

# **FMS-11/RSX**

## **Installation Guide/Release Notes**

Order No. AA-HU91A-TC

**June 1986**

This document describes how to use the Form Management System (FMS-11) on RSX-11M and RSX-11M-PLUS systems. It provides the information required to design forms for display on the VT200 video terminal and to develop BASIC-PLUS-2, COBOL-11, COBOL-81, DIBOL-83, FORTRAN IV, FORTRAN-77, and MACRO-11 programs that use FMS-11 forms for gathering and displaying data.

<b>SUPERSESSION/UPDATE INFORMATION:</b>	This is a revised manual containing information that pertains to the latest release of FMS-11/RSX, V2.0
<b>OPERATING SYSTEM AND VERSION:</b>	RSX-11M V4.2 RSX-11M-PLUS V3.0 RSX-11S V4.2

**digital equipment corporation • maynard, massachusetts**

Revised, January 1986  
First Printing, May 1981

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.


No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

Copyright © 1985 by Digital Equipment Corporation. All Rights Reserved.

The postage-paid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The manuscript for this book was created using generic coding and, via a translation program, was automatically typeset and paginated. Book production was done by Educational Services Development and Publishing in Bedford, MA.

The following are trademarks of Digital Equipment Corporation:

 ™	DECtalk	Professional
ALL-IN-1	DECUS	RAINBOW
CTS-300	DECwriter	RSTS
DATATRIEVE	DIBOL	RSX
DEC	DSM-11	TOPS-10
DECdirect	FMS-11	TOPS-20
DECmail	KI	UNIBUS
DECmat	KL10	VAX
DECmate	MASSBUS	VAXcluster
DECnet	MicroPower/Pascal	VAXstation VMS
DECservice	MicroVAX II	VT
DECsystem-10	PDP	Work Processor
DECSYSTEM-20	P/OS	

# Contents

## Chapter 1 Introduction

1.1	Using FMS-11/RSX.....	1-1
1.2	The FMS-11/RSX Documentation Set.....	1-1

## Chapter 2 Installing FMS-11/RSX

2.1	Installation Procedure.....	2-2
2.2	Verifying Installation Procedures.....	2-4
2.3	Sample Installation .....	2-9

## Chapter 3 User Environment Test Package (UETP)

3.1	Running the UETP .....	3-1
3.2	UETP Step-by-Step Procedure .....	3-2

## Chapter 4 VT52 Support

## Chapter 5 Form Driver Resident Library

## Chapter 6 Release Notes

6.1	VT200 Terminal Support.....	6-1
6.2	Multiple Forms .....	6-1
6.3	I+D Space.....	6-1
6.4	Cluster Library Support .....	6-1
6.5	European Decimal Syntax Support.....	6-2
6.6	VT200 Variations .....	6-2
6.7	High-Level Language Changes.....	6-2
6.8	Demo Application Programs .....	6-2
6.9	New Call Parameters .....	6-3
6.10	Building COBOL Task Images .....	6-3
6.11	COBOL Interface Information for COBOL Users .....	6-4

## Figure

2-1.	Program Naming Conventions .....	2-5
------	----------------------------------	-----

## Tables

2-1.	Dependencies.....	2-3
2-2.	Demo Program Configurations .....	2-7



---

# Chapter 1

## Introduction

This document serves two purposes. As an installation guide, it describes the installation verification procedures for FMS-11/RSX software and the FMS-11/RSX User Environment Test Package (UETP); it also provides information about using FMS-11 on a VT200 terminal and information about the Form Driver resident library. As release notes, it provides general information about the changes to FMS-11/RSX included in this release.

### 1.1 Using FMS-11/RSX

The following steps summarize use of your FMS-11/RSX software:

1. Follow the installation procedures in Section 2 of this document.
2. Verify that your hardware and software are working properly by running the demonstration programs provided and by completing the UETP procedure described in Section 3 of this document.
3. Read the *FMS-11/RSX Software Reference Manual* to learn to use the FMS-11/RSX software components.
4. Use the extended examples referenced in the *FMS-11/RSX Software Reference Manual* to become acquainted with the software.
5. Write your own simple FMS-11 application.

### 1.2 The FMS-11/RSX Documentation Set

The three documents for the FMS-11/RSX software and their order numbers are:

1. *FMS-11/RSX Software Reference Manual* AA-H855B-TC
2. *FMS-11/RSX Mini-Reference* AV-H856B-TC
3. *FMS-11/RSX Installation Guide/Release Notes* AA-H857B-TC



---

## Chapter 2

# Installing FMS-11/RSX

The procedure for installing FMS-11 on an RSX-11M or an RSX-11M-PLUS system consists of copying the files from the distribution media to the device where they will be used and building the FMS-11 components. The entire process is controlled by an indirect command file.

When the command files finish executing, the FMS utilities can be copied to the system account and installed with the MCR INSTALL command.

You must be logged in under a privileged account for successful completion of the installation. If you are installing FMS-11 on an RSX-11M-PLUS system, your terminal must be set for MCR mode commands.

