MIMER/SH
SCREEN HANDLER

USER GUIDE

Version 3.2
April 1984

MIMER/SH USER GUIDE

Chapter **Page**

This manual is divided into two sections - a user guide and a reference manual.

The user guide provides a general introduction to the Editor which is used for screen building and screen handler routines which are used to build application programs with the defined screens.

The reference manual provides more detailed information about different routines and about the checking, editing and validation options available.

1. INTRODUCTION

MIMER/SH consists of four main parts:

- An editor for interactive definition of screen pictures (SHE).
- A compiler that creates MIMER/SH object code from the picture definitions (SHC).
- A subroutine package to manipulate the screen from an application program.
- A utility to list picture definitions (SHL).

MIMER/SH has, like MIMER/DB, been made as hardware independent as possible by using a number of machine dependent base routines. By writing the application program in a machine independent manner (using MIMER/DB for data base manipulation) a system is created which is machine independent.

MIMER/SH is specially designed to present/read information that goes together in records but is to be read in pieces. Each piece of information is associated with a field. A field is a number of continuous positions on the screen. A number of lead texts and fields constitute a picture.

All fields of a picture can be read and checked in one call from the application program.

MIMER/SH makes it easy to alter texts and picture definitions, as information is retrieved at run time from a MIMER/SH object file. This can then be altered without recompiling and linking the program.

1.1 Programmer tools provided by MIMER/SH

Tools for information handling include:

- the facility to reference fields by names.
- association of fields on the screen with an internal storage area, created and maintained by MIMER/SH.
- definition of paths for information transfer between the screen and the data base.

Tools for information checking include:

- Type checks — Each field is defined to accept numeric, decimal, alphabetic, alphanumeric or all printable characters.
- Minimum fill — Definition of the minimum number of characters that must be entered into the field.
- Validation — The information has to belong/not belong to a defined set of values.
- User written field checks — Check routines automatically called after reading the field.

Tools for editing the information on the screen include:

- Pattern editing of numeric information, including suppression of non-significant digits, editing delimeters (arbitrary characters) into numbers, sign handling.
- Automatic calls to user written Editing routines in which the programmer performs his own editing.

Tools for controlling reading include:

- Use of a built-in read control facility to determine how the reading is to proceed after information is read. Here, one can define exit characters that will either terminate the reading of a picture or, enter an exit routine if one has been defined.

Other convenient facilities are:

- Text strings brought into the application program by a call to MIMER/SH.

- Help texts for each field which are written when a '?' is given in the first position of the field. Texts are stored in the picture definition file which makes them easy to alter and saves program space.

- A dialogue function to read and write freely on the screen.

- Prompting function for dialogue messages.

1.2 Application development with MIMER/SH

Application development with MIMER/SH is divided into three parts:

1. **Defining the screen pictures, texts and parameters.** This is normally done by the MIMER/SH editor but can be done with an ordinary text editor. This stage is referred to as the Picture Definition Phase (PD Phase).

   When using a text editor, you specify your definitions record by record according to a Picture Definition Language. When using the MIMER/SH editor you interactively define your pictures on the screen. **Use the MIMER/SH editor where possible but it is only available on certain machine types.**

2. **Compiling the MIMER/SH source file generated in the PD-phase.** The compiler produces two output files:

   a) A MIMER/SH-object file containing the definitions in an internal format. This file will be used by the Application Program at runtime.

   b) A Fortran file coupling MIMER/SH to the user written check, editing and exit routines. This file must be compiled with a Fortran compiler and linked together with your program.

3. **Making your application program using calls to MIMER/SH.** Together with the application program, you write the check, editing and/or special character routines you may have specified during the PD-phase.

   You then link the application program, eventual check-, editing-, exit routines, coupling file and MIMER/SH-module library together and run. This process is illustrated in Fig. 1.1.

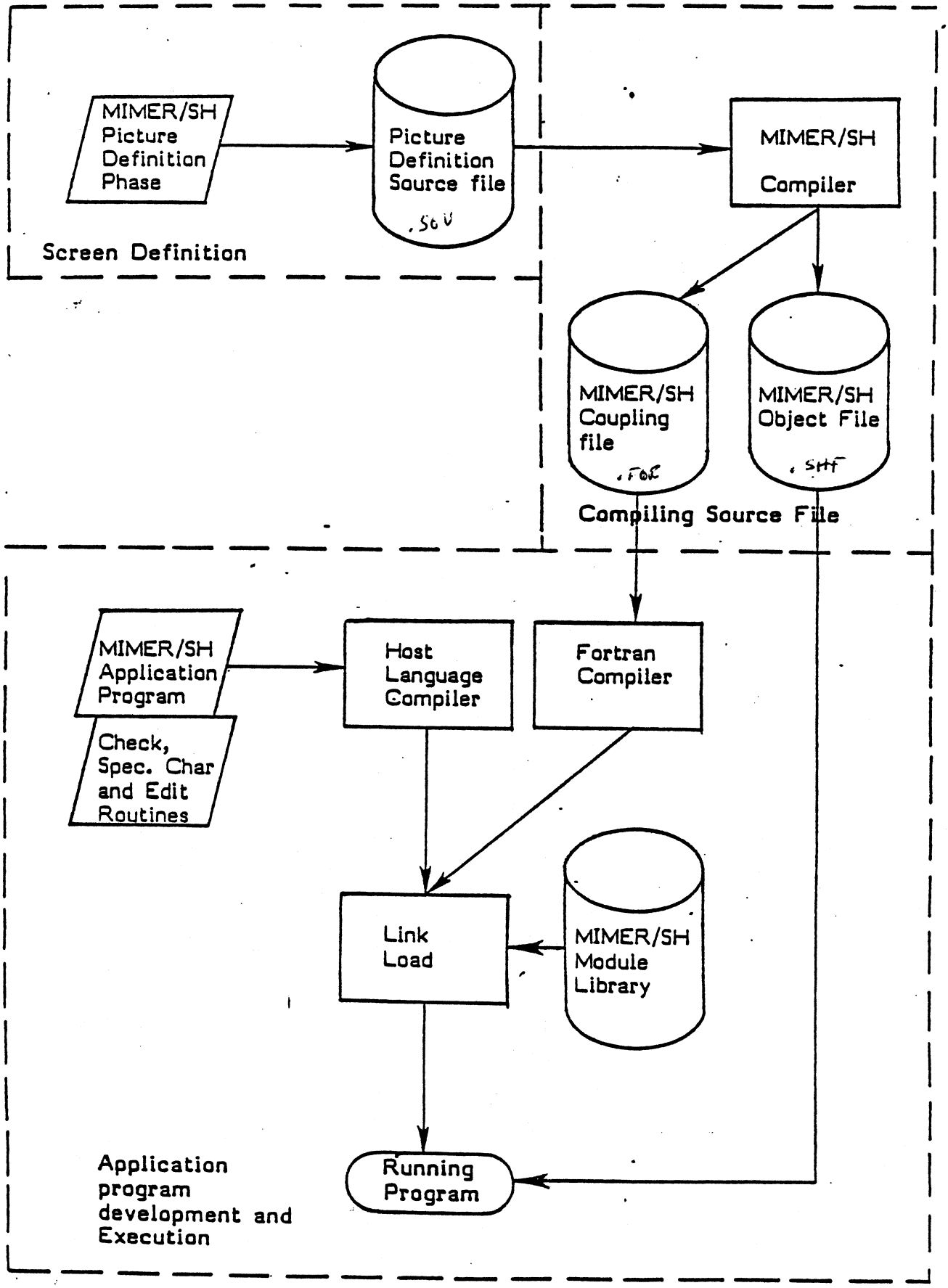

Fig. 1.1 BUILDING AN APPLICATION USING MIMER/SH

2. DEFINING PICTURES

2.1 Defining Pictures using the MIMER/SH Editor

The MIMER/SH Editor is used to interactively create, modify or delete Picture Definition source files from a screen terminal. The source files created by this editor have the syntax described in the Picture Definition Language, (description in Appendix A of the Reference Manual). Picture Definition files created by the editor can thus be created or altered using an ordinary text editor if so required. When a edit session is finished the generated source file has to be compiled using the MIMER/SH compiler.

