)

   Offset, in read order, of field to reposition to.

3. when exit with specified exit code (status(1)=2)

   Exit code to be used at exit.

STATUS(3): Information checking (input/output)

On input, STATUS(3) contains the same picture check information that will be output from RDFSH2 i.e.

- 0 = picture information ok.
- 1 = at least one required field has not been given
- 2 = a unit is still active.

3.5 Logging facility

A terminal input logging facility is included as a development tool. This may be useful when repeatedly testing an application under development.

The entrance to this facility is the routine LOGSH1, that the user can call e.g. from an exit routine. An example of an exit routine using the logging function is given here.
The logging will when started for write, write all input to MIMER/SH on a sequential file. After the write log has been turned off, the log may be opened for read, which means that the screen handler will take its input from the log and thus perform exactly the previous dialogue.

Example in FORTRAN

```
      SUBROUTINE LOGFNC (EXCODE, STATUS)
C-------------------------------------------------------------
C     Exit function routine to activate logging functions
C     through the MIMER/SH routine LOGSH1
C
C
C     input parameters to LOGSH1:
C
C     code:  Iw   1 = Start writing input to WLGFIL
C                 2 = Hold     "       "    "    "
C                 3 = End writing and close file
C                 4 = Start reading input from RLGFIL
C                 5 = Hold     "       "    "    "
C                 6 = End reading and close file
C
C
C     rdfile Cx   Read  log file name.
C     wrfile Cx   Write log name
C
C     fslen   Iw  Length of sequence to discard backwards
C                 In log. (used to trigger this routine)
C
C
C
C     This routine is just a suggestion and may be customized to
C     your liking.
C     E.g. you may prompt for name of log file.
C--------------------------------------------------------------
 INTEGER EXCODE,STATUS(4)
 INTEGER COND,CODE
 INTEGER CHR
 INTEGER RDFILE(17),WRFILE(17)

 C
 C--   Initiation
 C
       CALL MDRMVF (RDFILE,0,66,1H)
       CALL MDRMVB (RDFILE,0,12HSHLOG.LOG    , 0,12)
       CALL MDRMVB (WRFILE,0,RDFILE, 0,12)
```

```
C
C--     Prompt user

        CALL DIASH2 (1,0,37HLog Start wri/rea,End wri (SW,SR,EW):,38
*           ,ANSW,2,STATUS)
        DO 10 I=1,3
        CALL MDRCLB (ANSW,0,6HSWSREW,(I-1)*2,2,COND)
        IF (COND.EQ.0) GOTO 20

10      CONTINUE
        CALL MSGSH2 (1,10HNo action    ,10)
        GOTO 100
C
C--     Determine function and call log routine
C
20      IF (I.EQ.1) CODE=1
        IF (I.EQ.2) CODE=4
        IF (I.EQ.3) CODE=3
C
        CALL LOGSH1 (CODE,WRFILE,RDFILE,4,COND)
        IF (COND.NE.0) GOTO 29
        CALL MSGSH2 (1,10HOk        ,10)
        GOTO 100
C
29      CALL MSGSH2 (1,10HError     ,10)
C
C--     Reposition to field from which exit routine was called
C
100     STATUS(1)=1
        STATUS(2)=-1
        RETURN
        END
```

4. STRUCTURE DESCRIPTION

4.1 Information Checking

When a field is read from the screen there are mainly three ways to check that the desired information is entered correctly, namely:

-- check codes
-- validations
-- user written check routines

4.1.1 Check codes

A check code for a field is specified during definition of the field. The check code is a mnemonic that specifies the type of characters that may be entered in the field. One of the following check codes may be specified:

N_	Numeric
D_	Decimal
A_	Alphabetic
AN	Alpha numeric
AP	All printable
DT	Date
TX	Text

A character entered that does not satisfy the check code will result in a bell signal. (*)

Together with the decimal check code the desired number of decimals is specified. As decimal delimiter dot, semicolon or comma may be used but it is always translated into a dot. The decimal dot is included in the field storage length. For instance, if a field has a storage length of five positions and is specified to have two decimals, two digits can be entered before a decimal point is given.

The date check is a combination between a numeric code and a check routine checking for correct date in the notation yymmdd.

The text code gives the same check on entered characters as the AP code but gives the possibility to edit the information of the field. See field editing.

(*) **Note:** The time at which the character is rejected depends on the type of reading MIMER/SH works with. When character by character reading the reject is done immediately as the illegal character is entered. When field reading the reject is done when the field is sent to MIMER/SH (i.e. when end of field (CR) is given). In the last case an error message will also be displayed.

Additional check code

An additional check code may be specified:

along with N_ and D_ check codes a sign code may be specified.

S - sign required
— sign optional
U - unsigned

The sign character is included in the field length.

Along with A, AN, AP and TX check codes a conversion code may be specified.

UP upper case conversion
— no conversion
LO lower case conversion

The minimum number of characters to be entered may also be specified. The field will not be accepted if fewer characters are entered.

4.1.2 Validation

Validation is also specified when the field is defined. The validation consists of values or ranges of values that the information must satisfy.

Before the validation is applied the information is justified and padded according to the definitions for the field.

If the information entered does not satisfy the validation requirements an error message is displayed and the field is reread.

4.1.3 Check routines

Check routines are routines written by the application programmer to perform desired checks on entered information.

The name of the check routine is specified when the field is defined. This makes MIMER/SH call the routine when the field has been entered.

