MIGRATION RPG SCREEN FORMAT REFERENCE MANUAL

June 2011

Revision/Update Information:	This revised manual supersedes the Migration RPG Screen Format Reference Guide, Version 7.3.
Operating System and Version:	OpenVMS VAX Version 7.1 or higher
Operating System and Version:	OpenVMS Alpha Version 7.3 or higher
Operating System and Version:	OpenVMS Integrity Version 8.2 or higher
Software Version:	Migration RPG Version 8.3 or higher

Migration Specialties International, Inc. Florence, Colorado

First Printing: October 1999
Revised: January 2001
Revised: March 2002
Revised: January 2005
Revised: December 2005
Revised: June 2011

Contents

FIGURES

TABLES

Preface

This manual describes the features, uses, constructs, and syntax of interactive programming using the Migration RPG programming language on OpenVMS systems.

Intended Audience

The *Migration RPG Screen Format Reference Manual* is intended for programmers who are familiar with computer concepts and the RPG II programming language. Migration RPG was originally developed for users moving from IBM® System/36™ platforms to OpenVMS systems and was modeled after IBM System/36 RPG II. It has since been enhanced beyond the scope of IBM System/36 RPG II to provide a more generic and complete RPG environment under the OpenVMS operating system. Migration RPG still maintains a high degree of IBM System/36 compatibility.

This manual is designed to be used as a reference manual.

Conventions Used In This Manual

The following conventions are used in this manual to describe commands and keystrokes:

Convention	Meaning
CTRL/X	This sequence indicates that the user must hold down the key labeled CTRL while pressing another key.
PF1 + X	This sequence indicates that the user must first press and release the key labeled PF1, then press and release another key.
RETURN or <RETURN>	A key name is shown enclosed or within angle brackets to indicate a key on the keyboard to be pressed by the user.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format. The items are omitted because they are not important to the topic being discussed.
( )	In format descriptions, parentheses indicate that, if more than one option is chosen, the options must be enclosed in parentheses.
[ ]	In format descriptions, optional parameters in a command are denoted by square brackets. If a command delimiter, such as a comma or slash, is included within the square brackets, it is also optional. If the delimiter is outside the square brackets, it is required in the command line. Never include the square brackets in the command line.

Convention	Meaning
BOLDFACED TEXT	Commands entered by the user at the terminal are printed in boldfaced type.

Associated Documents

Additional information concerning Migration RPG can be found in the following manuals:

- *Migration RPG User's Guide*
- *Migration RPG Language Reference Manual*

Additional information concerning the use of OpenVMS with Migration RPG programs can be found in the following manuals:

- *Guide to Using OpenVMS Command Procedures*
- *OpenVMS Convert and Convert/Reclaim Utility Manual*
- *OpenVMS DCL Concepts Manual*
- *OpenVMS DCL Dictionary*
- *Guide to OpenVMS File Applications*
- *OpenVMS File Definition Language Facility Manual*
- *Guide to OpenVMS Files and Devices*
- *OpenVMS Librarian Utility Manual*
- *OpenVMS Linker Utility Manual*
- *Guide to Using OpenVMS*
- *Introduction to OpenVMS*
- *OpenVMS Sort/Merge Utility Manual*

Additional information concerning the conversion of IBM® System/36™ RPG applications to an OpenVMS system can be found in the following manuals:

- *OpenVMS S/3X Conversion Assistance Manual*
- *OpenVMS S/3X Conversion Tools User's Guide*

Additional information concerning RPG programming can be found in the following books:

- *Computer Programming - RPG II*, by Gary B. Shelly & Thomas J. Cashman
- *RPG and RPG II Programming; Applied Fundamentals, A Job Approach to Learning*, by Willian E. Bux & Edward C. Cunningham
- *RPG II Programming*, by Edward L. Essick

A special thanks to Kathy T. Parker and Gail Claremont for proofing this manual.

1 Overview of WORKSTN Files

WORKSTN files allow a user to use a Migration RPG program to interactively view and enter data via a computer terminal (CRT). Interactive Migration RPG programs have two parts: (1) language syntax in the Migration RPG program that defines the WORKSTN file, and (2) screen format(s) outside of the Migration RPG program that define the appearance of the interactive screens with which the user interfaces. Screen formats are defined in a screen format file. The screen format file is compiled by the Screen Format Generator (SFG) and is linked with the RPG WORKSTN program. A screen format is composed of Screen (S), Help (H), and Description (D) specifications.

Migration RPG interactive screens are optimized for display on VTxxx style terminals and terminal emulators. Migration RPG WORKSTN programs treat the display like a block mode device. The RPG program sends a record to the WORKSTN device for display. The WORKSTN device returns a record to the program for processing. Hence, I/O to a WORKSTN device is very much like I/O to a standard DISK file.

1.1 Screen Format Specifications

Migration RPG WORKSTN screen formats are described using Screen (S), Help (H), and Description (D) specifications. These specifications are created and maintained in a screen format source file, which is maintained separately from the Migration RPG source file.

Screen formats can contain constant data, output only fields, input only fields, and input-output fields. Display attributes such as highlighting, blinking, underlining, and bolding are provided. Field format and exit control attributes are available for input fields. Data output and attributes can be conditioned by the RPG program indicators 01 - 99. Limited data input validation is also available.

Migration RPG screens also provide the means to code help facilities via Help specifications. Help displays can be keyed to screens, portions of screens, or individual fields.

1.1.1 Screen Format Files

Screen format files typically use a .FRM extension. It is customary to use the following naming convention with screen format files:

program-nameFM.FRM

For example, the screen format file associated with the RPG program ADDRESS_BOOK.RPG would be ADDRESS_BOOKFM.FRM. See the *Migration RPG User's Guide* for more information concerning screen format file naming conventions.

1.1.2 Creating and Modifying Screen Formats

Screen format files can be created using the Migration RPG RED editor or any other text editor. The RED editor has Screen, Help, and Description specifications included within the editor to make building screen specifications as easy as possible. The *Migration RPG User's Guide* contains more information on the RED editor.

1.2 WORKSTN File Limitations

An RPG program can define and use only one WORKSTN file. If a program uses a WORKSTN file, it cannot use a KEYBORD, CRT, or CONSOLE file.

Use of the *DSPLY* opcode and a WORKSTN file in the same program can produce undesirable run-time results on the screen. Displayed information can become mixed and garbled because the WORKSTN and *DSPLY* screen handling mechanisms are unaware of each other and do not coordinate display actions. Concurrent use of a WORKSTN file and the *DSPLY* opcode in an RPG program is discouraged.

1.3 Interacting with RPG Workstation Programs

Users can interact with Migration RPG programs using the keys and command sequences described in the following sections. These sections detail workstation keyboard mapping, field editing, and characteristics of Migration RPG interactive programs.

Interactive RPG terminal support is available on all VT series terminals and terminal emulators. Support is also available for terminals and devices using DECwindows or Motif via a DECterm window.