2.1.1 Running the editor

When the editor has been started, a list of the available terminal types is presented and you are prompted for the type you are using. MIMER/SH then accesses an internal file of terminal configurations to continue processing. Next, the 'File Definition' picture is displayed. Here you specify the name of the source file you want to work with.

The editor consists of four more pictures:

- the 'Common definition' picture
- the 'String definition' picture
- the 'Text of picture definition' picture
- the 'Field definition' picture.

For each item which can be specified it is possible to ask for help by typing a '?'.

To get help on how to use the editor, type Ctrl-V (i.e. press Control key and V key at the same time).

2.1.2 General control characters and display features of the editor.

(forward arrow, forward tab), (backward arrow, rubout)
(upward arrow), (downward arrow)

are used to 'walk around' definition pictures.

When defining Strings, Texts, Helptexts, Validations and Edit-patterns the following hold:

- space is displayed as underscore.
- nothing is displayed as space.
- illegal commands or definitions produce a bell.

2.2 File definition picture

This picture is the first picture presented when entering the editor, or when control B is given in another picture. Fig. 2.1 illustrates this picture.

Here, the following specifications are made (the numbers refer to Fig. 2.1.):

1 **Field marks**
In the Text definition-picture, fields will be marked on the screen according to the settings of the field marks, as follows:

1st character is used to mark the beginning of a field
2nd character is used to mark the middle of a field
3rd character is used to mark the end of a field
4th character is used to mark the excess characters for edited length.

EXAMPLE: **********

A field defined with a length of 5 and an edited length of 7 and with the field marks set to: XXXx will be presented as
XXXXXxx

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

2 **Picture Definition source file name**
Here, you may specify the name of an existing source file or a new source file name. If the editor finds the file specified, the user is prompted whether to read, not read or delete the file. If you choose not to read the file, you are prompted for a new file name.

Control Commands

Legal control commands from this picture are:
Ctrl-V Show short description of editor commands.
Ctrl-P Make hard copy of screen on local printer.
Ctrl-R Refresh screen.
Ctrl-D Change to 'Common Definition' picture.
Ctrl-W Change to 'String Definition' picture.
Ctrl-T Change to 'Text Definition' picture.
Ctrl-X Exit from editor.

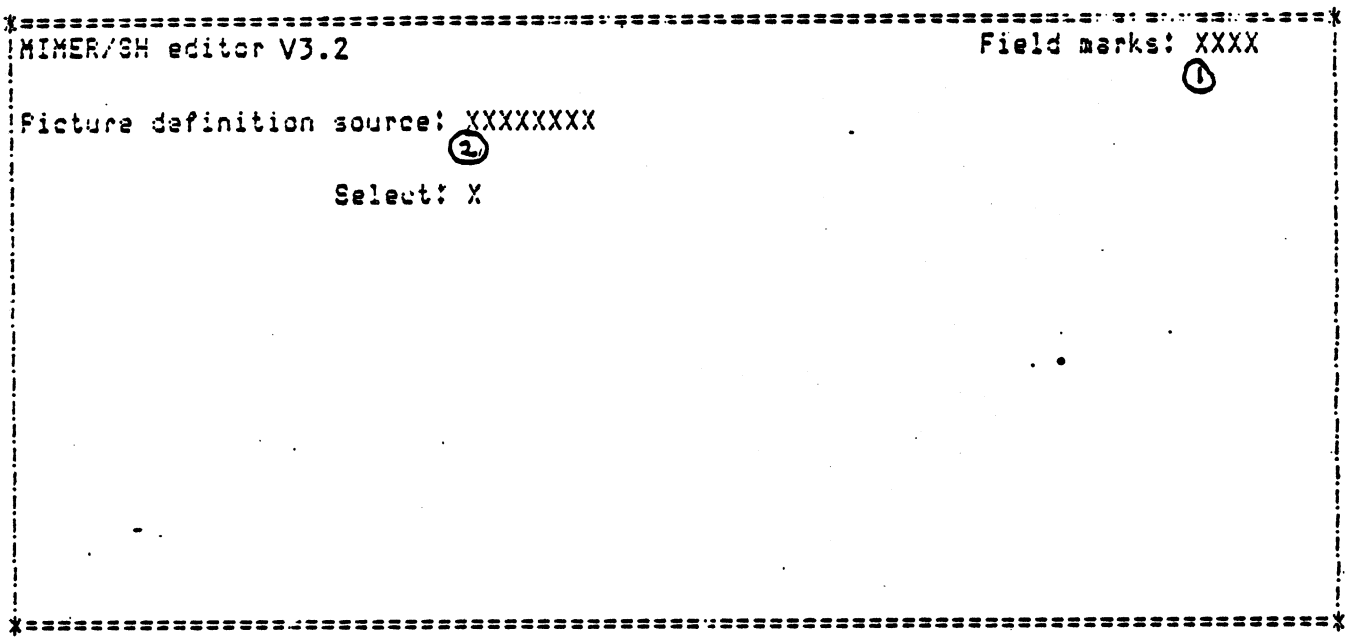

Fig. 2.1 Field Definition Picture.

2.3 Common definition picture

With this picture, the user makes definitions which are common to all pictures. The following definitions are made: (the numbers refer to Fig. 2.2)

1. **Exit character.** The decimal number for the character you want as exit character.

2. **Check of information.** Here a yes or no answer defines if picture information check should be done when the exit character exits from reading the picture. The check consists in checking that all required fields have been given and unit checks are satisfied.

3. **Name of exit routine.** If a routine name is given, this routine will be called when the exit character is given.

4. **General fill character.** This character will be used to mark the field positions when fields are cleared with the clear-fields routine. Also used to mark answer positions when prompting with prompting routines.

5. **Language.** The language in which MIMER/SH internal messages will be written.

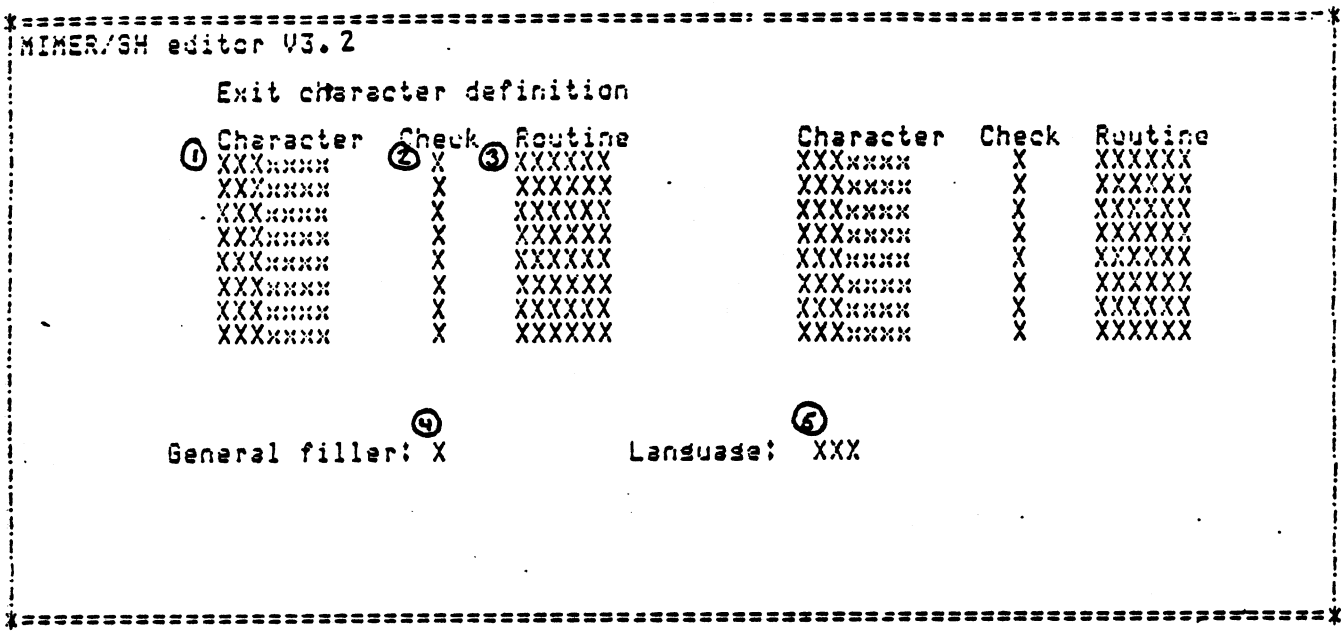

Fig. 2.2 Common Definition Picture.