Before you install FMS-11, be sure the following system tasks are available:

System Task	Installed Name
Task Builder	... TKB
PIP	... PIP
Install	... INS
UFD	... UFD
FLX	... FLX
Mount	... MOU
Indirect	... AT

It is assumed that FMS-11 is being installed on the system device. Before you begin the installation procedure for FMS-11 V2.0, make sure UIC [30,10] is empty, and then set your default to UIC [30,10]. All FMS-11 files are moved into account [30,10]. To run the Form Editor, the demonstration programs, and the UETP (built as part of the installation procedure), you must have a VT200 or VT100 terminal.

Approximately 6800 disk blocks are required to install FMS-11/RSX if all the demonstration programs and Form Driver Resident Libraries have been built. The demonstration program in a particular language is built only if the corresponding language processor is installed in the system.

In the installation procedure detailed below, the parameter 'ddn' is the name and unit number of the device on which FMS-11 is being installed. The parameter 'dev' is the name and unit number of the device on which the distribution medium is mounted.

## 2.1 Installation Procedure

```
MCR> ASN ddn:=SY:           ! Make the installation device
                             ! the default system device
MCR> UFD ddn:[30,10]        ! Create the UFD for FMS-11
                             ! if it doesn't already exist
MCR> SET /UIC=[30,10]       ! Set the default UIC
MCR> PIP [30,10]*.*;*/DE    ! Make sure UFD is empty
                             ! before installation begins
```

### For magnetic tapes on RSX-11M systems:

```
MCR> ALL dev:               ! Allocate the drive
MCR> FLX SY:=dev:FMSINS.CMD/DO ! Copy installation command file
```

### For magnetic tapes on RSX-11M-PLUS systems:

```
MCR> MDU dev:/FOR          ! Mount tape as foreign device
MCR> FLX SY:=dev:FMSINS.CMD/DO ! Copy installation command file
```

### For disks on both systems:

```
MCR> MDU dev:FMSRSX        ! Mount the disk
MCR> PIP SY:=/NV=dev:FMSINS.CMD! Copy installation command file
```

### For all media:

```
MCR> @FMSINS               ! Copy files from distribution
                             ! media and build utilities
                             ! and demo programs
```

The installation procedure will prompt for information concerning the building of FMS Demo and User Test Program (UETP) applications. If you want to build these demo programs, the appropriate compilers and supporting files must be installed on your system. Table 2-1 defines the files required for each high-level language demo.



**Table 2-1. Dependencies**

Language	Installed Compiler Name	Dependent File	Dependent File
MACRO-11	... MAC	LB:[1,1]SYSLIB.OLB	
FORTTRAN IV	... FOR	LB:[1,1]FOROTS.OLB	
FORTTRAN-77	... F77	LB:[1,1]F77FCSOTS.OLB	LB:[1,1]F77RMSOTS.OLB
COBOL-11	... CBL	LB:[1,1]COBLIB.OLB	
COBOL-81	... C81	LB:[1,1]C81LIB.OLB	
BASIC-PLUS-2	... BP2	LB:[1,1]BP2OTS.OLB	
DIBOL-83	... DIB	LB:[1,1]DBLUESL.OLB	LB:[1,1]DBLOSSL.OLB

The installation procedure will run through a series of queries. In response to these queries, you indicate what demo applications you want built. For example, the procedure will query:

Build FORTRAN IV demo programs [Y,N]?

You then enter 'Y' for yes or 'N' for no. If the proper compiler and supporting files exist on the system, the installation procedure will go ahead and build this demo program. If the proper compiler or supporting files do not exist on the system, the installation procedure will explain what files are missing. For example:

```
FORTTRAN IV compiler (...FOR) not available.  
FORTTRAN IV demos won't be built.
```

Whether or not the demo programs are built, the installation procedure will continue to ask questions until it has covered all available high-level languages. If you answer with a 'Y,' it will attempt to build the demo applications unless the compiler or supporting files are not on the system. After the query phase is complete, all MACRO demo and UETP programs, the Form Editor (FED), and the Form Utility (FUT) will be task built automatically.

FMS-11 is distributed with two prebuilt Form Driver libraries: an object library using FCS (FDVLIB.OLB), and an object library using RMS (FDVLRM.OLB). The distributed versions of the Form Driver provide support for the VT200 and VT100 only and require the full-duplex terminal driver. Support for all features other than debug mode is included. If you wish to reconfigure the Form Driver to tailor it to specific application requirements, execute the command file FDVBOLD.COMD.

FMS-11 is distributed with a command file that controls the building procedure for FMS resident libraries. The command file FDVRESBLD.CMD performs the following functions:

- Prompts for different configurations for the resident library that will be built.
- Builds the resident library.
- Copies the resident library to LB:[1,1].
- Installs the resident library in memory.
- After the resident library has been built and installed, queries the user as to whether or not the application demonstration programs should be built and linked against the newly created resident library.

The system on which FED and FUT are run must include checkpointing to a system checkpoint file and the full-duplex terminal driver.

## **2.2 Verifying Installation Procedures**

The installation procedure is almost self-verifying. As part of the installation, demonstration programs are compiled and task built if the appropriate language processors are installed in the system. The demonstration programs listed in Table 2-2 all use the same application in MACRO-11, FORTRAN IV, FORTRAN-77, BASIC-PLUS-2, COBOL-11, COBOL-81, and DIBOL-83.