The parameters of the check routine are used as follows:

procedure CHECKR (INFO, LEN, STATUS);

parameters

INFO	in/out	The information entered into the field. Here the information has already been checked in respect of check code and validation. See Chapter 1 on Information Handling.
LEN	in	The number of characters that has been entered into the field.
STATUS	in/out	See section 3.4 on the STATUS parameters.

4.1.4 Unit check routines

Unit check routines are check routines that are specified for more than one field and called by a read dependent triggering mechanism. A unit consists of a unit id that is given along with:

a. The field definition for the fields that shall belong to the unit.

b. The unit check routine.

The unit check routine is called when passing in the forwards direction the last field (in read order) of the unit and at least one field of the unit has been entered.

The unit check routine will also be called when an exit function is given and a unit is active.

Summary: Units consists of a set of fields.
The unit is set active when one or more fields of the unit is entered.
The unit check routine is called if the unit is active when passing the last field of the unit in 'forward' direction or when an exit function is given. Here the unit routine is called before the exit routine.
The unit is de-activated when the unit routine has been called.

EXAMPLE **

```
*=================================================*
!                                                 !
!      field1:  xx     field2:  xx     field3:  x !
!      field4:  xx     field5:  xx                !
!                                                 !
*=================================================*
```

We assume that the fields 1 and 3 belong to a unit. If we enter anything into field 1 (or 3) the unit will be activated.
If we now reposition in between fields 1 - 3 the unit check routine will not be called, but trying to reposition to field 4, the last field of the unit (3) will be passed and the unit check routine called. Trying to exit from picture using an exit function will also call the unit check routine.

*** EXAMPLE END

See section 3.10 for a description of the parameters of Unit Check routines. The parameters on input are used slightly differently.

INFO	in	Contains contents of the last field in the unit.
LEN	in	Length of the last field in the unit.
STATUS	out	see section 3.4 on the status parameters.

4.2 Field display

A field is written to the screen, either through an explicit write call or when a field has been successfully entered. When the field is written, it may be display edited, i.e. only edited for writing to the screen, not in the storage of the field. This editing can be done in two ways:

1. Specifying an edit pattern for the field.

2. Specifying an edit routine for the field.

Only one of these alternatives may be used at a time.

4.2.1 Display editing pattern

With this pattern you can edit a digit string, i.e. the contents of a numeric field, and specify whether leading zeroes should be substituted, if you want to insert characteres in the field (e.g. a comma or a semicolon) and what characters you want to be displayed as signs for the field.

The first character of the pattern determines the character to use when leading zeroes are to be substituted.

The following characters are:

- o either pattern zeroes or nines that will be substituted for digits from the field

- o or any character that is to be inserted into the edited field.

Pattern nines will always be substituted with a digit from a field. Pattern zeroes will only be substituted with a digit from a field if the digit is not a zero, otherwise they are substituted with the substitute character mentioned above. The pattern zeroes are only significant in the beginning of the pattern, i.e. if a pattern zero is given in a position to the right of a pattern nine the pattern will be erroneous.

All together the number of given zeroes and nines should correspond to the length of the field.

In the last three positions of the pattern, the characters to be used for sign handling are given. These characters are given in the order: negative character, zero character and positive character. The sign character is put into the edited field in front of the first pattern nine or the first nonzero digit.

```
EXAMPLE: *******************************************************

pattern:    _009-Z+
This pattern is used with a field of 3 positions length and display
length 3.
If the field contents are     : 002 the field will be displayed as      _+2
                              : 022                                     +22
                              : 000                                     _Z0

pattern:    _099-Z+
If the field contents are     : 002 the field will be displayed as      +02
                              : 022                                     +22
                              : 000                                     Z00

pattern:    _09.9___
Field length 3 and display length 4.
If the field contents are     : 002 the field will be displayed as     _0.2
                              : 022                                    _2.2
                              : 000                                    _0.0
                              : 222                                    22.2
```

A special feature of the pattern is that you can specify characters that will be written after the last digits in the edited field, only if the field is negative. This may be useful in a book keeping application.

```
pattern:    _09.9 neg___
Field length 3 and display length 8.
If the field contents are     : 022 the field will be displayed as     _0.2
                              : -02                                    _0.2 neg

******************************************* - EXAMPLE END
```

4.2.2 Display editing routine

This routine will be called before writing the field for which it has been specified. In this routine you edit the field contents to the shape you want it.

The parameters of the display editing routine are as follows:

procedure DISPED (INFO,LEN,EDINFO,DUM);

parameters

INFO	in	The contents of the field for which the routine has been defined.
LEN	in	Field length.
EDINFO	out	Area to put edited string into.
DUM	n/a	reserved for future use.

4.3 Exit functions

Exit functions are used to exit from reading the fields of a picture. When an exit function is activated, reading of the picture is interrupted and control returned either back to the application program (from the call to RDFSH2) or to an exit routine. An exit function is activated by the end user typing an exit character on the keyboard. Before exiting picture reading, a picture information check is done; this checks whether all required fields have been entered and whether unit checks have been made. If the picture exit is restricted, then:

1) a message prompts for those fields which are required but not given and the cursor is repositioned to read the first field which was not given.

2) triggered unit checks will call the unit check routine.

If the picture exit is not restricted, the picture reading will be terminated with a status code signalling information status; see section 3.4. The exit function definition consists of a definition of an exit character, which defines which picture information checks shall restrict exit and an eventual exit routine name.