2.4 String definition picture

This picture is entered when a Ctrl-W command is given. Fig. 2.3 illustrates this picture.

Here, you may define or redefine your string definitions. (The numbers below refer to Fig. 2.3).

1 **You specify the string number;** if the string already exists it is shown on the screen.
If 'return' is given, the next free string number will be given.
If 'L' is given, a listing of defined strings will be given.

2 **You may enter new text or alter an existing string.**
When 'return' is given, the string will be stored up to the current cursor position. If there is any text following, it will be deleted.
If 'return' is given in the first position, the string will be deleted and the string number freed.
The current position within the row is displayed at **A**.

Control Commands

Legal control commands from this picture are:
Ctrl-V Show short description of editor commands.
Ctrl-P Make hard copy of screen on local printer.
Ctrl-R refresh screen.
Ctrl-B Change to 'File Definition' picture.
Ctrl-T Change to 'Text Definition' picture.
(down arrow) Positions the cursor at the end of the string when typed within the string text area.

```
*=============================================================================*
!MIMER/SH editor V3.2  String definition                                      !
!String no: XXX ('L' = list / RETURN / New string number)                     !
!          (1)                                                                !
!XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX!
!col (A)     (2)                                                              !
!                                                                             !
!                                                                             !
*=============================================================================*
```

Fig. 2.3 **String Definition Picture.**

2.5 Text definition picture

This picture is entered when a Ctrl-T command is given. This picture consists of two parts:

Part 1
This is where the name of the picture to work with is selected. This part is entered when Ctrl-T is given for the first time, i.e. when no earlier picture has been worked with or when Ctrl-X has been given in the second part of the text definition picture.

The picture description is shown in Fig. 2.4.

The source file **A** and all defined pictures **B** are displayed. The name of the picture to work on, new or old, is prompted from the user **1**. When the user has given this, the second part is entered.

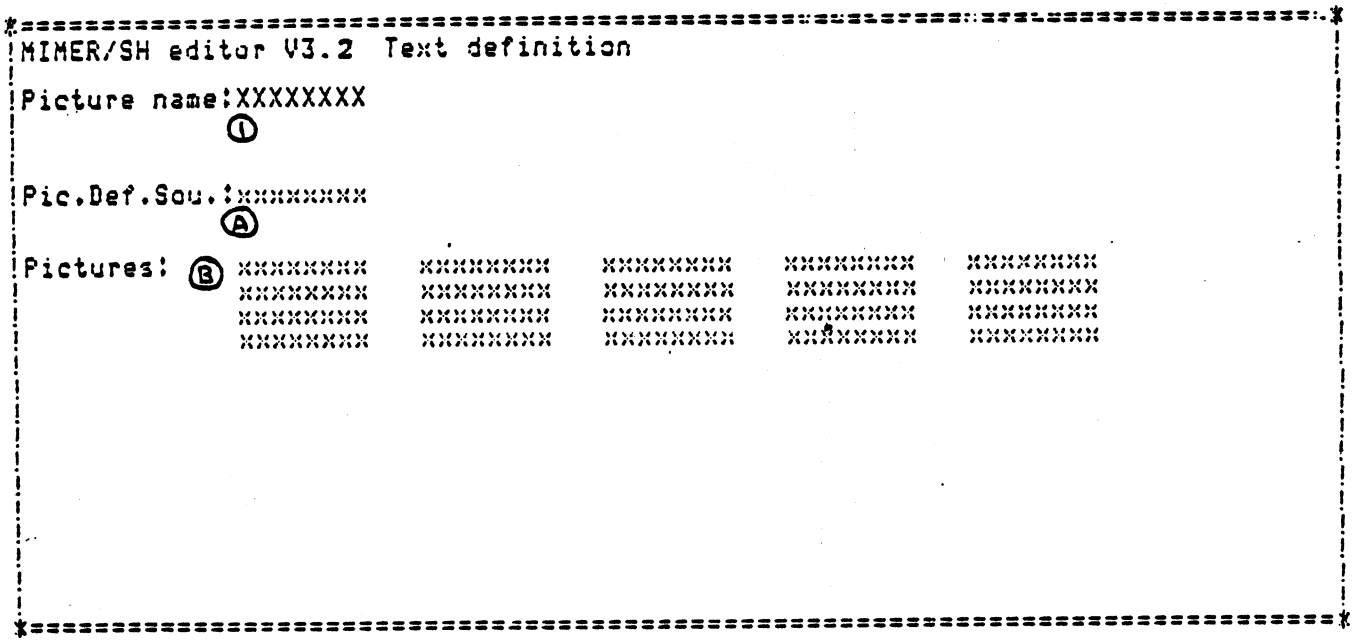

Fig. 2.4 Text Definition Picture Part 1.

Text definition picture Part 2
This is where text and fields of the selected picture are to be defined or redefined,(as shown in Fig. 2.5.). The picture name A and the current cursor position C are displayed on the bottom row. The field number is displayed B when a Ctrl-N is given. Texts are defined by typing into the desired positions of the picture, which are reached by giving arrows (forward tab, rub, del). Texts can be edited with the commands Ctrl-L and Ctrl-C (see below). Fields are defined by positioning the cursor at the position where the first position of the field is to be located and then issuing the Ctrl-F command. Now the 'Field Definition' picture is entered. Here you define the field (see below). When this has been done, you return to the 'Text definition' picture Part 2 (giving Ctrl .T), which now has the defined field positions marked in the picture (according to the settings of the fieldmarks as above).

Control Commands

Legal commands for both parts of the Text Definition picture:
Ctrl-V Show short description of editor commands.
Ctrl-P Make hard copy of screen on local printer.
Ctrl-R Refresh screen.
Ctrl-B Change to 'File Definition' picture.
Ctrl-W Change to 'String Definition' picture.

Legal commands for Part 2 only:
Ctrl-F Change to 'Field definition' picture.
Ctrl-X Change to 'Text definition' picture Part 1.
Ctrl-N Move cursor to next field and show field number.