Figure 2-1 illustrates the demonstration program naming conventions. Table 2-2 illustrates what tasks are built by the FMS-11 installation procedure and how they are built.

Demo program name ----> AAABBBCCC.tsk

where:

- AAA = First 3 letters of the source compiler used
  - = MAC -- MACRO-11
  - = BAS -- BASIC-PLUS-2
  - = C11 -- COBOL-11
  - = C81 -- COBOL-81
  - = FOR -- FORTRAN IV
  - = F77 -- FORTRAN-77
  - = DBL -- RSX-11/DIBOL-83
- BBB = File system used for this demo
  - = FCS -- File Control System
  - = RMS -- Record Management System
- CCC = Configuration in which the demo was built
  - = DEM -- All language runtime and file control libraries are linked within the task
  - = RES -- Built with the file system resident library
  - = CLS -- Built with the file system resident library and the FMS-11 Form Driver resident library clustered together

**Figure 2-1. Program Naming Conventions**

**Table 2-2. Demo Program Configurations**

Task Name	FCS Support	RMS Support	RMS Resident Library Support	FMS Resident Library	FMS/RSX Resident Library Clustered Support	Task Disk Block Allocation
Demo Program Configurations						
FORFCSDEM.TSK	X					75
F77FCSDEM.TSK	X					78
MACFCSDEM.TSK	X					51
BASRMSDEM.TSK		X				213
C11RMSDEM.TSK		X				104
C81RMSDEM.TSK		X				100
F77RMSDEM.TSK		X				103
BASRMSRES.TSK			X			84
C11RMSRES.TSK			X			77
C81RMSRES.TSK			X			70
DBLRMSRES.TSK			X			42
F77RMSRES.TSK			X			70
BASRMSCLS.TSK					X	66
C11RMSCLS.TSK					X	59
C81RMSCLS.TSK					X	53
DBLRMSCLS.TSK					X	22
F77RMSCLS.TSK					X	52
Available FMS-11 Resident Libraries and Utilities						
FDVFCS.TSK	X			X		32
FDVRES.TSK				X		22
FDVRMS.TSK				X		22
FED.TSK	X					106
FUT.TSK	X					67
UETP Programs						
INITF.TSK	X					20
SIS.TSK	X					64
SISFOR.TSK	X					118
SISF77.TSK	X					118

To verify that these programs operate correctly, you should run them after the installation is complete. Be aware that if the corresponding language compilers are not installed on the system, you will encounter errors during the installation procedure. For example, to run the MACRO version of the demonstration program, type the following:

```
MCR> SET /UIC=[30,10]  
MCR> RUN MACFCSDEM
```

Note that any time you run one of the demonstration programs distributed with FMS-11, you must set the default to the account containing the form library and data files required by the program.

The demonstration programs implement a menu-driven application that allows you to enter customer information, parts descriptions, and employee information. The initial menu form (called "FIRST") allows you to select one of the three functions or to exit. The second menu form (called "LAST") allows you to repeat the function chosen from FIRST, to return to FIRST, or to exit. These menus are displayed simultaneously throughout the demonstration program. Each time you initiate a function, a file is created and the data collected is written sequentially into the file. The names of the files correspond to the functions: NEWCUS.DAT for customer information, PARTS.DAT for parts descriptions, and EMPLOY.DAT for employee information. You can inspect these files with an editor to verify the data.

To verify that the Form Utility (FUT) was built successfully, use it to get a directory of one of the form libraries distributed with FMS-11 and a listing of a form description. The following sample session with FUT gives a directory of the form library DEMLIB.FLB and a listing of the form PARTS contained in that library.

```
MCR> SET /UIC=[30,10]
MCR> RUN FUT
FUT> DEMLIB.FLB/LI
```

```
FUT V02.00
4-DEC-79
```

```
Library DL0:[30,10]DEMLIB.FLB;1 created: 5-AUG-85
Directory is 1 block long.
```

Form	Date	Impure area (bytes)
FIRST	5-AUG-85	399
CUSTPR	5-AUG-85	344
LAST	5-AUG-85	275
EMPLOY	5-AUG-85	754
PARTS	5-AUG-85	794
CUSTO	5-AUG-85	612
CLEARF	5-AUG-85	220

```
FUT>PARTS=DEMLIB.FLB/FD
```

```
DL0:[30,10]DEMLIB.FLB Form name? PARTS
```

```
DL0:A[30,10]DEMLIB.FLB Form name? CR
```

```
FUT>
```

The file [30,10]PARTS.FMD contains the form description for the form PARTS. The file can be spooled to a line printer to obtain a hardcopy listing.

To verify that the Form Editor (FED) was built correctly, use the example provided in Section 2.7 of the *FMS-11/RSX Software Reference Manual* to create a new form and modify an existing form.