4.3.1 Exit routines

For each exit function an exit function name can be defined. This routine is then called when the exit function is activated. From this routine you can either return to reading the picture or exit from reading, setting the status parameter accordingly.

The parameters of the exit routine are used as follows:

procedure EXITR (CODE,STATUS);

parameters

CODE	Iw	in	The number of the exit function that entered the routine. This can be useful if the same routine is defined for several exit functions.
STATUS	4Iw	out	Used to control continued reading when returning from exit routine; see section 3.4.

4.4 Read control and picture modes

Picture modes give the possibility to read the picture in several different ways. Each mode is designated by a mode number. The fields of a picture can take several mode definitions. A mode definition consists of a mode number and read control code.

Possible read control codes specify if the field is to be:

R_	required i.e. must be entered,
O_	optional i.e. may be entered,
RB	required may be blanked i.e. must be entered but may be blanked
OB	optional may be blanked i.e. may be entered or blanked
W_	write only i.e. displayed but not possible to enter or
E_	excluded.

Exclude is default for a field when reading with a mode number that has not been given a mode definition.

A required field cannot, but an optional field can be passed in forward direction if the field has not been given (a legal value entered). See section 4.5 on repositioning.

If 'may blank' has been defined for the field, function End of field (carriage return) in the first position of the field, is a legal value. The field will then be filled with the fill character defined for the field and further the field will not be checked in respect of validation or correct date.

E.g. if a field is defined with control code OB 'optional may blank' and a minimum fill of 2 characters, the user can:

i) pass the field without giving a value (optional)
ii) blank out the field giving carriage return in the first position of the field (may blank)
iii) enter at least two characters into the field.

The mode by which a picture is read is set by a call to the routine SETSH2. The mode holds for the picture until the next setting for the picture. The startup mode number (default if no setting is done) is zero.

EXAMPLE: **

The first field of a picture has a check routine defined that retrieves a record from the data base, checks a column that determines how this record may be treated, and sets the mode number to exclude some fields from reading.

```
      mode 0                                  mode 1
*=====================*   set             *=====================*
|RRRR    EEEE  EE     |   mode to 1 = >   | EEEE    RRRR OO     |
|EEEE EEE             |                   | RRRR                |
*=====================*                   *=====================*
      Set mode to 2

      mode 2
*=====================*
|EEEE   EEEE EE       |
|OOOO RRR             |
*=====================*
```

Here we see the same picture under different mode numbers. In the first case (mode0) the only field used in the picture is the one in the upper left corner. This field cannot be passed since it is defined as required. Setting mode no to 1 this field is excluded and four other fields constitute the picture.

Note: Specification of mode masks for fields must presently be done using an ordinary test editor to add picture mode records (see Appendix C, page 05.

** - EXAMPLE END

4.5 Repositioning the cursor among the fields of the picture (only for asychronous terminals)

Repositioning here means moving the cursor to the start position of another field. For repositioning the fields, the functions field forward, backward, up and down are used. The functions are usually implemented as the terminal arrow keys for corresponding direction, i.e. function field forward is implemented as forward arrow and so on. On fields that are not text fields these functions are only valid in the first position of a field.

Function field forward will reposition reading to the next field in read order. Function field backward will reposition reading to the previous field in read order. Function field down will reposition to the first field of the first line that contains a field and is below current line on the screen. Function field up will reposition to the first field on the first line above current that contains a field.

Function forward and down are not allowed in a field that has been defined as required but has not been filled in. Here a bell signal is given.

Neither is it possible to pass a required field further out on the same line defined as required but not filled in. Here the cursor will be repositioned to the required field. When end of field function is given in the field and the field is satisfactorily filled in, the next field in read order is used.

4.6 Field enter status

Whether fields have been given (entered) or not is kept as status for all fields of the current picture. The status is used to determine whether the user may reposition past a required field or not. This status can be tested or set for each field, from check, unitcheck, exit routines or after the call to read fields (RDFSH2). Thus you can use the get status information routine (GINSH2) to decide if a field has been given. This status is reset when a new call to read fields is done.

4.7 Editing when entering fields (asynchronous terminals only)

Ordinary fields (non text fields) have only one editing function and that is the delete function. This is usually implemented as the delete key. Delete will delete the last character typed in. To end the entering of a field, the function end of field (usually implemented as carriage return) is used.

Text fields have field right and field left functions to move the cursor in the field. These are usually implemented as the terminal arrow key right and left. Characters typed in a text field will be inserted (i.e. rest of field is shifted one position to the right) at current position and delete function will delete the character before the cursor (and the rest of the field is shifted one position to the left). The end of field function will leave the field and proceed to the next field.

4.8. Message areas

Message areas are parts of rows or whole rows, on the screen used for display of messages or prompters. Several message areas may be defined, each referred to by a message area number.
Message areas are handled by MIMER/SH in the following way:
If anything has ben written on the message area, the area is cleared immediately after information is entered into a field read by RDFSH2.

Here, cleared means writing of message area fixtext if such has been defined for the message area otherwise the message area is blanked.
A message area fixtext is a text that is to be displayed on the message area whenever it is not used for other purposes. The message area fixtext for a message area is defined along with the message area definition.
Default message area set up by MIMER/SH is message area one on line 24, column 1 and with a length of forty characters. (see ref man routines routine DEFSH2, page 3.07.)

Messages

There are three types of messages used in MIMER/SH namely: error, system and user messages.