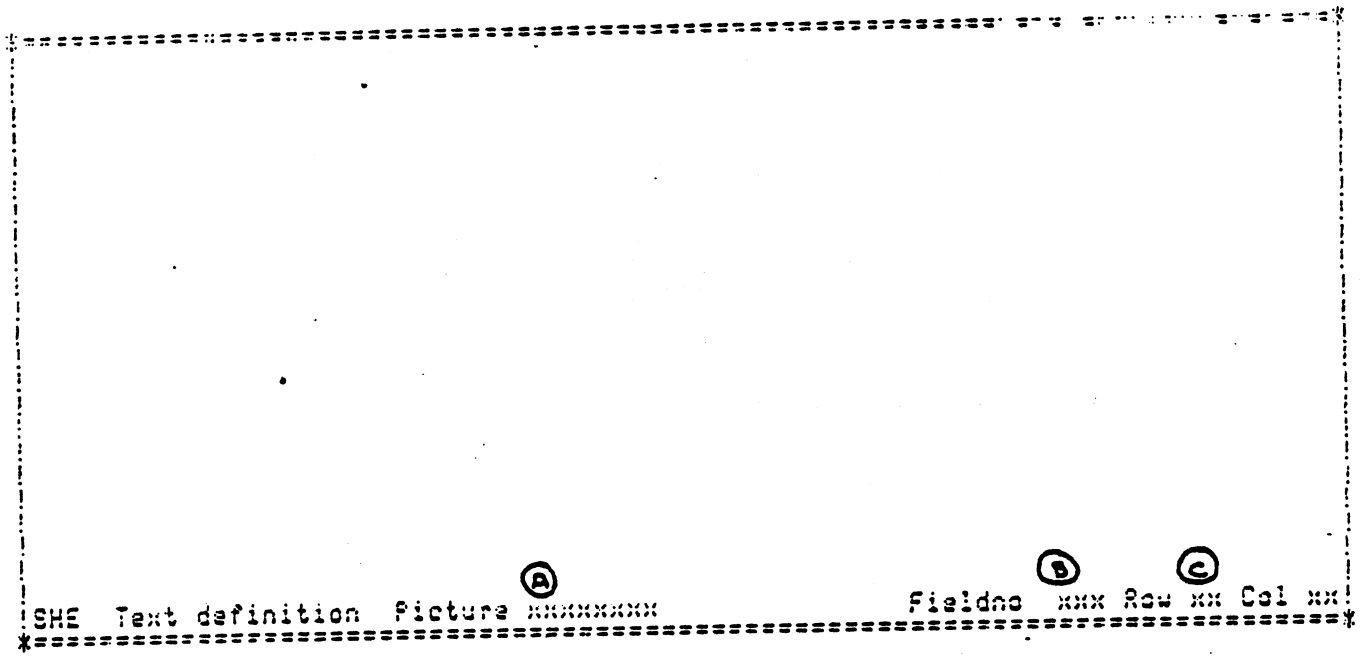

Fig. 2.5 Text Definition Picture Part 2

Ctrl L-Edit Row

Ctrl L is used to edit a row. It works on the row at which the cursor is positioned.

With the text definition picture the following subcommands can be given: (see Fig. 2.6.)

I = Insert empty row. Current row is pushed down and the last row is deleted.

D = Delete row. Rows below are moved up and an empty row is added at the bottom of the picture.

C = Copy row. Works as the I command but the current row will be in place of the empty row. Fields are not copied.

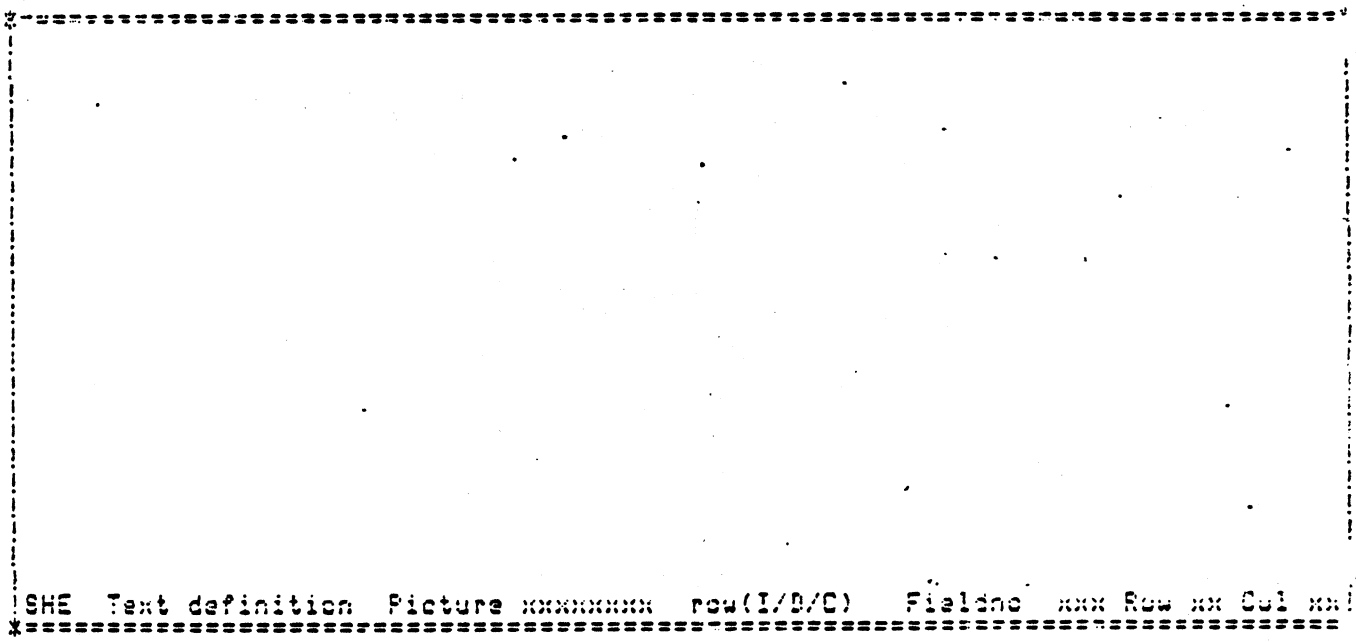

Fig. 2.6 Text definition picture for line editing.

Ctrl C-Edit Character

The command works on the character at which the cursor is positioned.

The following character subcommands can also be given: (see Fig. 2.7)

I = **Insert character.** Space is inserted at the current cursor position and the remainder of the row is moved to the right. The last character of the row is deleted.

D = **Delete character.** Character at current cursor position is deleted and the remainder of the row is moved to the left. A space is inserted at the end of the row.

It is not possible to give the Control-C command inside a field, nor is it possible to insert characters if a field will be pushed over the end of the row.

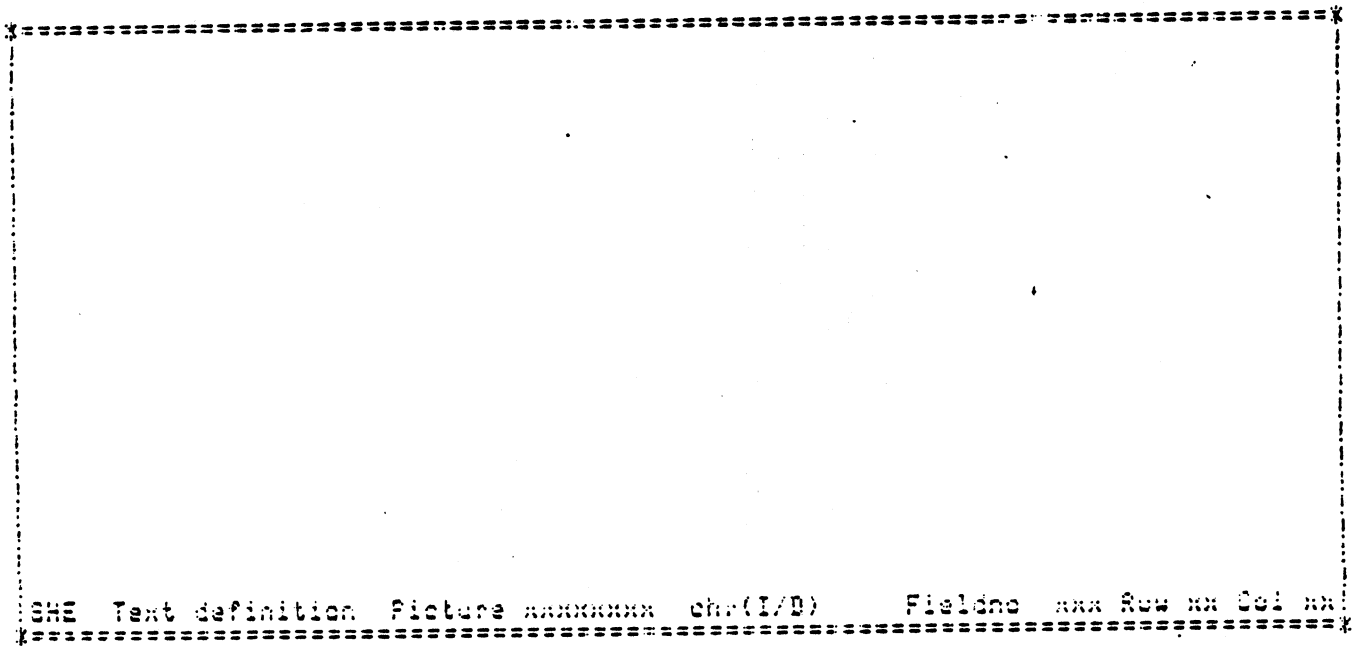

Fig. 2.7 Text definition picture for character editing.