## 2.3 Sample Installation

```
>;
>; FMSINS.CMD
>;
>;
>; Copyright (c) 1985 by
>; Digital Equipment Corporation, Maynard, MA
>;
>; Install FMS/RSX
>;
>SET /MCR-TI:
>;
>* Enter distribution device and unit number (S R:2-3): mm0
>;
>;
>* Is the distribution tape 1600 BPI? [Y/N]: y
>FLX SY:/RS=mm0:[30,10]*./DO/DNS:1600.
>FLX SY:/RS=mm0:[30,20]*./DO/IM/DNS:1600.
>DMO mm0:
DMO -- TT30: dismounted from MM0: *** Final dismount initiated ***
>;
>;
>; Build FMS Utilities and Demo Programs
>;
>;
>; FMSBLD.CMD
>;
>; Copyright (c) 1985 by
>; Digital Equipment Corporation, Maynard, MA
>;
>;
>* Build FORTRAN IV UETP and demo programs ? [Y/N]: y
>;
>* Build FORTRAN-77 UETP and demo programs ? [Y/N]: y
>;
>* Build COBOL-11 demo programs ? [Y/N]: y
>;
>* Build COBOL-81 demo programs ? [Y/N]: y
INS -- File not found
>;
>; COBOL-81 compiler (...CB1) not available.
>; COBOL-81 demos won't be built
>;
>;
>* Build BASIC-PLUS-II demo programs ? [Y/N]: y
>;
>* Build DIBOL-83 demo programs ? [Y/N]: y
CBD -- Common not in system
>;
>* Build RMS demos with RMSRES resident library ? [Y/N]: y
>;
>; Build the FMS utilities (FED and FUT)
>;
>TKB @FEDTKB
>TKB @FUTTKB
>;
>;
>; Build the FMS demonstration programs in FORTRAN,
>; FORTRAN-77, COBOL, BASIC-PLUS-II, and MACRO.
>; DEMBLD.CMD
>;
>; Copyright (c) 1985 Digital Equipment Corporation, Maynard, MA
>;
>; Build the demonstration programs
>;
>;
>;
>;
```

```

>;
>;
>;
>;
>;
>;          Build FORTRAN demo for FORTRAN IV
>;
>FOR @FORDEM
.MAIN.
MENUCH
STATCK
CHFMST
SCOPY
SCOMP
>TKB @FORFCSTKB
>;
>;          Task name is FORFCSDEM.TSK
>;
>;
>;          Build FORTRAN demo for FORTRAN-77
>;
>F77 @F77DEM
>TKB @F77FCSTKB
>;
>;          Task name is F77FCSDEM.TSK
>;
>TKB @F77RMSRES
>;
>;          Task name is F77RMSRES.TSK
>;
>;
>;          Build COBOL-11 demo
>;
>;          An error is expected at taskbuild time. The
>;          diagnostic message:
>;
>;          Module FDVDAT multiply defines symbol T$LUN
>;
>;          is the result of the redefinition of the taskbuild
>;          command file.
>;
>CBL @C11DEM
>TKB @C11RMSRES
TKB -- *DIAG*-Module FDVDAT multiply defines symbol T$LUN
>;
>;          Task name is C11RMSRES.TSK
>;
>;
>;          Task build BASIC+2 demo
>;
>BP2 @BASDEM
PDP-11 BASIC-PLUS-TWO V2.3-00
BASIC2
@BASDEM
OLD BASDEM.B25
COMPILE/LIST/OBJECT
BASDEM 11:32 AM 25-Oct-85
EXIT
>TKB @BASRMSRES
>;
>;          Task name is BASRMSRES.TSK
>;
>;
>;          Build DIBOL-83 demo
>;
>DIB DBLDEM,DBLDEM=DBLDEM

```



```

No errors detected
>TKB @DBLRMSRES
>;
>; Task name is DBLRMSRES.TSK
>;
>;
>; Build MACRO demo
>;
>MAC @MACDEM
>TKB @MACFCSTKB
>;
>; Task name is MACFCSDM.TSK
>;
>; Demo programs are built
>;
>;
>; Build the UETP
>;
>;
>; UTPBLD.CMD
>;
>; Copyright (C) 1985 Digital Equipment Corporation, Maynard, MA
>;
>; Build the FMS UETP
>;
>SET /MCR=TI:
>;
>;
>;
>; Assemble MACRO version of the file initialization program
>;
>MAC INITF,INITF/CR/-SP=INITF
>;
>; Task build file initialization program
>;
>TKB @INITKB
>;
>; Assemble script modules
>;
>; Assemble FDV terminal I/O module with script hook
>; included by SCRSYM.MAC
>;
>MAC UTPTIO,UTPTIO/CR/-SP=FMSMAC/ML,SCRSYM,FDVTIO
>;
>; Assemble UETP Script Processor
>;
>MAC SCRIPT,SCRIPT/CR/-SP=FMSFAC/ML,SCRIPT
>;
>; Assemble MACRO version of UETP
>;
>MAC SIS,SIS/CR/-SP=FMSMAC/ML,SIS
>;
>; Task build MACRO version of UETP
>;
>TKB @UTPTKB
>;
>;
>; Build FORTRAN IV version of UETP
>;
>FOR SISFOR,SISFOR/-SP=SIS
.MAIN.
SIRECV
SISHIP
SICBK
SICOR
CACCNT
CINV
CDESCR
TRACE