Error messages are messages displayed to inform the application programmer of the reason for program failure.
See ref. manual routines - errors, Appendix G.

System messages are predefined messages used to inform the end user of reason for rejecting entered information. These messages can be chosen to be in different languages and are all displayed on message area one.

User messages are messages displayed by the application programmer to inform the end user of application program behaviour.

4.9. Prompters

A prompter consists of a prompt text to display to the end user and a defined set of values that may be given as answers. Prompters are referenced by their prompter name and they return to the application program the order number, in the answer definition, of the correct answer.

4.10 Helptexts

For each field in a picture a number of helptexts can be defined. These are displayed when the helptext function is evoked. This is usually implemented with the entering of a question mark ("?") in the first position of the field.

Helptexts can be chosen to be displayed either in desired positions on the screen or on a message area. When defining several helptexts, where some would be overwritten if displayed at the same time, the screen handler pauses and waits for input.

Now, if function end of field (carriage return) is given, no more helptexts will be displayed, else if any other key is pressed, previously written helptexts will be cleared and more helptext will be displayed.

If texts are defined directly to screen positions, the last helptexts written will be followed by a prompting and then cleared before the cursor returns to read the field.

Otherwise, if texts are defined to a message area, the message remains on the message area when the cursor returns to read the field.
See message areas, section 4.8.

APPENDICES

A GLOSSARY

B TERMINAL TYPE TABLE

C SEQUENTIAL DEFINITION FILES AND PICTURE DEFINITION LANGUAGE

D LIMITATIONS

E COMPILER ERROR CODES

F EDITOR ERROR CODES

G MIMER/SH ROUTINE ERRORS

H RESTART OF INTERNAL SPACE AREA

I TERMINAL INDEPENDENCE AND TERMINAL READ MODES

APPENDIX A GLOSSARY

Check code	Code specifying what information may be entered into a field.
Check routine	A user written routine defined along with the field definition. The routine automatically is called after the field has been entered.
Current picture	The picture name that will be used when referring to fields using a field specification (type 2) without picture name.
Connection	Two fields or more may share the same storage area.
Exit code	Number denoting a exit function.
Exit function	Specification of characters, character sequences or function keys that will exit from reading.
Exit routine	A user written routine defined along with an exit function. The routine is called when an exit function is given.
Offset	Relative distance measure with start (current) position referenced by zero.
Optional field	Field that can be entered if desired.
Picture mode	Mask over the fields of a picture. For each mode each field can be assigned a read control code deciding if the field is required-, optional- read, excluded or write only.
Picture offset	Base position on the screen for writing/reading texts and fields of a picture.
Read order	The order in which the fields are read on the screen.
Repositioning	Deciding what field will be read next.

APPENDIX A GLOSSARY

Required field
Field that must be entered.

Status
Status variable used to check read status when a picture has been read. Also used to control reading on output from check and special character routines.

Storage area
This is where the information is stored when reading the fields of a picture. It is also where the information is taken from when writing the fields of a picture.

Validation
A set of values to which field contents shall belong.

APPENDIX B TERMINAL TYPE TABLE

01	Comex 3380/Infoton 200.
02	VT52 / Teleray 10.
03	VT100 / Tandberg 2220 / Teleray 100.
04	Tandberg 2115.
05	Tandberg 2215.
06	VC 404.
07	VC414.
08	HP2640,2622.

This set of terminals may change between installations.

APPENDIX C

PICTURE DEFINITION LANGUAGE

This Appendix is a description of the file containing the punchcard oriented 'language' in which MIMER/SH information is to be defined.

Definitions are made record by record.

Position 1 on each record must contain a capital letter. This defines the type of information to be declared.

I =	Include file name.
C =	Exit character definitions.
G =	General filler for the screen.
L =	Language for MIMER/SH messages.
S =	Text strings.
P =	Picture name.
T =	Texts in a picture.
F =	Field description.
V =	Validations.
H =	Helptext definitions.
N =	Unit check routine.

D, *, ; and blank defines comment records.
X = Forces a change of block when information is stored in the MIMER/SH object file. This is used to improve performance.

C.1 Record Sequences in the Picture Definition File

1.	L ; Language definition	(1 rec)
	C ; Exit characters	(1 to 16 rec)
	G ; General filler	(1 rec)
2.	S ; Text strings	(1 to 999 rec)
3.	P ; Picture name	
3a.	T ; Texts for this picture	
3b.	F ; Fields for this picture	
3ba	V ; Validation for this field	
3bb.	H ; Help texts for this field	
3c.	N; Unit check routine for fields of this picture.	

Sequences under each group can be repeated any number of times. By a group is meant, e.g. 3., which consists of the sequence 3-3a-3b-3c. If a record is defined on a sub-level, there must be a record on the level above. E.g. one record in group 3bb. has been defined, there must be a record in 3b and 3.

Records on the same level can be left out (if there is no sub-level record), except for the N record that is defined together with declarations of fields (F records).

C.1.1 Include file name

Contents of file with given file name replace the record.

Pos	Description
1	Record type = I
2 - n	File name

C.1.2 Language

Language for MIMER/SH's own messages.

Pos	Description
1	Record type = L
2 - 4	Language abbreviation (SWE, ENG, GER, FRE)

C.1.3 Exit Character

Exit characters are used when reading with RDFSH2 or DIASH2. There is one record for each exit character. The return code for the read character will depend on the order in which the exit characters are defined. The return codes will be the order no. of the definition record +4. E.g. the first exit character defined gives return code = 5. A maximum of 16 exit characters can be defined.