1.3.1 Workstation Key Assignments

The following keys have been defined for Migration RPG WORKSTN programs. These key definitions cannot be modified or changed when using a VT series terminal. However, VT series terminal emulation software generally allows keyboard remapping.

If Migration RPG key mapping is altered using terminal emulation software, be sure that the command sequences sent to the program match those used for a standard VT series terminal.

Table 1–1 Command and Function Key Definition Chart

Command	Keystroke(s)
Clear	PF1 + C
Command 1 - 12	PF1 followed by 1-9, 0, -, =
Command 13 - 24	PF1 followed by !, @, #, $, %, ^, &, *, (,), _, +
DUP	PF3
ENTER/REC ADV	PF4
Help	Help or PF2 or PF1 + H
Home	PF1 + T
Print	PF1 + P
Roll Up	Next Screen or PF1 + U
Roll Down	Prev Screen or PF1 + D

1.3.1.1 Function and Command Keys

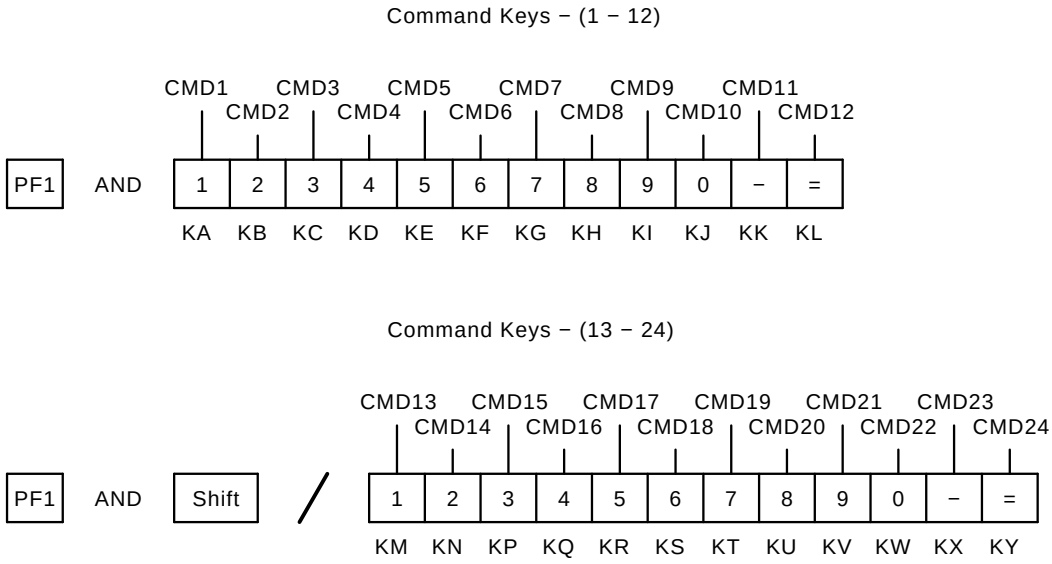
This diagram displays the command keys and their associated indicators as they would be used on a standard LK201 or LK401 keyboard. To enter a Command 7 (Cmd7), turning on the KG indicator, the user would use the following key strokes:

PF1 + 7

To enter a Command 15(Cmd15), turning on the KP indicator:

PF1 + Shift / 3

Figure 1–1 Command Key Definition Diagram



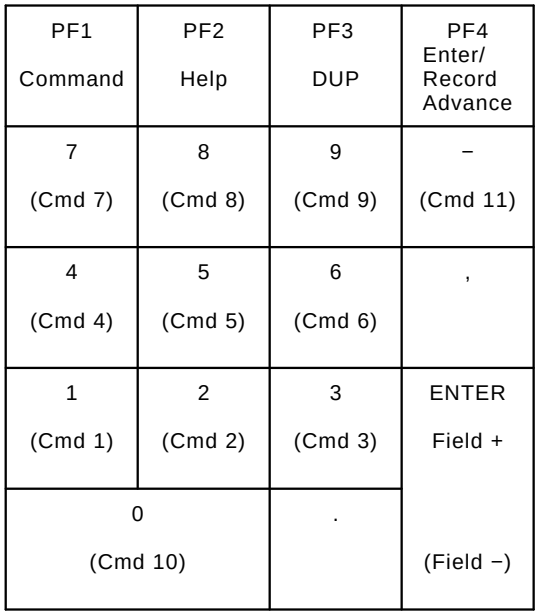
1.3.1.2 Field Editing Keys

Table 1–2 Field Editing Keys

Keystroke	Function
Up Arrow	Field exit and position cursor at first input enabled field on previous line.
Down Arrow	Field exit and position cursor at first input enabled field on next line.
Left Arrow	Within a field: Move cursor one position to the left. At beginning of field: Position cursor at the beginning of previous input enabled field.
Right Arrow	Within a field: Move cursor one position to the right. At end of field: Position cursor at the beginning of the next input enabled field.
LF or F13	Clear field.
RETURN or ENTER	Field Exit key. Any data to the right of the cursor is deleted.
PF1 + RETURN or PF1 + ENTER	Field Minus Key. Can only be used on fields defined as signed numeric. Enters the numeric field as a negative number and places a minus sign at the end of the field.
TAB	Field Advance. Advances cursor to next input enabled field on the screen.
BACKSPACE or F12	Field Backup. Current field is exited and the cursor is positioned at the beginning of the previous input enabled field.
CTRL/A	Toggles the terminal between overstrike and insert mode for character entry. If the last character in a field is not blank, the terminal will not switch to insert mode. When the last character in a field is filled, the terminal will default to overstrike mode. If CTRL/A is entered and the terminal is already in insert mode, it is switched back to overstrike mode. The terminal always defaults to overstrike mode when entering a new field.
CTRL/W	Screen Refresh Key. This command will cause the CRT screen to be cleared and repainted with the contents of the current screen.

VT series terminal keypads can also be used to enter Command Keys 0 - 11, as well as several of the Function keys. The keypads are defined as numeric within the context of the RPG program and can be used to enter numeric data. The following diagram displays the commands available on the keypad.

Figure 1–2 VT Series Terminal Keypad Diagram



The functions enclosed in parentheses are activated by first entering a Command (<PF1>) Key, followed by the desired function key. The command keys 12 - 24 are located on the standard keyboard as the characters =, !, @, #, $, %, ^, &, *, (,), _, and +, respectively.

1.3.2 Field Editing Within An RPG Workstation Screen

Editing of characters entered in a data field on a workstation screen takes place as the characters are keyed in by the user. The system will not accept invalid characters into a data field. For example, if a user attempts to key the character 'A' in a field defined as numeric (N in column 27 of the Description specification), the terminal bell will sound, the cursor will remain at the same position in the field, and the character will not be accepted by the data entry screen. The invalid character will remain displayed on the screen and the user will be able to key over it.

Invalid command and function keys are treated as field exit keys. A bell will sound and the cursor will be repositioned at the beginning of the field.