```

## 2-12 Installing FMS-11/RSX

```

>TKB @BASRMSCLS
>;
>;          Task name is BASRMSCLS
>;
>;
>;          Build DIBOL-83 demo with RMSRES and FDVRMS and DBLRSX
>;          clustered
>;
>TKB @DBLRMSCLS
>;
>;          Task name is DBLRMSCLS.TSK
>;
>;
>;          FDV RESIDENT LIBRARY GENERATOR
>;
>;          The options are :
>;
>;          1) Build FDV resident library with RMS-11 support
>;          2) Build FDV resident library with FCS CODE
>;          3) Build FDV resident library with FCS support
>;          4) None (exit)
>;
>;
>*          Which resident library would you like to build [0]: 4
>;
>;
>;
>;
>;          Installation complete
>;
>@ <EOF>

```



---

## Chapter 3

# User Environment Test Package (UETP)

The User Environment Test Package (UETP) for FMS-11/RSX is designed as an integrity test of the Form Management System. The intent is to verify that a typical FMS-11 application can be built and executed with the installed software. The UETP application is a Simple Inventory System (SIS). It will be referred to throughout the remainder of this document as SIS.

The UETP is built as part of the installation procedure. The MACRO version (SIS.TSK) is built automatically. The FORTRAN IV (SISFOR.TSK) and FORTRAN-77 (SISF77.TSK) versions are built only if the corresponding compilers are installed in the system. The UETP requires FORTRAN IV-PLUS with FCS support for file I/O. Therefore, a FORTRAN-77 version of the UETP is built only if the FORTRAN-77 FCS OTS is on the system (LB:[1,1]F77FCSOTS.OLB).

The UETP is built to include a script processor for automatic execution. The script provided is designed to exercise a majority of the capabilities of the Form Driver component of FMS-11.

### 3.1 Running the UETP

To run the UETP, set the default UIC to [30,10]. Initialize the necessary application files by running the initialization program as follows:

```
MCR> RUN INITF
```

You can then run either the MACRO or one of the FORTRAN versions of the UETP. For example, to run the FORTRAN IV version, type:

```
MCR> RUN SISFOR
```

Since the program executes under script control, character input is taken from the script file rather than the terminal keyboard. Script characters are supplied every half second to simulate human typing. Some delays have been included in the script to facilitate reading. The progress of the script can be controlled from the keyboard.

Type the letter 'S' to halt the script temporarily; type 'G' to resume the script. After halting the script, you can step through it one character at a time by pressing the SPACE key. Since escape sequences are several characters long, it is necessary in some cases to step through more than one character before the Form Driver takes any action. You can abort the script at any time by typing 'Z' (not CTRL/Z). After the script is aborted, all subsequent input is taken from the keyboard rather than the file. If you abort in the middle of an escape sequence, the Form Driver might signal an error (by sounding the bell) in response to the first few characters entered manually, but this should not persist.

## 3.2 UETP Step-by-Step Procedure

In the following description of the execution of the script, each step corresponds to a single form displayed by the application. Within each step, there is a description of the input to be entered and the resulting actions.

### Step 1 – SIS Introduction

1. The script types the **HELP** key (the PF2 key on the keypad) and there is a one line explanation of how to respond to the prompt NEXT? displayed on the last line of the screen.
2. The script types the **HELP** key again, and FDV displays the HELP form associated with the introduction form.

### Step 2 – Introduction HELP Form

The script types the RETURN (**RET**) key, and FDV displays the introduction form again. The cursor is at the NEXT? prompt.

### Step 3 – SIS Introduction

The script accepts the default response (4) to request a correction form. SIS processes the response, and the correction form is displayed.

### Step 4 – Correction Request

The script overrides the default response (N) and types D to request the form for changing inventory descriptions. SIS processes the response and displays the requested form.

### **Step 5 – Change Description**

1. The script types a stock number, advances to the next input field (“Description”) by typing the **(TAB)** key, and types the name of the item.
2. The script provides several new descriptions, advancing from field to field by using the **(TAB)** key, and types **(RET)** when the form is complete.
3. SIS processes the entries and displays the next form, a menu of SIS processes.

### **Step 6 – Menu**

The script overrides the original default response (0) and types 4 to request the correction form again.

### **Step 7 – Correction Request**

The script types I to request the form for changing inventory quantities, and the form is displayed.

### **Step 8 – Change Inventory Quantity**

1. The script types a stock number and the **(TAB)** key. SIS displays the corresponding description.
2. SIS advances the cursor to the “Quantity in Stock” field.
3. The script types a quantity and the **(TAB)** key. The initial quantity for each item is 100.
4. SIS advances the cursor to the next line of the form.
5. The script types several stock numbers and quantities and then types **(RET)**. SIS processes the entries, and the menu form is displayed.

### **Step 9 – Menu**

SIS has been designed to modify the default response to the menu form. The original default response is 0. After any other response is typed, SIS changes the default response to the last response typed. Therefore, the default is 4 at this point because the script typed 4 in Step 6.