Pos	Description
1	Record type = C (K)
n + 1	1 = < n= < 4 reserved
6	, which is a compulsory delimiter.
7	This position decides if unit checks shall be made : 0 = no check 1 = check if fields are changed
8	, which is a compulsory delimiter.
9 - 11	Decimal ASCII/EBCDIC value for the character. All decimal values 1 - 127/255 may be used, (ASCII/EBCDIC). Not zero.
12 - 17	The name of the exit character routine IF one is to be defined.

C.1.4 General filler

Pos	Description
1	Record type = G
2	ASCII/EBCDIC-character to be written on the screen when CLFSH2 or DIASH2 is called. Default character = blank

C.1.5 Text strings

A maximum of 999 text strings can be stored in the MIMER/SH object file. The strings that can be arbitrarily used in the application are fetched by issuing a call to GSTSH2 along with a string no.

Pos	Description	
1	Record type = S	
2 - 4	No. of the string	001 - 999
5 - 6	Length of the string	01 - 79
7 -	Text	

C.1.6 Picture name

Record defining a picture name for the text and field definitions to be stored. The name is referenced when initiating the picture with INPSH2.

Pos	Description
1	Record type = P
2 - 9	Picture name
10 -	Reserve

C.1.7 Picture texts

Picture texts are written on the screen by WRPSH2.

Pos	Description
1	Record type = T
2 - 3	Row no. where the text is to be written on the screen 01 - 24
4 - 5	Column no. where the text is to be written on the screen 01 - 79
6 - 7	Length of the text 01 - 79
8 -	Text

C.1.8 Field definition

Definition of the characteristics of the field.

The field definition consists of two or three records.

Record 1

Pos	Description
1	Record type = F.
2 - 3	Row no. of the field position on the screen 01 - 24.
4 - 5	Column no. of the field position on the screen 01 - 79.
6 - 7	Storage length of the field.
8 - 9	Display length of the field.
10 - 11	No. of decimals for the field. Only when check code = D_.
12 - 13	Read control code mode 0. Valid: R_ , O_ , W_ or E_.
14 - 15	Check code for the field. Valid: N_ , D_ , A_ , AN , AP , TX or DT.
16 - 17	Additional check code. Valid: S_ , __ , U_ , LO or UP.
18 - 19	Minimum number of characters.
20	Justification of the field. Valid: R or L.
21	Filler for the field. Used when field is justified or cleared with CLFSH2.
22 - 23	Unit definition. Specification of the unit to which the field belongs. Position 22 = U, position 23 = unit identification; valid values: 1-9 and A-V.
24 - 29	Check routine name. If the field is to be checked by a user written check routine, the name of the routine is given here.
30 - 35	Edit routine name. If the field is to be edited by a user written edit routine, the name of the routine is given here.
36	Edit pattern. If the field is to be edited using an edit pattern, an E is given in this position. Pos 37 - defines the pattern.
37 - 38	Length of the pattern. Valid: 1-30.
39	Edit pattern.

Field definition record 2.

Pos	Description
1	Record type = F.
2	Record subtype = X, field mode record.
3 - 9	Field name.

Field definition record 3.

Pos	Description
1	Record type = F.
2	Record subtype = M, field mode record.
3-	Mode definitions on the form: : m,c : m,c : m,c...... where m defines a mode number and c the read control code to hold for the mode number. Mode numbers should be given in ascending order. example FM: 1,R_:3,0_

C.1.9 Validation

Validation record(s), if any, are to be put immediately after the field record to be validated.

If more than one validation record is given, the validation must be of the same kind (position 2 - 7 identical).

Pos	Description
1	Record type = V
2 - 2	Kind of validation. D = discrete, I = interval.
3 - 3	Relational operator. '=' equal or belonging to ' ' not equal or not belonging to.
4 - 5	Position in the field where validation is to start.
6 - 7	Length of validation. Note: start position + length of validation - 1 field length.
8 -	String with validation values, each value is length long and the string is terminated by a ' ' character.

C.1.10 Help texts

Help text record(s) shall be declared after validation records, if any, else after the field record to which the help text belongs.

Pos	Description
1	Record type = H
2 - 3	Row no. where the help text is to be written 01 - 24.
4 - 5	Column no. where the help text is to be written 01 - 79.
6 - 7	Length of the text 01 - 79.
8 -	Text.

C.1.11 Units

The unit record defines the name of the unit check routine to be called for a certain unit. The unit record shall be placed after the last field defined which belongs to the unit.

Pos	Description
1	Record type = N
2 - 2	Definition of unit 1 - 9, A - V, (makes 31 possible units)
3 - 8	Name of the unit check routine.
9 -	Reserve.

C.2 Picture definition with texts and fields (record types P - N)

You start by defining a simple picture, sufficient to test reading and writing. A P-record defines the picture name, T-records define text and F-records define the fields of the picture.

```
+-----------------------------------------------------------+

        *
        *EXAMPLE OF PICTURE DEFINITION
        *
        EXAMP
        *
        T010130TEXTS ARE WRITTEN IN DEFINED ORDER
        T101026LEAD TEXT FOR FIRST FIELD:
        T151012SECOND FIELD:
        *
        F1037040400R_N___00R0
                           FX FIELD1
       F1523050500R_AP__01L
        *                FX FIELD2

+-----------------------------------------------------------+
```

The first four numbers (0101) of the T-record define the line (01) and column position (01) where the text is to be written. The next two numbers (30) define the length of the text; the text is padded with blanks to the end. The first four numbers (1037) of the F-record define the line (10) and column position (37) for the field. The next four numbers (0404) give, in pairs, the length of the field and the display length of the field. These have the same value if no display editing is done. The control code R_ specifies that the field is required. The check code N_ specifies that only digits may be entered into the field and the 00 means that the minimum number of characters to be entered is zero.