2.6 Field definition picture.

This picture is entered when a Ctrl-F command is given from the Text definition picture part 2. In this picture the definitions of a field are made. If the field is new, some default values will be displayed otherwise the old definitions of the field are displayed.

The following fields must be given:

1. **Field number.** This will be the read sequence number for the field with range 1 - 9999. If carriage return is given, you are prompted whether to delete the field or not. In order to put new fields in between old fields in the read sequence you should displace the field numbers with e.g. an interval of 10. When a previously defined picture is read by the editor, you will then get field numbers numbered with intervals of 10, if updating .

2. **Field name.** This is a unique name for the field within the picture. This name is used to reference the field from the application program.

3. **Field length.** This gives the storage length for the field and the maximum number of characters that can be entered into the field.

4. **Display length.** This is used when the field is edited on the screen to a length greater than its field length.

5. **Control code.** This is a code specifying how the field shall be treated when reading or writing the picture. The field can be specified to be:

R_	= required	, must be given
O_	= optional	, may be given
RB	= required	, must be given but may be blanked.
OB	= Optional	, may be given or blanked.
W_	= write only	, only written
E_	= excluded	, neither written nor read

6. **Check code.** This is a code giving predefined checks for entered information. The following codes can be given:

N_	= numeric	, only 0-9's may be entered
D_	= decimal	, as above with one decimal
A_	= alphabetic	, a-z or A-Z
AN	= alphanumeric	, as N_ and A_ together
AP	= all printable characters.	
DT	= date	, numeric with date check of the form YYMMDD
TX	= text	, as AP but can be edited

Field definition picture

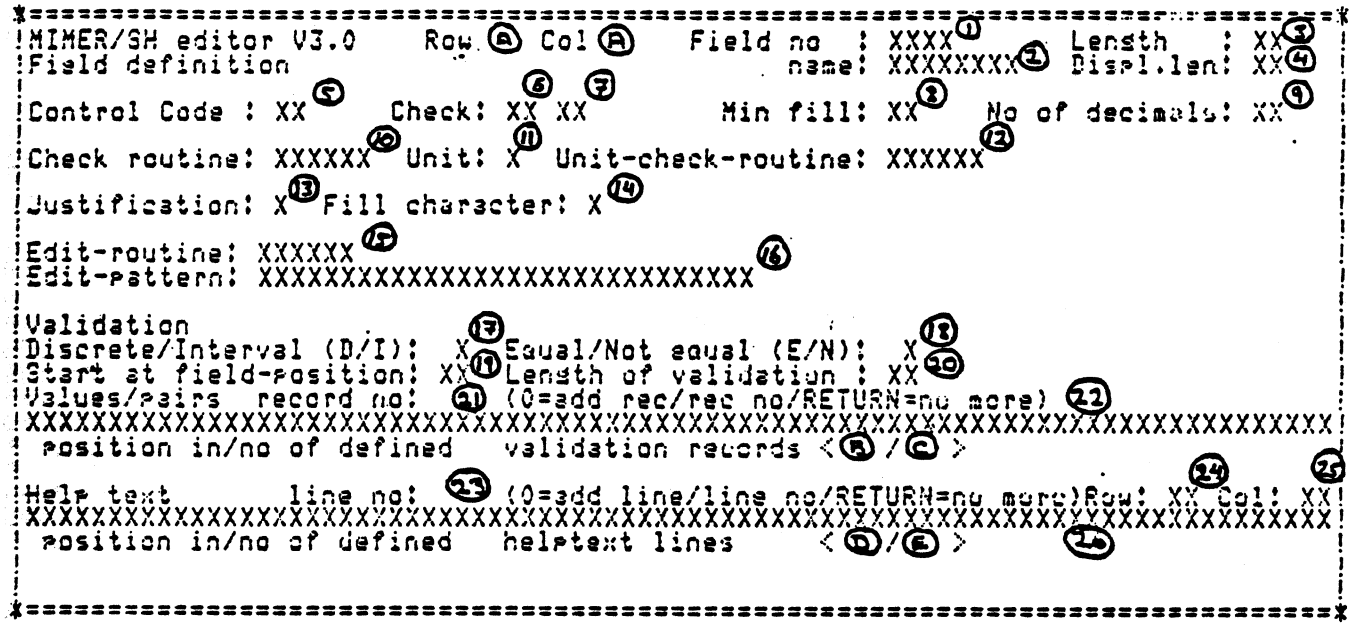

Fig. 2.8 Field Definition Picture.

7. **Additional check.** This is a code giving extended check codes. The following codes can be given:

   with N_ and D_ above:
   - S_ = signed ,+ or - must be given
   - __ = sign/unsign ,+ or - may be given
   - U_ = unsigned ,+ or - may not be given

   with A_, AN, AP and TX:
   - UP = upper case ,convert to upper case
   - — = any case ,no case conversion
   - LO = lower case ,convert to lower case

8. **Minimum number of characters.** Gives the minimum number of characters that must be entered into the field.

9. **Number of decimals.** Gives number of decimals to be given with check code D_.

10. **Check routine name.** Name of user written routine to be called when the field has been read.

11. **Unit identifier.** If the field is to belong to a check unit, the id (identity) of the unit is specified. If the unit has been previously defined in the picture the name of the unit check routine is displayed (12), otherwise you are prompted for the unit check routine (12). A unit identifier is a triggering mechanism that will enter a unit check routine if any of the fields belonging to the unit have been altered.

12. **Unit check routine name.** This is the name of the unit check routine called by the unit identifier.

13. **Justification.** This defines how the read information will be justified when the field has been read.

14. **Fill character.** This is the character used to pad the field when it is justified.

15. **Edit routine name.** If specified, this is the name of the routine that will be called for display editing of the field, when correctly entered or when written to the screen.

16. **Edit pattern.** This pattern determines the display editing of the field, when correctly entered or when written to the screen.

Field Content	Field Length	Displayed	Display Length	Pattern
'0000'	4	' = '	2	' 00-=+'
'1234	4	'2.34'	4	' 9.99- '
'1234'	4	'12.34'	5	' 09.99- +'
'1234'	4	'12.34'	5	' 09.99- '
' 123'	4	'+1.23'	5	' 09.99- +'
' '	4	' 0.00'	5	' 09.99- '
' -12'	4	'-0.12'	5	' 09.99- '
' 00012'	7	'*** 0.12***'	11	'*0,009.99 CR '
' -12345'	7	'* 123.45 CR'	11	'*0,009.99 CR '
'-123456'	7	'1,234.56 CR'	11	' 0 009.99 CR '
'-123456'	7	'1,234.56 '	11	' 0 009.99 CR '
' '	1	' '	2	'CR'
'X'	1	'CR'	2	'CR'

Validation (either none or all of following fields)

17 Type of validation D=discrete or I=interval. If carriage return is given, validation will be deleted.

18 Relation Equal/belonging to or Not equal/not belonging to.

19 Position in the field where the validation is to start.

20 Length of validation values.

21 Validation row to work on:
0=create new row with validations for the field.
n=number, get earlier specified validation row.
carriage return = no more validation rows.