The script accepts the default by typing only the **(RET)** key, and the correction form is displayed.

### **Step 10 – Correction Request**

The script types A to request the form for changing account information, and the form is displayed.

### **Step 11 – Change Account**

1. The script completes each field and types the **(TAB)** key to advance to the next field.
2. When the information for the first account is complete, the script accepts the default response (Y) for the prompt MORE ACCOUNTS? by typing **(RET)**.
3. SIS processes the new account information, displays a blank version of the same form, and places the cursor at the first field in the form.
4. When the information for the second account is complete, the script overrides the default response to the prompt MORE ACCOUNTS? by typing N and RETURN.
5. SIS processes the new account information, and the menu form is displayed.

### **Step 12 – Menu**

The script types 3 to use the inventory status request form, and the form is displayed.

### **Step 13 – Status**

1. The script enters several stock numbers and types **(TAB)** after each one.
2. SIS displays the corresponding descriptions and quantities in stock. At this point in the script, there are 100 pieces in stock for each item.
3. After checking the inventory status of each item, the script types **(RET)** to indicate that work with the form has been finished.
4. The menu form is displayed.

The “Stock Number,” “Description,” and “Quantity in Stock” fields in the inventory status request form are indexed fields. Each data line in the form contains the same fields. Within a line, SIS references the fields by name. Within the form as a whole, SIS references lines by indexes, with the first data line having the index 1, the second line the index 2, and so on.

### **Step 14 – Menu**

The script types 1 to use the form for recording incoming shipments, and the form is displayed.

### **Step 15 – Receiving Form**

1. The script types the supplier’s account number and **(TAB)**. SIS displays account information from the account file and locates the cursor at the Invoice Number field.
2. The script types an invoice number and **(TAB)**. SIS locates the cursor at the first “STOCK #” field.



3. For each item received, the script types the stock number and **TAB**.
4. SIS displays the corresponding description and places the cursor in the "QUANTITY" field.
5. The script types the quantity received and **TAB**. SIS places the cursor at the "STOCK #" field in the next line of the form.
6. On the fourth line, the script types the stock number 500 by mistake. However, the typing error is not detected until **TAB** is typed.
7. With the cursor at the "QUANTITY" field, the script types the BACKSPACE key to move the cursor back to the "STOCK #" field.
8. The script uses the DELETE key to erase the incorrect stock number.
9. When the script types the correct stock number (501) and **TAB**, SIS corrects the description and relocates the cursor at the "QUANTITY" field.
10. After recording all items received, the script types the **RET** key. SIS processes the entries and updates the data base, and the menu form is displayed.

#### **Step 16 – Menu**

The script types 3 to request the inventory status request form.

#### **Step 17 – Status**

1. The script checks the new inventory status of several items by typing their stock numbers and the **TAB** key.
2. After each stock number is complete, SIS displays the description of the item and the current inventory quantity. The quantities reflect the incoming shipments recorded in Step 15.
3. The script types **RET** to indicate that work with the status form is complete, and the menu form is displayed.

#### **Step 18 – Menu**

The script types 2 to request the form for outgoing shipments, and the form is displayed.

#### **Step 19 – Shipping Form**

1. The script types the customer's account number and **TAB**. SIS displays account information from the account file and locates the cursor at the "PICKING LIST #" field.
2. The script types a picking list number and **TAB**. SIS locates the cursor at the first stock number field.

3. For each of several items in the shipment, the script types a stock number and **(TAB)**. SIS supplies the description and locates the cursor at the "QUANTITY" field.
4. The script types the quantity shipped and **(TAB)**, and SIS locates the cursor at the next "STOCK #" field.
5. After recording all the items in the shipment, the script types **(RET)**. SIS processes the entries and updates the data base, and the menu form is displayed.

#### **Step 20 – Menu**

The script types 3 to request the inventory status request form.

#### **Step 21 – Status**

The script requests the inventory status for several items. The quantities reflect the outgoing shipment recorded in Step 19. After the script types **(RET)**, the menu form is displayed.

#### **Step 22 – Menu**

The script types 4 to request the correction form, and the form is displayed.

#### **Step 23 – Correction Request**

The script types D to request the form for changing inventory descriptions, and the form is displayed.

#### **Step 24 – Change Description**

To prepare for an incoming shipment of a new inventory item, the script types the item's new stock number and description. The script then types **(RET)**, and the menu form is displayed.

#### **Step 25 – Menu**

The script types 1 to use the form for recording incoming shipments, and the form is displayed.

#### **Step 26 – Receiving Form**

The script types the stock number and quantity of the incoming shipment. SIS displays the description of the item. The script types **(RET)**, and the menu form is displayed.

#### **Step 27 – Menu**

The script types 3 to request the inventory status request form, and the form is displayed.

### Step 28 – Status

The script types the stock numbers for several items in the inventory, including the new item recorded in Steps 24 and 26. SIS displays descriptions and current inventory quantities. The script types **(RET)**, and the menu form is displayed.

### Step 29 – Menu

The script types 1 to use the form for recording incoming shipments, and the form is displayed.