The character R, in the field definition, determines how the field will be justified when read. Here, R stands for Right-justify.

After the justification declaration follows the filler to be used when adjusting the field, here it is 0.

The defined picture will be written as:

```
+--------------------------------------------------------+

     TEXTS ARE WRITTEN IN DEFINED O


     LEAD TEXT FOR FIRST FIELD: XXXX


     SECOND FIELD XXXXX

+--------------------------------------------------------+
```

Note the truncation of texts DEFINED. O and SECOND FIELD. This is because the lengths defined were not large enough to accept all of the text. The X's are used to mark field positions.

C.3 Validation and Helptext Definition

You now want the values to be read into the first field to be in the ranges 50-250 and 2050-2100 and you also want to be able to print a helptext giving information about this limitation. To do this you do the following:

```
+--------------------------------------------------------------------+

  *
  F1037040400R_N___00R0
                  FX FIELD1
VI=010400500250205021OO
  H240170VALUE MUST BELONG TO THE INTERVALS  50-250  or  2050-2100
  F1523050500R_AP__01L
  *               FX FIELD2

+--------------------------------------------------------------------+
```

Here, a validation record and a helptext record are placed after the field record.

The validation record V, specifies by I= that validation values consist of intervals, between which the value shall belong. The four following numbers (0104) specify that the validation shall start in the first position of the field and validate the four following characters according to the values given. Note that the validation of the read data is applied after the field has been adjusted.

The helptext is specified by H240170 VALUE..... This says that the helptext is to be written on the bottom line (24) starting in the first column and is 70 positions long. Any number of helptexts and texts can be given on the same positions.

C.4 Check Routine Definition

If you want to make a check on the first field, that cannot be done with validation, you may define a check routine for the field.

The name of the check routine is given in the 24th position of the field definition. The routing is called when a field has been read.

```
+-------------------------------------------------+

        *
        F1037040400R_N___00R0  CHECK

+-------------------------------------------------+
```

Now write the routine check and link it together with your program.

C.5 Unit Check Definition

Any number of fields can be defined as belonging to a unit.

The defined unit check routine is called, on passing the last field of the unit, if any field of the unit has been read.

Unit checks are accomplished by specifying a unit (after the filler in the field definition) for the fields you want to belong to the unit.

The name of the unit check routine is specified by a unit declaration record after the field defined as being the last field of the unit.

```
+------------------------------------------------------------+

     F1037040400R_N___00R0 N1
     F1523050500R_AP__01L N1
     N1UNITRO

+------------------------------------------------------------+
```

In the example above, two fields are specified as belonging to unit one (N1). For unit one, the unit check routine UNITRO is specified.

Now you write your unit check routine and link it together with your program.

C.6 Display Editing of Field Definition

Sometimes, you want the information in a field to be displayed on the screen in a different way to how it is stored in the IO-area.

This can be done in two ways; either by using the edit pattern facility or defining the name of a user-written-editing-routine, together with the field definition.

Edit pattern

Edit patterns are used to edit numeric information.

Using edited patterns to edit a field, you give an E in the 36th position of the field definition. In the positions following the E you specify the pattern.

```
F1037040400R_N___00R0                E09 009:9*0X
```

The two positions following the E gives the length of the pattern. This pattern will, when the field is written (or rewritten), insert a : between the next-to-last and the last digits.

The first zeros of the pattern declare that these digits will only be written if non-zero, or else they will be substituted by the filler character. The filler character is the first character of the pattern. This immediately follows the length. Here, it is blank.

The last three characters of the pattern (*0X) are sign-characters written immediately before the first written digit depending on the sign, in the order -0+. -1 in the field described above, will be written as *0:1. 222 will be written as X22:2.

Between the last digit specification of the pattern and the sign-dependent characters, you can specify trailing characters that will be written only if a negative sign is found in the string.

Edit routine

If you want to edit a field that cannot be done using the edit pattern, you may use an editing routine. This is done by specifying the name in the 30th position of the field definition record.

```
F1037040400R_N___00R0          EDITRO
```

Note that the length of the field is specified as four positions, but the edited length is specified as 12 positions. Now you write your editing routine:

APPENDIX D LIMITATIONS

Number of:	maximum
Special characters	16
Strings	999
Pictures	(WORDPP-1)/(7/BYTEPW+2) where WORDPP is words per MIMER-page.
Pictures handled by editor	20
Fields in a picture	200
Check + Unit check routines	80
Edit routines	31
Special character routines	16
Units within picture	31
Number of message areas	4
Number of prompters	10

APPENDIX E ERROR MESSAGES FROM THE MIMER/SH COMPILER.

If an error is detected during the compilation of the picture definition language, an error code is written to the right of the listing file.