1.4 Help Support Within An RPG Workstation Program

RPG Help specifications are supported by Migration RPG. Help screen definitions must be included within the screen specification source file. The Screen Format Generator, as it compiles the Screen and Description specifications, will compile the help screens along with the data screens. Thus, the screen object module linked to the RPG workstation program will include the referenced help screens.

To access help screens from a workstation program, press either the <HELP> key, the <PF2> key on the numeric keypad, or the <PF1> key followed by the <H> key. The help screen defined for the screen location at which the cursor currently resides will be displayed. Depending upon the completion options defined in the help screen, the initial screen will be redisplayed and a prompt for input will occur, or control will be returned to the RPG workstation program.

See Chapter 4, Help Specification, for more information on Help screen specifications.

1.5 Column Separator Simulation

Description specifications can specify that column separators be used in a field (i.e., column 49 = Y). On a System/34 or System/36 using a 5251 display station, column separators are displayed using a vertical bar on either side of the column. The column separator bars do not require additional character positions on the 5251 display. This type of column separator display is not possible on a VT series terminal. Migration RPG simulates column separators by initializing each column to an underscore character ("_").

1.6 WORKSTN Example Programs

The Migration RPG Compiler Kit contains the following RPG WORKSTN program examples. These programs are used in the examples throughout this manual. The programs can be located in the S3X$EXAMPLES directory. It is recommended that the programs be copied to a different directory before working with them.

Table 1–3 RPG Example Programs

RPG Program	Screen Format File	Description
ADDRESS_BOOK.RPG	ADDRESS_BOOKFM.FRM	Interactive address book program. The program uses *READ* and *EXCPT* statements to control input and output to the WORKSTN device.
INVENT.RPG	INVENTFM.FRM	Simple interactive inventory program. The program uses *READ* statements to control WORKSTN input and *EXCPT* statements to control WORKSTN output.
INVENT_INQ.RPG	INVENT_INQFM.FRM	Simple interactive inventory inquiry program. The program uses the RPG cycle to control WORKSTN input and output.
TEMPLATE_INQ.RPG	TEMPLATE_INQFM.FRM	Interactive RPG file query program. This program template can be used to build interactive RPG file query programs.
TEMPLATE_MAINT.RPG	TEMPLATE_MAINTFM.FRM	Interactive RPG file maintenance program. This program template can be used to build interactive RPG file maintenance programs.

2 WORKSTN Files and RPG Program Specifications

Migration RPG handles WORKSTN files much like any other primary or demand input file. The coding in an RPG program for a WORKSTN file is similar to the coding for a DISK input file. WORKSTN files make use of File, Input, Calculation, and Output specifications. This chapter discusses the entries used in each RPG specification type to define and use a WORKSTN file.

2.1 WORKSTN File Specification

The File Description specification is used to describe the files accessed by an RPG program. Since a WORKSTN device is treated like a file by the RPG program, it uses a File Description specification to define the device. The WORKSTN File specification describes the following attributes of the WORKSTN file:

- File name
- File type
- File designation
- File format
- Record length
- Device type
- Continuation lines

Only one WORKSTN device can be defined in an RPG program. WORKSTN, CONSOLE, CRT, and KEYBORD devices cannot co-exist in the same program. It is strongly advised that the *DSPLY* opcode not be used in WORKSTN programs.

2.1.1 File Specification Entries

The following sections describe the entries permitted in a WORKSTN File specification. For a detailed description of all possible entries in a File specification, see the *Migration RPG Language Reference Manual*. Columns that are not specifically covered in the following sections should be left blank when defining a WORKSTN file.

2.1.1.1 Columns 1 - 5 (Line Number)

Columns 1 - 5 can be blank, specify a line number to assist in sequencing the program specifications, or can be used as a comment. Columns 1 - 5 are ignored by the compiler.

2.1.1.2 Column 6 (Specification Identification)

Column 6 must always contain an F to identify a File specification.

2.1.1.3 Columns 7 - 14 (File Name)

Columns 7 - 14 are used to assign a file name to the WORKSTN device.

The file name must begin in column 7 with an alphabetic character (A - Z). It can be 1 to 8 characters in length. The remaining characters can be A - Z, 0 - 9, $, or an underscore (_). The WORKSTN file name must be unique.

Unlike a data file, a logical name cannot be associated with the WORKSTN file name specified in columns 7 - 14. A WORKSTN device always points to the input and output user logicals SYS$COMMAND and SYS$OUTPUT.

Migration RPG allows a maximum of 98 files, including one WORKSTN file, to be open in a single program. The number of files a user can open depends on the open file quota set in the user's UAF [1] record by the system manager. To determine the number of files a user can have open at any one time, review the file quota entry in the UAF file or enter the SHOW PROCESS/QUOTA command from the user's process and look at the number to the right of *Open File Quota*. The default file quota on most OpenVMS systems is 15 open files.

2.1.1.4 Column 15 (File Type)

WORKSTN files must include the letter C in column 15 for the file type to indicate that the file is a combine input and output file.

2.1.1.5 Column 16 (File Designation)

Column 16 must contain a P (primary) or D (demand) to indicate how the program accesses the WORKSTN file.

If the WORKSTN file is defined as primary, the file is automatically read during the input phase of the RPG logic cycle. Record identifying indicators are set off at input time in the logic cycle. If the WORKSTN file is designated as primary, no secondary files are allowed in the program.

If the WORKSTN file is defined as demand, the *READ* opcode must be used within the Calculation specifications to read data from the WORKSTN device. Record identifying indicators are not set off when a *READ* operation occurs.

WORKSTN screens for both primary and demand WORKSTN files can be output using the output phase of the RPG logic cycle or by using the *EXCPT* opcode.

2.1.1.6 Column 19 (File Format)

Column 19 is used to specify the file format. A WORKSTN file always uses a fixed-length format. The entry in column 19 for a WORKSTN file must be F or blank.

[1] User Authorization File

2.1.1.7 Columns 24 - 27 (Record Length)

Columns 24 - 27 are used to specify the record length of the fixed-length WORKSTN file. Record length entries must be right-justified and numeric. Leading zeros are optional. The WORKSTN record length should be equal to or greater than the longest WORKSTN input or output record. The maximum record length is 9999.

2.1.1.8 Columns 40 - 46 (Device Code)

Columns 40 - 46 must contain the device name WORKSTN.

2.1.1.9 Column 53 (Continuation Lines)

Column 53 is used to indicate a continuation line associated to the WORKSTN File specification. Continuation lines are optional. Columns 7 - 52 of a continuation line are blank. Section 2.1.2, Continuation Lines, discusses continuation lines in detail.

WORKSTN continuation lines must immediately follow the WORKSTN specification line.

2.1.1.10 Columns 71 - 72 (File Conditioning Indicator)

Columns 71 - 72 can be used to specify an external indicator (U1 - U8) which conditions access to the WORKSTN device. If the indicator is off when the program is initialized, the WORKSTN file is set to end-of-file and is ignored.