### Step 30 – Receiving Form

1. The script types the account number 1 by mistake, but the error is not detected until after **(TAB)** is typed and SIS displays the information for account number 1.
2. With the cursor at the “INVOICE NUMBER” field, the script types the BACKSPACE key to move the cursor back to the “ACCOUNT NUMBER” field.
3. The script uses the <DELETE> key to erase the account number.
4. The script types the correct account number (500), and SIS displays the account information for that account. SIS has not processed the incorrect data because the error was corrected before typing the **(RET)** key. Therefore, trace records for this transaction will show only the correct account number (500).
5. The script types an invoice number and **(TAB)**.
6. To demonstrate how a scrolled area within a form is displayed, the script fills all of the displayed blank lines in the form.
7. When the script types **(TAB)** after the quantity received in the last line, the entire field of stock information scrolls upward. The top line scrolls out of sight, and a new blank line scrolls into view at the bottom.
8. The script continues to demonstrate scrolling by typing the BACKSPACE key to move the cursor toward the top line of stock information. Lines that were not displayed scroll downward into view, and the bottom lines scroll out of sight.

This step also demonstrates a special programming technique. SIS has been designed to process the BACKSPACE function in this context by locating the cursor at the preceding “STOCK #” field in all cases. In this way, the cursor is prevented from moving backward from a “STOCK #” field to the preceding “QUANTITY” field.

9. When the demonstration of scrolling is finished, the script types **(RET)**, and the menu form is displayed.

### Step 31 – Menu

The last step for the automatic script is to type 6 and **RET**. The menu form does not show that 6 is a valid choice, but SIS can nevertheless process the response. This technique might be useful in another application designed with special features for use by a site supervisor. The supervisor can use the features without requiring site operators to learn about them.

SIS displays a special menu form that signals the end of the automatic script. At this point you can experiment with the UETP application by requesting the forms you want and by typing your own information in their fields.

To stop the UETP application, type **RET** until the menu form is displayed, then type 5 and **RET**. When the UETP application stops, the monitor prompt is displayed.

---

## Chapter 4

### VT52 Support

FMS-11 was designed to take advantage of the features of the VT100 and VT200 terminals. The Form Editor requires at least a VT100 terminal (it will not operate on a VT52). The Form Driver, however, can be built to support either the VT200, the VT100, or the VT52. Support for all these terminals cannot exist in a single Form Driver library; separate libraries are required. The Form Driver libraries distributed with FMS-11 kits (FDVLIB.OLB for FCS and FDVLRM.OLB for RMS) provide VT200 and VT100 support. If you wish to create a Form Driver with VT52 support, you must go through the Form Driver configuration procedure, which you invoke by executing the command file FDVBLD.CMD. You will be prompted as to which Form Driver options are desired, including VT52 support. If you select VT52 support, the Form Driver libraries F52LIB.OLB (for FCS) or F52LRM.OLB (for RMS) are created. These libraries should be included in the task-build for form applications exactly as the VT200 versions of the libraries are.

The VT52 version of the Form Driver provides essentially the same capabilities as the VT100 version. The differences are as follows:

1. VT100 video attributes

The VT52 does not provide support for reverse, bold, blinking, underscore, 132-column mode, or reverse screen background. The VT52 Form Driver ignores these attributes when specified for a form.

2. Scrolling

The VT52 does not support split screen scrolling. However, the VT52 Form Driver will emulate scrolling in software by redisplaying all fields in a scrolled area when the area is scrolled.

3. Function keys

Because of differences in the VT200, VT100, and VT52 keypads, the positions of the following function keys will vary as follows:

Function	VT200	VT100	VT52
Insert/Overstrike	PF1	PF1	CTRL/A
Exit scrolled area backward	PF3	PF3	Blue key
Exit scrolled area forward	PF4	PF4	Gray key

The HELP key is in the same position on both terminals: the PF2 key on the VT100, and the Red key on the VT52. All other functions are the same.

After creating Form Driver libraries with VT52 support, you can build the FMS-11 demonstration programs to run on the VT52 by task-building with the correct library. However, the script provided for the UETP is for the VT200/VT100 version of the Form Driver and will not work with the VT52 version.

---

## **Chapter 5**

# **Form Driver Resident Library**

The Form Driver with FCS support can be built as a resident library. This release will also provide resident and cluster support for RMS.

Task-build command files are distributed to build both the VT100 and VT52 versions of the Form Driver as resident libraries (FDVRES.CMD and F52RES.CMD). Before a Form Driver resident library can be built, the Form Driver configuration procedure must be executed to create the files required for the resident library.





---

## Chapter 6

### Release Notes

The following features and additions have been included in this version of FMS-11/RSX. For detailed information on use and operation of these new features, consult the *FMS-11/RSX Software Reference Manual*.

#### 6.1 VT200 Terminal Support

FMS-11/RSX V2.0 incorporates full support for the VT200 terminal, including 8-bit communications, multinational character set, and LK201 support (most function keys on the LK201 now return unique terminator values).

#### 6.2 Multiple Forms

With FMS-11/RSX V2.0, you can display multiple forms on one screen and you can control multiple impure areas.