22 New validation values can be entered or old values changed. When 'return' is given, the defined values to the left of the cursor are stored and the values to the right are deleted. The value row is a string defining valid values for the field. Several values can be given on the same row.
e.g. TYPE=D,REL=E,POS=1,LEN=3 and VAL=123124125
means that the values 123, 124 and 125 may be entered in the first three positions of this field.
TYPE=I,REL=E,POS=1,LEN=2 and VAL=11223344
means that the values in the ranges 11-22 and 33-44 may be entered in the first two positions of this field. The cursor position for validation B and the total number of specified validation rows C are displayed.

Helptext (give either none or all fields)

23 Helptext line to work on:
0=create new helptext line for this field.
n=number, get earlier defined helptext line.
return=no more helptexts.

24 Row number within the defined picture for helptext. If carriage return is given, the line is deleted.

25 Column number within the defined picture for the helptext. If column number 0 is specified the helptext will be displayed on a message area. The row number given will then be taken as a message area number.

26 Helptext new or old. When 'return' is given, the text to the left of the cursor is stored and the text to the right is deleted. Cursor position within the line D and the total number of defined helptexts E are displayed.

Legal commands within the Field Definition picture:

Ctrl-V Show short description of editor commands.
Ctrl-P Make hard copy of screen on local printer.
Ctrl-R Refresh screen.
Ctrl-T or Ctrl-X Change to 'Text Definition picture'
Ctrl-K Show Check routines previously specified for this picture.
Ctrl-E Show Editing routines previously specified for this picture.
Ctrl-A Show Unit check routines previously specified for this picture.

2.7 Commands summary for MIMER/SH Editor

Ctrl-V Show short description of editor commands.
Ctrl-P Make hard copy of screen on local printer.
Ctrl-R Refresh screen.
Ctrl-B Change to 'File Definition' picture.
Ctrl-D Change to 'Common Definition' picture.
Ctrl-W Change to 'String Definition' picture.
Ctrl-T Change to 'Text Definition' picture
Ctrl-F Change to 'Field Definition' picture.
Ctrl-X Change to 'File Definition' from 'Text Definition' part 1 and 'Common definition' picture.
Change to 'Text Definition' picture Part 1 from Part 2.
Or exit from editor.
Ctrl-N Move cursor to next field and show field number.
Ctrl-L Edit row, works on the row on which the cursor is positioned.
I = Insert empty row, current row is pushed down
D = Delete row, rows below are moved up
C = Copy row.
Ctrl-C Edit character, works on the character on which the cursor is positioned.
I = Insert character. Space is inserted at the current cursor position.
D = Delete character. Character at current cursor position is deleted.
Ctrl-K Show Check routines previously specified for this picture.
Ctrl-E Show Editing routines previously specified for this picture.
Ctrl-A Show Unit check routines previously specified for this picture.

Fig. 2.9 presents a short command description.

N.B. These codes may change between installations/machines

```
MIMER/SH editor V3.0      Command description
     The Editor consists of 5 pictures:
  0. File                        Define definition file to work on
  1. Common                      Definitions common to all pictures
  2. Picture Text part 1/2       Definition of picture/texts in picture
  3. Picture Field               Definition of field in picture
  4. String                      Definition of strings
  ^X = End editor (pic0) /   'Up one picture level'
  ^B = Change to picture 0 from picture 2  and 4
  ^D = Change to picture 1 from picture 0
  ^T = Change to picture 2 from picture 0, 1, 3 and
    ^L = Line editing(pic2). Prompts for: I/D/C to insert/delete/copy line
    ^C = Char editing(pic2). Prompts for: I/D   to insert/delete character
    ^N = Position cursor in next field and display fieldno (pic 2)
    ^F = Change to picture 3 from picture 2
      ^K = Display defined User-check routine names, valid in picture 3
      ^A = Display defined Unit-check routine names, valid in picture 3
      ^E = Display defined Edit       routine names, valid in picture 3
  ^W = Change to picture 4 from picture 0, 1 and 2
  ^P = Make a hardcopy of screen on locally connected printer
  ^R = Refresh screen
  ^V = Show this HELP text
   ? = Will give HELP for current field (only in first position of field)
```

Fig. 2.9 Help picture called by Ctrl V.

3. BUILDING AN APPLICATION WITH MIMER/SH ROUTINES

Having defined your pictures, you build an application program using these.

Here we will present a small example and discuss the routine calls to the screen handler.

Example 1 shows the routines necessary to read the fields of a picture called CUST.

Example 2 then shows how to link the picture CUSTREQ to the tables of MIMER/DB. In this example a table CITIES in the data base is accessed and one column 'CITYNAME' is written to the screen where a specific search condition is satisfied.

Example 3 demonstrates the use of a user written check routine for the search condition to be applied to the CITIES table.

Example 4 demonstrates the use of exit characters.

Example 5 extends 4 to show how to use user written exit functions.

The routine INISH2 is used to initiate MIMER/SH. INISH2 has two parameters, terminal number and definition. The terminal number must be a variable (i.e. not a constant like 8). If you set this variable to zero MIMER/SH will handle the terminal initiation. The second parameter specifies the name of your definition file. This is the file created as output from the MIMER/SH compiler called picture definition object file. This file is a small database file containing your definitions. You may use screen handler calls, that do not use picture definitions, if you specify a blank file name. There are two variables you have to make definition statements for, to get a simple screen handling program going. These are terminal type variable and the status array. Both are of integer data type and the latter has four elements.

The language used in the following example has an ADA like syntax.

Notation:	begin	-	start of module code
	--	-	following part of the row is a comment
	;	-	end of statement
	"xx"	-	the text string 'xx'
	:=	-	gives a value to a variable