Table 2–1 Columns 71 - 72 (File Conditioning Indicator)

Entry	Explanation
Blank	The file is not conditioned by an external indicator.
U1 - U8	The file is conditioned by the specified external indicator. If the indicator is off, the file is processed normally. If the indicator is on, the file is set to an end-of-file condition and ignored by the program. If the conditioning indicator is on, no records are processed from the file.

External indicators can be set within a program or a command procedure. See the *Migration RPG User's Guide* and *Migration RPG Language Reference Manual* for more information concerning external indicators.

When a program is initialized, conditioning indicators are evaluated and access is granted or denied to the associated file. If the external indicator is later set on within the program, it has no effect on file access. The only exception to this rule is if the program in question is an RPG subprogram. It is possible to modify external indicators settings within a subprogram, exit the subprogram, then initialize and call the subprogram again. The subprogram would re-evaluate the updated external indicators and access its files accordingly. See the *Migration RPG Language Reference Manual* for more information concerning RPG subprograms.

If a WORKSTN file is conditioned by an external indicator, all associated operations within the Calculation specifications should also be conditioned by the same indicator.

2.1.1.11 Columns 75 - 80 (Comments)

Column 75 and beyond can be used for comments. The compiler will ignore any entries in these columns.

2.1.2 Continuation Lines

A WORKSTN File specification can be immediately followed by one or more continuation lines. Continuation lines provide additional information concerning the WORKSTN device.

Continuation lines are defined by inserting a K in column 53 of the following File specifications. Continuation lines must immediately follow the WORKSTN specification. The only fields available for use on a continuation line are columns 53 through 67. The following continuation line format is accepted by Migration RPG.

Table 2–2 Continuation Line Format

Column	Contents
53	K
54-59	Option name
60-67	Value

2.1.2.1 Rules for Specifying a Continuation Line for a WORKSTN File

A WORKSTN file can have 1 to 8 continuation lines associated with it. These specifications are used to pass information between the RPG program and the WORKSTN screen and to specify the INFDS data structure and INFSR exception handling routine.

Table 2–3 WORKSTN File Continuation Options

Option Columns 54 - 59	Entry Columns 60 - 67
FMTS	Specifies the name of the screen format file containing the screens used by the program. The entry can be up to 8 characters long, running from columns 60 - 67. This entry is ignored by the Migration RPG compiler and is supported for compatibility purposes with non-OpenVMS versions of the RPG programming language. However, the *BUILD* procedure provided with the Migration RPG Compiler Kit will process the FMTS continuation line. See the description of the BUILD procedure in the *Migration RPG User's Guide* for more information on the BUILD procedure.
ID	A two-character, alphanumeric field identifying the terminal or WORKSTN from which the program is running. See the *Migration RPG User's Guide* for information on establishing and changing WORKSTN ID's.
IND	This function has no meaning under Migration RPG. It is supported for compatibility purposes with non-OpenVMS versions of the RPG programming language. The Migration RPG compiler will ignore IND continuation lines.
INFDS	The name of a data structure which will receive information from a WORKSTN screen concerning its return status and any exceptions processed. The data structure must be defined in the Input specifications. The INFDS data structure is described in detail in Chapter 8, INFDS Data Structure.
INFSR	Name of the exception handling subroutine. This subroutine must be defined in the Calculation specifications. The INFSR subroutine is called if an exception occurs during a WORKSTN read and no other error trapping has been specified for the read. The INFSR subroutine is described in detail in Chapter 9, INFSR Subroutine.
NUM	This function has no meaning under Migration RPG. It is supported for compatibility purposes with non-OpenVMS versions of the RPG programming language. The Migration RPG compiler will ignore NUM continuation lines.
SAVDS	This function has no meaning under Migration RPG. It is supported for compatibility purposes with non-OpenVMS versions of the RPG programming language. The Migration RPG compiler will ignore SAVDS continuation lines.
SLN	A 2-digit, 0-decimal, numeric field which is passed to the screen format file. The SLN field defines the starting line for a variable start line screen (V coded in column 17 of the Screen specification). The start line field must be defined within the RPG program. If a variable start line number is not specified, all screens having a variable start line will start on line 01.

2.1.3 WORKSTN File Specification Example

Example 2–1 WORKSTN File and Continuation Specifications

```
         1         2         3         4         5         6
123456789012345678901234567890123456789012345678901234567890123456789
      *
     FWORKSTN CP  F     167            WORKSTN
     F                                                KFMTS  MNVACFM
     F                                                KINFSR INFSR
     F                                                KINFDS INFDS
     F                                                KSLN   SL
```

This example shows a WORKSTN file defined as a combine primary file. The WORKSTN file has FMTS, INFSR, INFDS, and SLN continuation specifications associated with it.

2.2 WORKSTN Input Specifications

Migration RPG Input specifications are used to describe the input records and fields associated with a WORKSTN file. The Input specifications extract data from the WORKSTN record buffer and set any record identifying indicators on. Input specifications for a WORKSTN file are no different than Input specifications for any other DISK device input file. Input specifications are described in detail in the *Migration RPG Language Reference Manual*.

The WORKSTN Input specification describes the WORKSTN input records. The specifications can be divided into two categories:

- File and record descriptions. Columns 7 - 42 describe the file and its records.
- Field descriptions. Columns 43 - 74 describe the fields in each record.

2.2.1 Input Record Specification Entries

The following sections describe the entries permitted in a WORKSTN Input record specification. For a detailed description of all possible entries in a Input record specification, see the *Migration RPG Language Reference Manual*.

2.2.1.1 Columns 1 - 5 (Line Number)

Columns 1 - 5 can be blank, specify a line number to assist in sequencing the program specifications, or can be used as a comment. Columns 1 - 5 are ignored by the compiler.

2.2.1.2 Column 6 (Specification Identification)

Column 6 must always contain an I to identify an Input specification.

2.2.1.3 Columns 7 - 14 (File Name)

Columns 7 - 14 must contain the WORKSTN file name. This is the same name that appears in columns 7 - 14 of the WORKSTN File specification.

If this entry is blank, the compiler will assume that the information in this line describes a record from the last file named in an Input record specification. All records and fields associated with one file should be described in one contiguous section of Input specifications.

2.2.1.4 Columns 14 - 16 (AND/OR)

AND or OR can be specified in columns 14 - 16 to show a relationship between record identifying indicators or record types. The entry must be left justified.

2.2.1.5 Columns 15 - 16 (Sequence)

Columns 15 - 16 are used to specify the sequence that defines the ordering sequence of the record types in a file. The program does not order records according to sequence. Sequencing is used to determine if the records are input in the correct order and to notify the user if they are not.

Input specifications with an AND or OR specified in columns 14 - 16 cannot have a sequence code.