#### 6.3 I+D Space

With this version of FMS-11/RSX, you can build applications that make use of D-space APRs. For more information about I+D separation, see the *RSX-11M/M-PLUS Task Builder Manual*.

#### 6.4 Cluster Library Support

This release of FMS-11/RSX supplies the command files for cluster library support.

## 6.5 European Decimal Syntax Support

FMS-11/RSX V2.0 provides European decimal syntax support for the Form Driver.

## 6.6 VT200 Variations

If you are using a VT200 terminal, the F12 key is equivalent to the BACKSPACE key on the VT100; the F13 key is equivalent to the LINEFEED key on the VT100; and the HELP key is equivalent to the PF2 key on the VT100.

## 6.7 High-Level Language Changes

The high-level language definition file is no longer kept in the Form Driver (FDV) libraries as it was in previous releases. Existing command procedures should be modified to include [30,10]HLLDFN.OBJ for FORTRAN, COBOL, and BASIC applications, and [30,10]DBLDFN.OBJ for DIBOL-83 applications.

DIBOL-83 users should remove the following modules from LB:[1,2]DBLOSSL.OLB: HLLDBL and DBLDFN. This can be done by using the following commands:

```
LBR LB:[1,1]DBLOSSL = HLLDBL/DE
```

```
LBR LB:[1,1]DBLOSSL = DBLDFN/DE
```

The DIBOL-83 HLL interface contains the routine FATTCH. To attach your terminal, call this routine after you perform the FINIT call, e.g., CALL FATTCH().

## 6.8 Demo Application Programs

Demo application programs have the following characteristics:

- All demo applications display multiple forms simultaneously.
- Each displayed form contains its own name within the border of that form.

## 6.9 New Call Parameters

1. FCHIMP is a new call, added for this release.

For the parameter list for the FCHIMP call, refer to the *FMS-11/RSX V2.0 Software Reference Manual*.

2. FIDATA is an old call, available in previous releases.

As of this release, the FIDATA function accepts the optional "fval" parameter.

## 6.10 Building COBOL Task Images

COBOL application default LUN assignments conflict with those of the FMS-11/RSX Form Driver. Therefore, it is necessary to reserve those LUNs used by the Form Driver during link time to avoid conflicting LUN allocations at run time. This is accomplished using the Task Builder option GLBPAT.

COBOL uses a byte table to keep track of free and allocated LUNs.

Global symbol \$RESCH points to the third byte of this table. The first two bytes are reserved for use by COBOL. A zero byte indicates the corresponding LUN is available and a 1 byte indicates LUN is presently reserved.

At task-build time, LUNs to be used by the Form Driver should be allocated (contain a 1 in the appropriate byte) to avoid LUN allocation conflicts at run time.

At least two LUNs should be allocated for each application that uses the Form Driver. These are LUN 5 used by the Form Driver for terminal I/O and a LUN other than 1 or 5 to be used as the library channel.

For example:

```
GLBPAT=T:$RESCH+2:00401
```

This TKB option will move a 1 into byte 4 and byte 5, where byte 4 would be used for the Form Driver library channel and LUN 5 would be used for the Form Driver terminal I/O.

COBOL-81 provides the BLDODL utility, which generates the proper TKB and ODL files for building COBOL applications. The /LUN option allocates specified LUNs as described above. (Refer to the *COBOL-81 RSX-11M/M-Plus User's Guide*, order number AA-M179D-TC, Part 1, Appendix D, page D-7, Section D-2 for more information on the BLDODL utility.)

## 6.11 COBOL Interface Information for COBOL Users

1. HLLCOB.OBJ is the supported interface for COBOL-11 and COBOL-81. Follow the calling conventions for FMS-11/COBOL calls as described in the *FMS-11/RSX Software Reference Manual*.

The *FMS-11/RSX Software Reference Manual* describes the standard calling conventions used in passing array arguments. These arguments should be passed BY DESCRIPTOR.

2. HLLCBL.OBJ is the standard interface for COBOL-11 compilers V4.0 and earlier. These versions of COBOL pass arguments BY REFERENCE only.

It is recommended that users of these compilers upgrade to the newest version of COBOL-81 and convert existing applications, wherever feasible, to the new coding standards. This will ensure future support from Digital Equipment Corporation, and will result in greatly improved performance. Note that HLLCBL.OBJ might not be distributed with future releases of FMS-11/RSX.

3. CBLDEM.CBL is a source copy of COBDEM.CBL with all arguments passed BY REFERENCE. The FMS-11/RSX kit contains source and command files needed to compile and build a BY REFERENCE example of the COBOL demo. Although this method of COBOL interfacing is unsupported (it is not documented in the *FMS-11/RSX Software Reference Manual*), it is provided here for customers who have not upgraded beyond COBOL-11 V4.0.
4. CBLDEM.CMD is the COBOL compile command file for CBLDEM.CBL.
5. CBLRMSTKB.CMD is the TKB task-build command file to generate the executable version of CBLDEM.CBL. It requires the RMS library LP: 1,1 RMSLIB.OLB and the FMS-11 library 30,10 FDVLRM.OLB.
6. CBLRMSDEM.TSK is the task image name produced by CBLRMSTKB.CMD.