1. File errors

1.1	Read error, from terminal.
1.16	Open error, screen file.
1.17	Open error, source file.

3. Exit character errors

3.1	More than 16 exit characters.
3.2	Unit switch not equal to 0 or 1.
3.4	Conversion error, special character.
3.5	Conversion error, unit switch.

6. Include command errors

6.1	More than one included level.
6.2	Open error, include file.
6.3	Read error, source file.

7. Source file errors.

7.2	Read error, source file.

8. String errors.

8.1	Conversion error, string number.
8.2	Conversion error, string length.
8.3	String length not in range (1-79).
8.4	String number not in range (1-999).
8.6	Switch type error. Internal compiler error.

9. Language file errors.

9.1	Open error, language file.
9.2	Read error, language file.
9.3	Specified language does not exist.
9.4	Read error, source file.

10. Picture name errors.

10.1	No room for form name in picture name table. See limitations.

11. Text specification errors.

11.1 Conversion error, text length.
11.2 Conversion error, row number.
11.3 Conversion error, column number.
11.4 Conversion error, text length.
11.6 Text length not in range (1-79).
11.7 Row number not in range (1-24).
11.8 Column number not in range (1-79).

12. Routine number error.

12.1 Routine number does not exist in table.
Internal compiler error.

13. Field specification errors.

13.1 Conversion error, length of edit pattern.
13.2 Edit length not in range (1-30).
13.3 Conversion error, row number.
13.4 Row number not in range (1-24).
13.5 Conversion error, column number.
13.6 Column number not in range (1-79).
13.7 Conversion error, length of field in I/O-area.
13.8 Conversion error, length of edited field on screen.
13.9 Length of edited field on screen not in range (1-79).
13.10 Conversion error, number of decimals.
13.11 Illegal control code for mode 0.
13.12 Illegal check code.
13.13 Illegal check code extension.
13.14 Conversion error minimum no. of characters.
13.141 Minimum no. of characters greater than field length.
13.15 Right or left justified, position not R(H) or L(V).
13.16 Conversion error, unit check routine number.
13.17 Unit check routine number not in range (1-9, A-V).
13.20 Too many fields in picture.
13.22 Illegal mode specification (FM record).

14. Validation specification errors.

14.1 Length of validation values, conversion error.
14.2 Type of validation not 'T' or 'D'.
14.3 Operation code not '=' or ' '.
14.4 Conversion error, position in field where validation begins.
14.5 Conversion error, length of valid values.
14.6 Position + length -1 > length of field.
14.7 No ' ' mark for end of values.
14.9 Length of validation values, conversion error.

15. Help text specification errors.

15.1 Conversion error, length of help text.
15.2 Length of help text not in range (1-79).
15.3 Conversion error, row number.
15.4 Row number not in range (1-24).
15.5 Conversion error, column number.
15.6 Column number not in range (1-79).

16. Units specification errors.

16.1 Conversion error, unit number.
16.2 Unit not in range (1-9, A-V).
16.3 Specified unit check routine not used in picture.
16.5 N record, but no fields (or text) in picture.

17. Command sequence errors.

17.1 Wrong record type.
17.2 Record sequences not allowed.
17.3 Exit Character record misplaced.
17.4 Option record misplaced.
17.5 Clear character record misplaced (duplicated).
17.7 String switch misplaced.
17.8 Language record misplaced (duplicated).
17.9 Read error, record.

18. Fortran routine errors.

18.1 Internal compiler error.
18.2 Write error, Fortran routine file.
18.3 Conversion error, routine number.
Internal compiler error.

30. Miscellaneous errors.

30.1 Record type not allowed.
30.2 Open error, Fortran routines.

APPENDIX F

MIMER/SH editor error codes

1-5,7-10,13-28 Fatal editor codes. Should not occur.

6	Error reading main picture definition file.
11	Picture sourcefile not found.
12	Error reading picture source file.

Reading errors 6 and 12 occur when the definition files do not match the format specified in the picture definition phase.

APPENDIX G MIMER/SH RUN TIME ERROR MESSAGES

The following messages are displayed when a MIMER/SH routine call fails in some respect and hereby cause program exit.

The format of the error messages is as follows:

```
+-------------------------------------------------------+

     MIMER/SH error: XXX%xxx% text <     > <     >

+-------------------------------------------------------+
```

Here XXX Shows the first three letters of the /SH routine, called from the application program that detected the error. XXX can be one of:

```
CON GET INA INP JMP PRO
PUT RMP SEL WRF WRP SET
RDF INI CLF CLP GST
...
```

which thus signals that routine XXXSH2 made the program stop. XXX = ... should not occur and is an error in the error handling.

xxx gives the name of the internal routine that detected the error condition.

text is the error message associated with the error:

< > Gives more information of the error, in most cases picture name followed by field name.

The following error messages, followed by a description can be displayed:

Application file not found.
The definition file specified in the INISH2 call could not be opened.

Conversion overflow.
The conversion failed due to value which was too large.

Edit pattern error.

Edit routine not found.
The specified display editing routine could not be found. Suggestion: you have made changes in the definitions and the MIMER/SH coupling routine is no longer valid, recompile (with fortran compiler) the coupling routine and relink application program.

Error text missing **

Exit routine not defined.
See "edit routine not found".

Field not numeric.

Init file faulty terminal no.
The specification of terminal number in the initiation file is incorrect.

Internal space area exhausted.
There is no more room in the internal space area.

Invalid field specification.
Syntax of field specification is not correct.

No room on screen for field.
Picture offset was set to a value that made the field go over the scope of the screen.