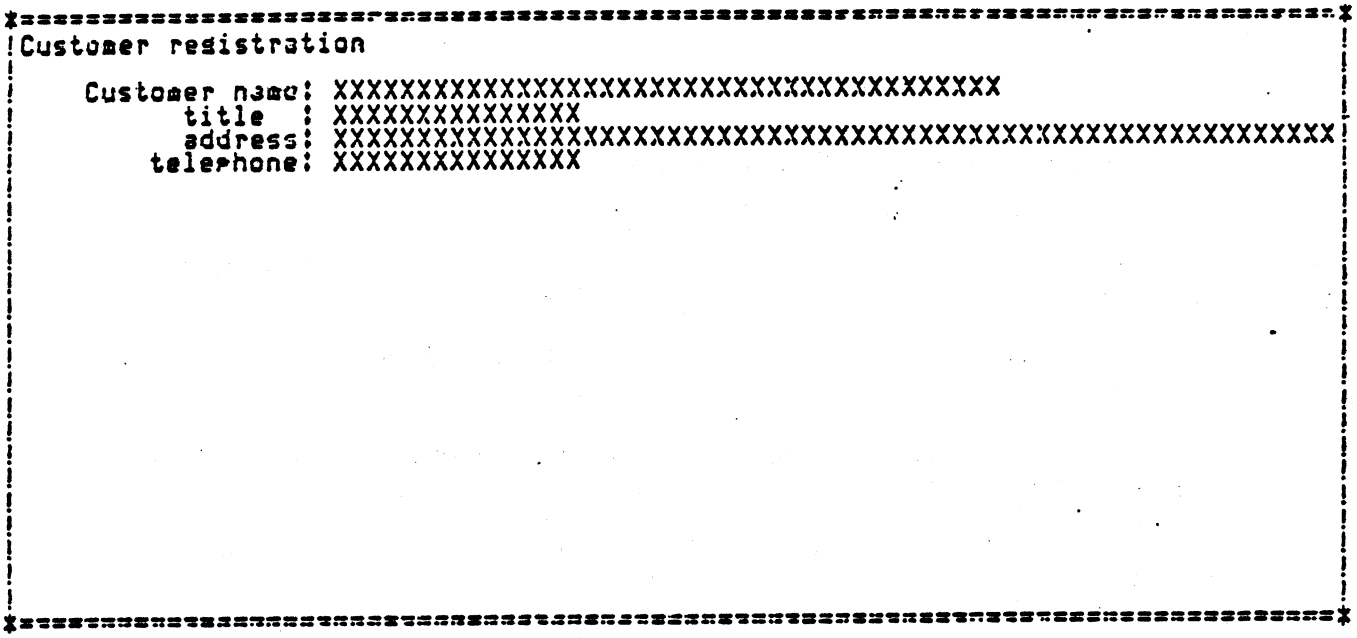

Fig.3.1 picture name : CUST

field name	length	justify	read control	check code	min fill
NAME	40	L	R_ = required	A_ = alphabetic	3
TITLE	15	L	O_ = optional	A_	5
ADDRESS	60	L	R_	AP = All printable	0
PHONO	15	L	O_	AP	0

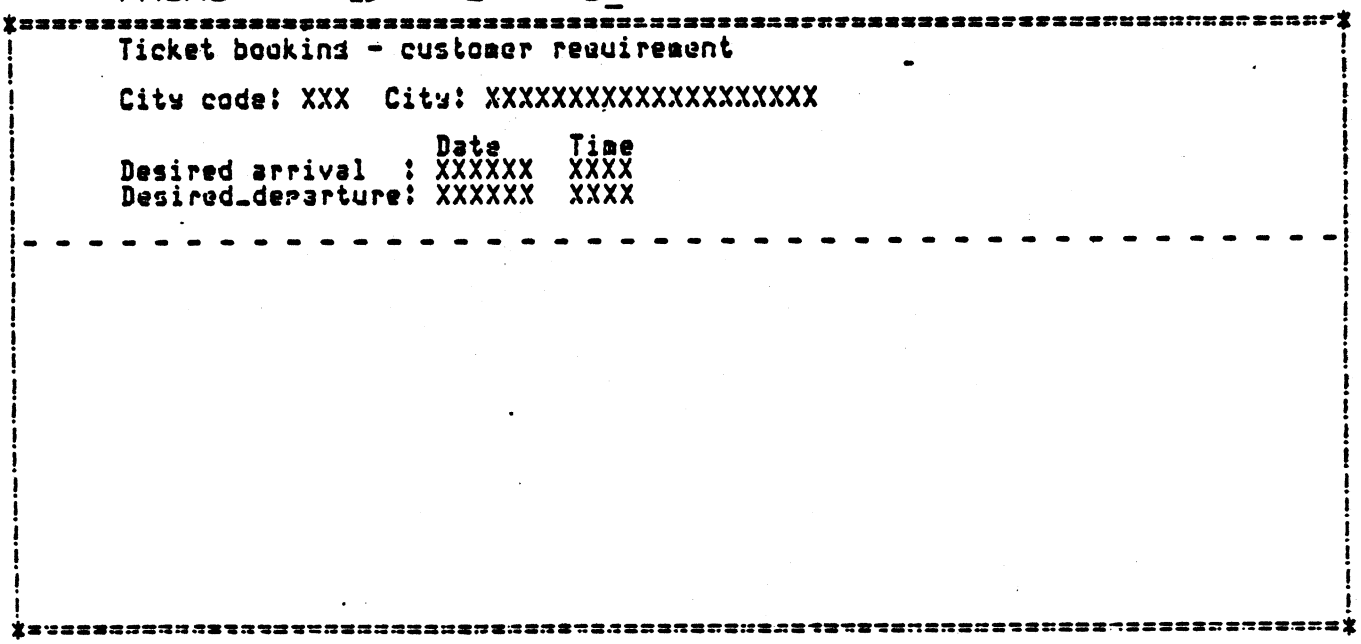

Fig.3.2 picture name : CUSTREQ

field name	length	justify	read control	check code	min fill
CITYCODE	3	L = left	R_	N_ = Numeric	1
CITYNAME	20	n/a	W_ = Write only	n/a	n/a
ARRIVDAT	6	R = right	R_	DT	1
ARRIVTIM	4	L	O_	N_	0
DEPARTDAT	6	R	R_	N_	1
DEPARTIM	4	L	O_	N_	0

EXAMPLE: **********

Example 1 Simple picture write/read
The following routines are introduced:
INISH2 - routine to initiate session.
INPSH2 and INASH2
- routines to initiate a picture. After this call, you may refer to the picture name and field names of the picture.
CLSSH2 - routine to clear the whole or parts of the screen.
WRPSH2 - routine to write the texts of the picture.
CLFSH2 - routine to clear fields from the screen and reset the storage area for the same.
RDFSH2 - routine to read fields from the screen into the storage of each field.

```
     begin
     integer TTYPE,STATUS(4);
--
-- Initiate MIMER/SH using definition file TICKET.
-- Then initiate the picture CUST.
--
     TTYPE:=0;
     INISH2 (TTYPE,"TICKET ");
     INPSH2 ("CUST     ");
     INASH2 ("CUST     ");
--
-- Clear the whole screen and write picture texts
--
     CLSSH2 ("WH",0,0,0,0);
     WRPSH2 ("CUST     ");
--
-- Make a loop that:   Clears the fields of the picture on the screen
--                     and in the storage area.
--                     Read the fields of the picture.  The picture
--                     is exited when the last field is given or
--                     when an exit or arrow function out of the
--                     picture is used.
--                     The loop will only be exited in the latter
--                     case.
     loop
          CLFSH2 ("CUST     ","* ");          -- * denotes all fields
          RDFSH2 ("CUST     ","* ",STATUS);
          exit when STATUS(1) /= 0 ;
     end loop;
     end program;
```

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

EXAMPLE: **********

Example 2 Coupling to MIMER/DB.
The following routines are introduced:
PROSH2 - routine to set up a transfer path between a field of the screen and a column of a table.
SELSH2 - routine to set up a constraint on a column of a table versus the contents of a field, for rows retrieved from the table.
MSGSH2 - routine to write a message to a message area.
For the other used routines see MIMER/DB manual.

```
     begin
     integer TTYPE,STATUS(4);
     integer DID(4),TID(4),USER(2),PASS(2);

     TTYPE:=0;
     INISH2 (TTYPE,"TICKET ");
     INPSH2 ("CUSTREQ ");
     INASH2 ("CUSTREQ ");

     CLSSH2 ("WH",0,0,0,0);
     WRPSH2 ("CUSTREQ ");
--
-- Initiate MIMER/DB and initiate a database and a table
--
     MDRMVB (USER,0,"USER     ",0,8);
     MDRMVB (PASS,0,"PASS     ",0,8);
     BEGIN2 (STATUS,USER,PASS);
     OPEND2 (DID,"BOOKING ","S");
     OPENT2 (TID,DID,"CITIES  ","S");
--
-- Set up projections for data transfer between fields and columns.
--
     PROSH2 (TID,"F","CODE     ","RW",BA,"CITYCODE ","*");
     PROSH2 (TID,"A","NAME     ","RW",BA,"CITYNAME ","*");
--
-- Set up select constraint.
-- This constraint will use the current contents of the field
-- CITYCODE and fetch only records with the column CODE equal to
-- this, when a record is fetched from the table through a
-- SET2 followed by a GET2 call.
-- When specifying the fields we have used the
-- "current picture" setting, here set by INPSH2
-- (is also set by RDFSH2).
-- A full field specification would have been "CUSTREQ: CITYCODE".
--
     SELSH2 (TID,"F","CODE     ","EQ",BA,"CITYCODE ","*");

-- The *'s used for project length in the PROSH2 and SELSH2
-- calls, give the field length.
```

```
--
-- Make a loop that:      Reads the field and checks in the table
--                        if a record with this field as key
--                        exists,
--                        if it exists writes the record and
--                        reads
--                        the fields and updates the record
--                        else
--                        clears the fields of the picture, reads
--                        and inserts the record.
     loop
          CLFSH2 ("CUSTREQ ","CITYCODE ");
          RDFSH2 ("CUSTREQ ","CITYCODE ",STATUS);
          exit when STATUS(1) /= 0 ;

--
-- See if city record exists, if so display else give error message
--
           SET2 (TID,BA);
           GET2 (TID,BA);
           if TID(1) = 0 then
--
--              A record in the table 'CITIES', having a value in the
--              column CODE that is equal to the value of the field
--              CITYCODE is found (this search is determined by the
--              SELSH2 call above).  The name of the city is returned
--              in the storage for the
--              field CITYNAME (transfer set up by PROSH2 above)
--
                WRFSH2 ("CUSTREQ ","CITYNAME ");
                CLFSH2 ("CUSTREQ ","DEPARDAT-$LAST ");
                RDFSH2 ("CUSTREQ ","DEPARDAT-$LAST ",STATUS);
           else
--
--              Write a message.  The message will be written on
--              message area one.  This is by default set to row 24,
--              column 1 and length of 40 characters.
--
                MSGSH2 (1,"No such city exists ",20);
           end if;
     end loop;
     (ENDSH2;
     (END2;
     end program;
```