2.2.1.6 Column 17 (Sequence Number)

If a numeric sequence code is assigned in columns 15 - 16, column 17 is used to indicate the number of records in a sequence group. Leave this column blank if an alphabetic sequence has been specified in columns 15 - 16.

Input specifications with an AND or OR specified in columns 14 - 16 cannot have a sequence number specified in column 17.

2.2.1.7 Column 18 (Option)

If a numeric sequence code has been assigned in columns 15 and 16, column 18 can be used to specify whether a record of that type must be present in a sequence group.

Leave this column blank if a blank or alphabetic sequence has been specified in columns 15 - 16. If all records are listed as optional, no sequence checking is performed. Input specifications with an AND or OR specified in columns 14 - 16 cannot have a sequence option specified in column 18.

2.2.1.8 Columns 19 - 20 (Record-Identifying Indicator)

Specifying an indicator in columns 19 - 20 associates the indicator with a particular record type. When a record of the type specified for this program line is processed, the indicator is set on. The indicator remains on through detail-time output, unless set off by the programmer. After detail-time output, all indicators used as record-identifying indicators are set off. See the *Migration RPG Language Reference Manual* for more information on the RPG logic cycle and record identifying indicators.

2.2.1.9 Columns 21 - 41 (Record Identification Conditions)

Columns 21 - 41 can be used to specify information used to define a record type. If all records in a file are to be processed regardless of type, or if all records have the same type, leave columns 21 - 41 blank.

2.2.1.10 Column 42 - 70 (No-op)

Columns 42 - 70 must be left blank on an Input record specification.

2.2.1.11 Columns 71 - 80 (Comments)

Column 71 and beyond can be used for comments. Entries in these columns are ignored by the compiler.

2.2.2 Input Field Specification Entries

The following sections describe the entries permitted in a WORKSTN Input field specification. For a detailed description of all possible entries in an Input field specification, see the *Migration RPG Language Reference Manual*. Field descriptions must begin one line below the Input record specification. Use a separate line to describe each field.

2.2.2.1 Columns 1 - 5 (Line Number)

Columns 1 - 5 can be blank, specify a line number to assist in sequencing the program specifications, or can be used as a comment. Columns 1 - 5 are ignored by the compiler.

2.2.2.2 Column 6 (Specification Identification)

Column 6 must always contain an I to identify an Input specification.

2.2.2.3 Columns 7 - 42 (no-op)

Columns 7 - 42 must be left blank on an Input field definition.

2.2.2.4 Column 43 (Data Format)

Column 43 is used to describe the format in which the field defined in columns 53 - 58 is stored. Column 43 should always be blank for WORKSTN fields.

2.2.2.5 Columns 44 - 47 and 48 - 51 (Field Start and End Positions)

Columns 44 - 47 and 48 - 51 are used to define the physical start and end positions of a field in a record. These entries do not refer to the location of the field on the screen. The input fields are placed in the Input record in the order in which they appear in the screen Description specifications.

The maximum length of a WORKSTN field depends on the type of data it contains.

- The maximum length of a numeric field is 15.
- The maximum length of an alphanumeric field is 256.
- The maximum length of a data structure is 9,999 characters.

Fields can overlap as long as each field is given a different name.

Both start and end position entries must be right-justified and numeric. Leading zeros can be omitted.

2.2.2.6 Column 52 (Decimal Positions)

Column 52 is used to identify a field as numeric and to specify the number of digits to the right of the decimal point. The decimal point is implied. It does not appear in numeric fields passed to a screen unless the fields are output using an edit code or mask. RPG does not expect a decimal point to be present in numeric fields returned from the screen to the program.

A value must be specified in this column for a numeric field even if the field has no decimal points. In this case, specify a zero.

2.2.2.7 Columns 53 - 58 (Field Name)

Columns 53 - 58 must contain the name of a field, array, array element, or data structure.

2.2.2.8 Columns 59 - 60 (Control Break Indicator)

Columns 59 - 60 must be blank for WORKSTN fields. Control break indicators are not allowed on WORKSTN fields.

2.2.2.9 Columns 61 - 62 (Matching Fields)

Columns 61 - 62 must be blank for WORKSTN fields. Matching record indicators are not allowed on WORKSTN fields.

2.2.2.10 Columns 63 - 64 (Field Record Relation Indicator)

Columns 63 - 64 are used to specify field record relation indicators. Field record relation indicators control the conditions under which data is extracted from the input buffer into a field. These conditions include control breaks, matching fields, halts, and external indicators.

2.2.2.11 Columns 65 - 70 (Field Indicators)

Columns 65 - 70 can be used to specify field indicators. Field indicators check the contents of numeric or alphanumeric fields when they are extracted from the input record.

2.2.2.12 Columns 71 - 80 (Comments)

Column 71 and beyond can be used for comments. Entries in these columns are ignored by the compiler.

2.2.3 First Cycle Read Processing of a WORKSTN Device

The first record read from a combine primary WORKSTN device is blank unless the first page indicator (1P) is used to output an initial WORKSTN screen during program startup. The WORKSTN Input specifications should be coded to allow for the initial blank input record if a 1P screen is not output. A WORKSTN device defined as a combine demand file will only return a blank record if a *READ* is performed before a WORKSTN screen is output.

2.2.4 WORKSTN Input Example

Example 2–2 WORKSTN Input Specifications

```
         1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901234567890
      *
     IWORKSTN XX       1 CK
     I                                          2  06 DLR#
     I                                          7  41 DNAME
     I         XX      1 CI
     I         XX      1 CF
```

This example shows WORKSTN Input specifications. Note the use of record identification codes in columns 21 - 27. It is a common practice to reserve position 1 of the WORKSTN input record for a screen identification code when using multiple WORKSTN screens. This practice helps ensure that the input data is placed in the correct input fields. Optional input record indicators in columns 19 - 20, which are not used in this example, can be used to identify WORKSTN input records.

2.3 WORKSTN Calculation Specifications

A WORKSTN file is treated just like any other update capable file in the Calculation specifications. There are no WORKSTN specific commands or command parameters in the Calculation specifications.

If the WORKSTN file has been defined as a combine primary file, input is handled via the RPG logic cycle. If the WORKSTN file has been defined as a combine demand file, input is achieved via the *READ* opcode.

Both combine primary and combine demand WORKSTN files can use the output phase of the RPG logic cycle or the *EXCPT* opcode to control screen output. *EXCPT* statements will cause output from all exception Output specifications with conditioning indicators set on and matching EXCPT names.

See the *Migration RPG Language Reference Manual* for a detailed description of the *READ* and *EXCPT* opcodes.

2.4 WORKSTN Output Specifications

The Output specification is used to describe the records and fields in an output file and the conditions under which data is output. Output specifications for WORKSTN files are similar to Output specifications for other DISK device output files. The first field in each WORKSTN output record must define the screen upon which the output data is to be displayed. WORKSTN screens do not accept packed-decimal, zoned-decimal, or binary output fields. Output specifications are described in detail in the *Migration RPG Language Reference Manual*.