No default picture exists.

No such field name in picture.

No name for field given.

String not defined.
String with given number has not been defined.

Open error terminal def. file.
The system terminal definition file was not found or could not be accessed. See machine dependent information.

Picture not initiated.

Picture not defined.

Read error MIMER/SH init file.
The initiation file could not be read.

Read error terminal def. file.
The system terminal definition file could not be read. The file may be corrupt.

APPENDIX H RESTART OF INTERNAL SPACE AREA

Internal space area

MIMER/SH uses an internal storage area to store information and status parameters.
If many pictures are initiated at the same time or if pictures have many fields, the default setting of the size of this area may not be sufficient, you get the error:

Internal space area exhausted.

You may then either alter your program to use less storage (e.g. by removing pictures you do not currently use, using RMPSH2) or raise the size of the storage area.

The size of the storage area can be altered by changing a setup routine and relinking your program:

```
      SUBROUTINE EXITSH

C
C--   FORTRAN example of setup routine for mimer/sh
C
      INTEGER SPACE
      COMMON/POOSH2/SPACE(size)
      SPACE(1) =size
      RETURN
      END
```

Here 'size' is a number, the default for 'size' is 1000; this you may alter to e.g. 2000:

EXAMPLE **

```
      SUBROUTINE EXITSH

C
C--   FORTRAN example of setup routine for mimer/sh

      INTEGER SPACE
      COMMON/POOSH2/SPACE(2000)
      SPACE(1)=2000
      RETURN
      END
```

** EXAMPLE END

This routine you link together with your application either in front of the MIMER/SH library or substituting the supplied routine EXITSH depending on machine type.

APPENDIX I TERMINAL INDEPENDENCY AND TERMINAL READ MODES

Terminal read modes

Reading pictures (or forms) from a terminal may be done in three different logical ways:

i) Character-wise read, reading character by character,
ii) Field-wise read, reading a field at a time,
iii) Screen-wise read, reading a number of fields at a time (picture),

before processing the entered information.

The physical read of information (the actual terminal communication) can also be grouped in the same way.

a) Character by character reading communication.
b) Field wise reading communication.
Computers using front end computers or switching networks usually communicate this way.
c) Screen wise reading communication.
Large batch oriented machines using synchronous terminal communication usually communicate this way.

MIMER/SH supports at present the character and field wise reads and will in a coming version support the screen-wise read.

Character-wise read

Each character read from the terminal is sent from the terminal to the screen handling routines.
These check each read character and. acts immediately as the character has been read.

Field-wise read

A number of characters are typed in at the terminal and sent from the terminal (or front end) when a termination key is pressed. This key is usually carriage return.
This means that the screen handling routines cannot act upon entering characters until the user presses the termination key.

Screen-wise read

The user types in a number of fields at the terminal
He also uses the terminal specific commands to reposition in between the fields he wants to enter.
The immediate checks (done at the terminal) of the information entered are delimited to the ones the terminal may perform (usually numeric/alphanumerical checking).

When the user is ready to enter the picture he presses a send key, and the picture is sent to the screen handling routines.

Here one has to be careful developing applications working in character or field-wise modes that are to be run in screen-wise mode as well.

Terminal independency

Do not mix MIMER/SH terminal I/O with other terminal IO. Use a space inbetween texts and fields when defining layouts of pictures.

Using the MIMER/SH editor on a system with fieldwise input

A system with fieldwise input is a system where the central computer itself cannot act upon the input until the terminal operator has pressed some kind of "send" key, normally carriage return. Typically, these are large systems having a front-end computer, buffering the input line until the operator hits carriage return. Examples of such systems are Control Data Cyber, IBM/GUTS, and Univac. The following is a short user's guide to a version of the MIMER/SH editor that is adapted to this type of operation.

The editor, in this version, acts like the character-wise input variant once carriage return has been pressed. Every time you want the editor to react to your input, you must press carriage return. On the "common definition" and "field definition" picture, this means you should press carriage return after each field entered, with the exception that you can "type-ahead" several arrows to skip fields - e.g. if you want to skip three fields ahead, you could press forward arrow three times and after this a single carriage return.

On the "text definition" picture, you can move around freely using the arrows to place out texts - when you press carriage return the editor will re-write your input and show row and column number in the lower right corner of the picture. However, you must press carriage return often enough so that you do not overrun the maximum number of characters that your front-end computer, or any other limiting factor in your system, can buffer. This, however, is immediately obvious when it happens, either via an error message from the front-end or via the fact that all of your input is not re-written - i e the cursor stops somewhere where you were not finished. A control-R plus carriage return in this situation refreshes the screen and shows you what has been actually entered.

Some fields in the "field definition" and "string definition" pictures, namely

- Edit mask row
- Validation row
- Help text row
- String row

are treated in a somewhat special manner.

- When you first enter input on these rows and press carriage return, your input will be re-written and the cursor is positioned to the right of the last character you entered.

- After having done so, you may use the back arrow and forward arrow to move into the string again if you want to change anything. Everything to the right of the cursor will be disregarded when pressing carriage return.
- The final action (message "Validation row stored", etc.) is taken when you press a single carriage return without any other input before it.

A final hint: By using programmable function keys, you could add several useful functions to your terminal. For instance, the arrows immediately followed by cariage return programmed into four function keys could save you some key-pressing and a function key holding the string ctrl-C - D - carriage return would act as a delete-character key on the "text definition" picture.