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

EXAMPLE: **********

Example 3 Check routine usage example, same as example 2, but using check routine.

```
     begin
     integer TTYPE,STATUS(8);
     integer DID(4),USER(2),PASS(2);

     TTYPE:=0;
     INISH2 (TTYPE,"TICKET ");
     INPSH2 ("CUSTREQ ");
     INASH2 ("CUSTREQ ");

     CLSSH2 ("WH",0,0,0,0);
     WRPSH2 ("CUSTREQ ");
     MDRMVB (USER,0,"USER     ",0,8);
     MDRMVB (PASS,0,"PASS     ",0,8);
     BEGIN2 (STATUS,USER,PASS);
     OPEND2 (DID,"BOOKING ","S");
     OPENT2 (STATUS(5),DID,"CITIES   ","S");
     PROSH2 (STATUS(5),"F","CODE     ","RW",BA,"CITYCODE ","*");
     PROSH2 (STATUS(5),"A","NAME     ","RW",BA,"CITYNAME ","*");
     SELSH2 (STATUS(5),"F","CODE     ","EQ",BA,"CITYCODE ","*");

     loop
          CLFSH2 ("CUSTREQ ","CITYCODE ");
          RDFSH2 ("CUSTREQ ","* ",STATUS);
          exit when STATUS(1) /= 0 ;
     end loop;
     (ENDSH2;
     (END2;
     end program;

     procedure CITY (INFO,LEN,STATUS);
     integer INFO,LEN,STATUS;
----
-- Check routine to check if a city with the given citycode exists.
-- Note: here the table identifier is transported from the
--          main program using the status variable.
--          This can be done using global or common variables,
--          if your programming language supports such.
----
--
-- See if city record exists, if exist display else give error
-- message.
--
     begin
     PUTSH2 ("CITYCODE ",INFO,"C");
     SET2 (STATUS(5),BA);
     GET2 (STATUS(5),BA);
     if STATUS(5) = 0 then
          WRFSH2 ("CUSTREQ ","CITYNAME ");
     else
          MSGSH2 (1,"No such city exists ",20);
          STATUS(1):=-1;
     end if;
     end CITY;
```

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

```
*****************************************************************************

                        DEFSH2 - routine to make runtime definitions.
                        PMTSH2 - routine to prompt user for predefined answer.

                        This example uses the default exit character
                        of the editor, this character is @ and is
                        associatied with the exit function number one.
                        We also assume that the character * has been defined
                        using the editor, as exit character with exit
                        function number two.
                        I.e. the value two will be returned in status(2)
                        when an the exit character is used to exit from
                        reading the picture.

        begin
        integer TTYPE,STATUS(4);
        integer COND;
--
        TTYPE:=0;
        INISH2 (TTYPE,'TICKET ');
        INPSH2 ('CUST     ');
        INASH2 ('CUST     ');
--
--      Define a prompter with name EXIT on message area one.
--      (Messagea one is default set up at initiation to row 24, col 1 and
--       40 positions long).
--
        DEFSH2 ('PMT ','EXIT',1,/Do you want to exit from program ? /Y/N//')
--
        CLSSH2 ('WH',0,0,0,0);
        WRPSH2 ('CUST     ');
--
        loop
            CLFSH2 ('CUST     ','* ');          -- * denotes all fields
            RDFSH2 ('CUST     ','* ',STATUS);
            if STATUS(1) = 2 then
--
--              An exit function was used to exit from picture.
--
                if STATUS(2) = 1 then
--
--                      @ was entered in one field of the picture.
--                        ask user if he wants to exit from program,
--                        using the promptins routine.
--
                        PMTSH2 ('EXIT',COND);
                        exit when COND = 1;       -- Possible values for COND
                                                  -- are 1 corresponding to 'Y'
                                                  -- and 2 corresponding to 'N'
                elsif STATUS(2) = 2 then
--
--                      * was entered in one field of the picture
--
                        MSGSH2 (1,'Next customer',13);
                end if;

        end loop;
        end program;
```

```
**************************************************************************
        Example 5  exit function with exit routine
                        Extension of example 3.
                        The following routines are introduced:
                        JMPSH2 - Reposition reading to a specified field
                        GINSH2 - Get status information.

                        We assume that you have defined the exit routine
                        GOPHON along with the exitfunction definition
                        for the exit character #.


        procedure GOPHON (CODE,STATUS);
--
--      This exit routine is used to position cursor to read the telephone
--      field when the exit character # is typed in at the terminal.
--
        integer FIELD(4);       -- define program storage for a character
                                -- string of eisth characters length.
        integer LEN,STAT,COND;
--
        begin
--
--      First check that the exit character was not given in the first
--      field of the picture.
--
        GINSH2 ('CF',FIELD,LEN,STAT);
        MDRCLB ('NAME    ',0,FIELD,0,8,COND); -- String comparation routine.
--
        if COND = 0 then
--
--              The exit function routine was entered from the first field
--              of the CUST picture, the field named NAME.
--              For this field we do not want the repositionings to be valid
--              set return status to reread field.
--
                STATUS(1):=-1;
        else
--
--              Set status to reposition reading to the phone number field
--              names PHONO, using the routine JMPSH2.
--
                JMPSH2 ('PHONO ',STATUS(2));
                STATUS(1):=1;                   -- Tell screenhandler to reposition
        end if;
        end GOPHON;


        Now typing a '#' character in any field, except the first,
        of the picture CUST, will result in a 'jump' of the cursor to read
        the field PHONO.
```

In Example 3 above, the check routine puts the information read from the screen (in the INFO variable) into the storage of the field CITYCODE, using PUTSH2.

This is done since the select constraint is set against the contents of the field CITYCODE but the information read from the screen is not yet stored into the field (done at exit from the check routine). Here you can use the pseudo field name ($INFO) for the field buffer INFO that is input to the check routine.

The selection will then be made as (replaces the select call above):

```
SELSH2 (STATUS(5),"F","CODE        ","EQ",BA,"$INFO ",3);
```

Now the select is done against the contents of the variable INFO that is input to all check routines, and retrieval can be made without moving the information into the field storage.

The statement PUTSH2 ("CITYNAME ",INFO,"C"); in the example is then no longer needed. Note that the select statement must contain a length specification if you use $INFO in this way.