An Output specification is similar to an Input specification in that it can be divided into two general categories. Columns 7 - 37 are used to describe output records. Columns 23 - 74 are used to describe output fields. Field description entries always begin one line below record description entries.

2.4.1 Output Record Specifications

This section describes the entries used in WORKSTN Output record specifications. Columns not specifically covered in this section should be left blank on WORKSTN Output record specifications.

2.4.1.1 Columns 1 - 5 (Line Number)

Columns 1 - 5 can be blank, specify a line number to assist in sequencing the program specifications, or can be used as a comment. Columns 1 - 5 are ignored by the compiler.

2.4.1.2 Column 6 (Specification Identification)

Column 6 must always contain an O to identify an Output specification.

2.4.1.3 Columns 7 - 14 (File Name)

Columns 7 - 14 must contain the name of the WORKSTN file. The name must be the same as the file name specified in columns 7 - 14 of the WORKSTN File specification.

If this entry is blank, the compiler will assume that the information in this line describes a record from the last file named in an Output specification. All records and fields associated with one file should be described in one contiguous section of Output specifications.

2.4.1.4 Columns 14 - 16 (AND/OR)

Columns 14 - 16 can be used to specify AND or OR conditions, which are used to link output lines together, allowing more than three conditioning indicators to be used to condition an output record.

2.4.1.5 Column 15 (Record Type)

Column 15 is used to specify the type of record to be output. An entry in column 15 is required for every output record defined in the Output specifications.

2.4.1.6 Columns 23 - 31 (Output Indicators)

Columns 23 - 31 can be used to specify indicators to condition the output of records. Up to three indicators can be specified on an Output specification. Preceding an indicator with N causes the condition to be valid only when the associated indicator is not on. Use columns 23 - 25 to describe the first indicator, columns 26 - 28 to describe the second indicator, and columns 29 - 31 to describe the third indicator. Using the indicators in this way forms an AND relationship. Use AND or OR codes in columns 14 - 16 if it is necessary to condition an output record definition with more than three indicators.

2.4.1.7 Columns 32 - 37 (EXCPT Name)

Columns 32 - 37 can be used to specify an EXCPT name for the output record if the record type specified in column 15 is E. An *EXCPT* operation can specify the name in factor 2.

2.4.1.8 Columns 37 - 70 (no-op)

Columns 37 - 70 must be left blank in an output record specification.

2.4.1.9 Columns 71 - 80 (Comments)

Column 71 and beyond can be used for comments. The compiler will ignore entries in these columns.

2.4.2 Output Field Specifications

This section describes the entries used in WORKSTN Output field specifications. Columns not specifically covered in this section should be left blank on WORKSTN Output field specifications.

2.4.2.1 Columns 1 - 5 (Line Number)

Columns 1 - 5 can be blank, specify a line number to assist in sequencing the program specifications, or can be used as a comment. Columns 1 - 5 are ignored by the compiler.

2.4.2.2 Column 6 (Specification Identification)

Column 6 must always contain an O to identify an Output specification.

2.4.2.3 Columns 7 - 22 (no-op)

Columns 7 - 22 must be blank when defining an output field.

2.4.2.4 Columns 23 - 31 (Output Indicators)

Columns 23 - 31 can be used to specify indicators to condition the output of fields. Up to three indicators can be specified on an Output field specification. Preceding an indicator with N causes the condition to be valid only when the associated indicator is not on. Use columns 23 - 25 to describe the first indicator, columns 26 - 28 to describe the second indicator, and columns 29 - 31 to describe the third indicator. Using the indicators in this way forms an AND relationship. Output field definitions can be conditioned by a maximum of three indicators.

2.4.2.5 Columns 32 - 37 (Field Name)

Columns 32 - 37 can be used to specify a field name, table name, array name, array element, data structure name, or special word for output.

Note: The first field entry in each WORKSTN output record must be an output constant containing the screen name.

2.4.2.5.1 Output Field Names

All output field names must have been previously defined in an Input, Extension, or Calculation specification. The fields must be coded in the Output specifications in the same order they are coded in the screen's Description specifications.

2.4.2.5.2 Output Special Words

The special words PAGE, PAGE1 - PAGE7, UDATE, UDAY, UMONTH, UYEAR, $UDAY, $UMNTH, $UYEAR, and $UDATE can all be used in WORKSTN Output field specifications.

2.4.2.6 Column 38 (Edit Codes)

Column 38 can be used to specify an edit code. Edit codes can be used to edit a numeric field without specifying an edit word.

2.4.2.7 Column 39 (Blank After)

Column 39 can be used to specify blank after, which initializes a field after it has been output. Alphanumeric fields are initialized to blanks and numeric fields are initialized to zeros.

2.4.2.8 Columns 40 - 43 (End Position in Output Record)

Columns 40 - 43 are used to indicate the location of an output field or constant in an output record. Specifying an end position is optional with Migration RPG. If no end position is specified, the compiler will calculate one based on the output field size. Field size calculation includes allowances for any edit codes or edit words being applied to the field.

Use K in column 42 if a WORKSTN screen format name is being specified.

2.4.2.9 Column 44 (Output Data Format)

The output data format should always be blank for WORKSTN output fields. Migration RPG WORKSTN screens do not support the output of packed-decimal, zoned-decimal, or binary fields.

2.4.2.10 Columns 45 - 70 (Constant, Edit Word, or WORKSTN Screen Format Name)

Columns 45 - 70 can be used to specify an output constant, edit word, or WORKSTN screen format name. An edit word can be used to modify an edit code specified in column 38 or to define how data will be formatted when it is output.

2.4.2.10.1 WORKSTN Screen Format Name

Columns 45 - 54 of the first field in a WORKSTN output record must be used to specify a WORKSTN screen format name. The WORKSTN screen format name is used to designate the screen format that will display the data being passed by the program.

A WORKSTN screen format name requires a K in column 42. Only one screen format name can be specified for each WORKSTN output record. A screen format name must be specified for each WORKSTN output record.

A WORKSTN screen format name must be enclosed by single quotation marks (’). The leading quotation mark must be placed in column 45.

A WORKSTN screen format name can be 1 - 8 characters in length. The length of the screen format name is specified in column 43. For example, if a WORKSTN screen format was named ASBECK, the entry in columns 40 - 43 (end position) would be K6.

The WORKSTN screen format name must match a screen format name specified in columns 7 - 15 of a Screen specification in the screen format file. If it does not, a runtime error will occur.

The Output specification containing the WORKSTN screen format name cannot be qualified by conditioning indicators in columns 23 - 31.

The Output field specifications that follow the screen specification define data that is to be displayed on the screen.

Example 2–3 WORKSTN Output Specification

```
         1         2         3         4         5         6
123456789012345678901234567890123456789012345678901234567890123456789
      *
     OWORKSTN E                KEY
     O                                   K3 'KEY'
     O                         DLR#
     O                         DNAME
     O                         ERRMSG
```

This example shows an output definition for a WORKSTN file. The screen KEY is to be displayed when this specification is output.

Note: In this example, field end positions have not been specified in columns 40 - 43. The Migration RPG compiler will compute the field end positions based on the field size.

2.4.2.11 Columns 71 - 80 (Comments)

Column 71 and beyond can be used for comments. The compiler will ignore entries in these columns.

3 Screen Specification

The Screen specification defines a screen format. One Screen specification is required for each screen defined in a screen format file. A screen format file can contain multiple screen formats. A Screen specification will be the first compilable line in any screen format file.

3.1 Columns 1 - 5 (Line Number)

Columns 1 - 5 can be blank, specify a line number to assist in sequencing the screen specifications, or can be used as a comment. Columns 1 - 5 are ignored by the compiler.

3.2 Column 6 (Specification Identification)

Column 6 must always contain an S to identify a Screen specification.

3.3 Column 7 (Comment)

An asterisk (*) in column 7 indicates that the line is a comment line. The line will be ignored by the compiler. Column 6 can be blank on a comment line.

3.4 Columns 7 - 14 (Screen Format Name)

Columns 7 - 14 are used to identify a screen format that can be accessed by an RPG program, the MENU Utility, or the PROMPT Utility. The screen name must begin in column 7 with an alphabetic character (A - Z). It can be 1 to 8 characters in length. The remaining characters can be A - Z, 0 - 9, $, or an underscore (_). Each screen name in a screen format file must be unique.

3.5 Columns 15 - 16 (Reserved)

Columns 15 - 16 are not used at present. They are reserved for future use.

3.6 Columns 17 - 18 (Starting Line Number)

Columns 17 - 18 contain the line number of the screen that this format will start on. Counting begins from the top of the screen with line one. A maximum of 48 lines is allowed on a Migration RPG screen format.

Table 3–1 Columns 17 - 18 (Starting Line Number)

Entry	Definition
Blank	Default value of 1 will be used as starting line number
01 - 48	The exact line number the display will start on. This number cannot exceed 48. A leading zero is not required, but the number must be right-justified
V	Variable line number means the line number is to be supplied by the program at execution time. If no line number is supplied by the program, the default value 1 will be used. The V must be in column 17 if used.

All lines numbers specified in Help and Description specifications for a screen format are relative to the start line number.

3.7 Columns 19 - 20 (Lines To Clear)

Columns 19 - 20 contain the number of lines to clear when the format is displayed. The starting line number is the first line cleared. For example, if the start line is 10 and the lines to clear value is 6, lines 10 - 15 will be cleared on the CRT before the current screen format is displayed.

Table 3–2 Columns 19 - 20 (Lines To Clear)

Entry	Definition
Blank	All lines of current display starting with the start line are cleared.
0 - 48	The number of lines that will be cleared starting with the start line. This number cannot exceed 48. A leading zero is not required, but the number must be right-justified

Caution should be used when not clearing an entire display, since part of the previous display will be left on the screen. If overlaying a format that has input fields, only the input fields on the new display will be active.

3.8 Column 21 (Allow Lowercase)

Column 21 specifies if lowercase alphabetic characters are allowed to be input within the display format. This entry can be overridden for specific fields by the allow lowercase entry in column 51 of the Description specification.

Table 3–3 Column 21 (Allow Lowercase)

Entry	Definition
N or Blank	All alphanumeric fields default to uppercase.
Y	Lowercase entry in alphanumeric fields is allowed.

3.9 Column 22 (Return Input)

Column 22 specifies if data is always returned from the screen format to the program. This feature is supported to maintain compatibility with other versions of RPG. Normally column 22 is left blank in Migration RPG screen formats.

Table 3–4 Column 22 (Return Input)

Entry	Definition
Y or Blank	All input fields are returned to the program. If mandatory entry fields (Y in column 29 of the Description specification) are present in the screen format, the user must enter data in these fields before the display will return to the program.
N	The contents of the input fields are only returned to the program if the user enters data in one or more fields on the screen. If the user enters no data, a blank record is returned to the program. If the user enters no data, then any mandatory entry fields in the screen format are ignored.

3.10 Columns 23 - 24 (Reserved)

Columns 23 - 24 are not used at present. They are reserved for future use.

3.11 Columns 25 - 26 (Sound Alarm)

Columns 25 - 26 specify if an alarm should sound when the screen is displayed.

Table 3–5 Columns 25 - 26 (Sound Alarm)

Entry	Definition
N or Blank	Alarm will not sound.
Y	Alarm will sound when screen format is displayed.
01 - 99	If the program indicator is on when the screen format is displayed, the alarm will sound. If the indicator is off, the alarm will not sound.

3.12 Column 27 (Enable Function Keys)

Column 27 specifies if function keys are enabled or disabled. For a user to use a function key, it must first be enabled. Section 7.2, Function Keys, describes function key usage in detail.

The specific function keys to be enabled or disabled are defined by numbers in the key mask in columns 64 - 79 of the Screen specification.

Table 3–6 Column 27 (Enable Function Keys)

Entry	Definition
Blank	All function keys are enabled. Key mask entries are ignored.
N	Function keys defined in the key mask are disabled. Function keys not defined in the key mask are enabled.
Y	Function keys defined in the key mask are enabled. Function keys not defined in the key mask are disabled.
R (retain)	Function keys that were enabled before the current format is displayed will remain enabled.

3.13 Column 28 (Enable Command Keys)

Column 28 specifies if command keys are enabled or disabled. Order for a user to use a command key, it must first be enabled. Section 7.1, Command Keys, describes command key usage in detail.

The specific command keys to be enabled or disabled are defined by alphabetic characters in the key mask in columns 64 - 79 of the Screen specification.

Table 3–7 Column 28 (Enable Command Keys)

Entry	Definition
Blank	All command keys are enabled. Key mask entries are ignored.
N	Command keys defined in the key mask are disabled. Command keys not defined in the key mask are enabled.
Y	Command keys defined in the key mask are enabled. Command keys not defined in the key mask are disabled.
R (retain)	Command keys that were enabled before the current format is displayed will remain enabled.

3.14 Columns 29 - 30 (Blink Cursor)

Columns 29 - 30 specify if the cursor should blink when the screen format is displayed. This qualifier is maintained for compatibility with other versions of RPG. Cursor attributes can also be set via the terminal setup functions, which may override the Migration RPG settings.

Table 3–8 Columns 29 - 30 (Blink Cursor)

Entry	Definition
N or Blank	Cursor will not blink.
Y	Cursor will blink when screen format is displayed.
01 - 99	If the program indicator is on when this screen format is displayed, the cursor will blink. If the indicator is off, the cursor will not blink.

3.15 Columns 31 - 32 (Erase Input Fields)

Columns 31 - 32 specify that when a screen format is displayed, the input and input/output fields are to be erased.

Table 3–9 Columns 31 - 32 (Erase Input Fields)

Entry	Definition
N or Blank	Input and input/output fields in screen format will not be erased.
Y	Input and input/output fields in screen format will be erased when screen format is displayed.
01 - 99	If the program indicator is on when this screen format is displayed, the input and input/output fields in screen format will be erased. If the indicator is off, the fields will not be erased.

3.16 Columns 33 - 34 (Override Fields)

Columns 33 - 34 specify that fields in the screen format can be overridden based on indicator settings. This function is useful when trapping and displaying input errors. Overrides should be carefully planned, since the processing is somewhat complex.

Override operations generally work like this:

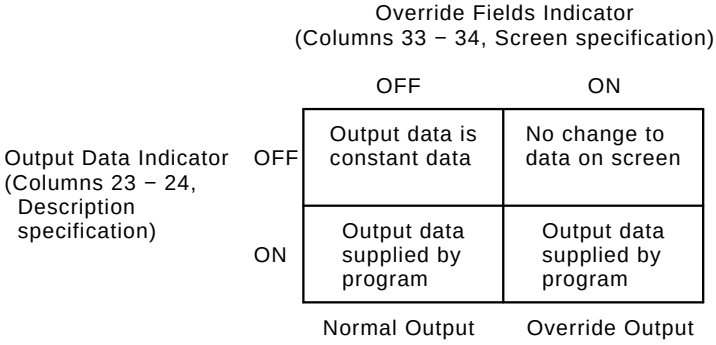
1. A user completed entry of data on a screen and uses Enter/Record Adv to return the data to the program. The override indicator in columns 33 - 34 would be off.
2. The program locates an error in the data entered. It turns on the override indicator defined in columns 33 - 34 and redisplays the screen format. During the override display, only fields using indicator control on output are modified by the screen output function. Output data, highlighting, and cursor control indicators are used to direct the user to the fields containing the invalid data.
3. The user corrects the invalid fields and resubmits the data to the program. The override cycle can be repeated as long as the data submitted is not valid.

Table 3–10 Columns 33 - 34 (Override Fields)

Entry	Definition
N or Blank	The screen format does not allow overrides.
Y	Override operations are performed every time the screen format is output. This is not recommended, since overrides should be indicator controlled.
01 - 99	If the program indicator is on when the screen format is displayed, the following occurs: • If a field has an indicator specified for output data (columns 23 - 24 in the Description specification) and the indicator is off, the data in the field is unchanged. Fields with a blank, Y, or N in columns 23 - 24 are also unchanged. • If a field has an indicator specified for output data and the indicator is on, output data from the program is displayed. Any previous data entered on the screen is overridden. If the override indicator is off in columns 33 - 34 of the Screen specification, override operations are not performed.

Figure 3–1 summarizes the effect of indicators on output data during an override operation:

Figure 3–1 Indicator-based Output Data and Override



3.17 Columns 35 - 36 (Suppress Input)

Columns 35 - 36 specify whether input is to be returned from the screen. If input is suppressed, a blank input record is returned to the program after the screen is displayed. Suppressing input is useful when displaying multiple screen formats where input will only be required from the last screen.

Table 3–11 Columns 35 - 36 (Suppress Input)

Entry	Definition
N or Blank	Input will be returned to the program.
Y	No input will be returned to the program.
01 - 99	If the program indicator is on when the screen format is displayed, no input data will be returned to the program. If indicator is off, input data will be returned to the program.

3.18 Columns 37 - 38 (Reserved)

Columns 37 - 38 are not used at present. They are reserved for future use.

3.19 Column 39 (132-Column Format)

Column 39 specifies if the screen will use an 80 or 132 character screen width.

Table 3–12 Column 39 (132-Column Format)

Entry	Definition
N or Blank	80 character screen width is used.
Y	132 character screen width is used.

Note: When designing screen formats, it is recommended that the screen width on all screens used by the program be the same, either 80 column or 132 column. Switching screen widths between formats within a program changes the display characteristics of the display terminal and may produce undesireable results. Maintaining a consistent display width produces consistent output.

Note: If PC-based terminal emulators are being used to access Migration RPG interactive programs, be sure to test 132-column displays on the emulator before putting them into use. Some terminal emulation software has problems supporting a 132-column display.

3.20 Column 40 (Right-to-left Display)

Column 40 controls cursor movement between fields on a screen. By default, the cursor will move from left to right, top to bottom across the screen as the user enters data. Placing a Y in column 40 causes the cursor to move from right to left, top to bottom as data is entered. This option is useful when writing software for cultures that use a language that is read and written right to left.

Column 27 in the Description specification controls cursor movement within a field.

Table 3–13 Column 40 (Right-to-Left Display)

Entry	Definition
N or Blank	Cursor starts in first unprotected input field in upper left portion of screen and moves left to right, top to bottom, between the fields.
Y	Cursor starts in first unprotected input field in upper right portion of screen and moves right to left, top to bottom, between fields.

3.21 Columns 41 - 63 (Reserved)

Columns 41 - 63 are not used at present. They are reserved for future use.

3.22 Columns 64 - 79 (Key Mask)

Columns 64 - 79 specify the function and command keys to enable or disable for the screen. Command keys are identified by alphabetic characters and function keys are identified by numbers. The key mask cannot contain embedded blanks. Command and function keys can be specified in any order in the mask. Chapter 7, WORKSTN Command and Function Keys, contains a detailed description of command and function key definitions and usage. Section 3.12, Column 27 (Enable Function Keys), and Section 3.13, Column 28 (Enable Command Keys), describe how to enable or disable function and command keys for a screen.

Table 3–14 Columns 64 - 79 (Key Mask)

Function Keys Enabled	Command Keys Enabled	Key Mask Entries
N	N	All function and command keys listed in the mask are disabled.
Y	N	All function keys listed in the key mask are enabled. All command keys listed in the mask are disabled.
N	Y	All function keys listed in the mask are disabled. All command keys listed in the mask are enabled.
Y	Y	All function and command keys listed in the mask are enabled.

The following tables define the mask entries for command and function keys:

Table 3–15 Command Key Mask Entries

Command Key	Key Mask Entry
Cmd1	A
Cmd2	B
Cmd3	C
Cmd4	D
Cmd5	E
Cmd6	F
Cmd7	G
Cmd8	H
Cmd9	I
Cmd10	J
Cmd11	K
Cmd12	L
Cmd13	M
Cmd14	N
Cmd15	P
Cmd16	Q
Cmd17	R
Cmd18	S
Cmd19	T
Cmd20	U
Cmd21	V
Cmd22	W
Cmd23	X
Cmd24	Y

Table 3–16 Function Key Mask Entries

Function	Key Mask Entry
Print	1
Roll-Up	2
Roll-Down	3
Clear	4
Help	5
Home	6

3.23 Column 80 (Reserved)

Column 80 is not used at present. It is reserved for future use.

4 Help Specification

Help specifications are optional specifications that can be placed between the Screen specification and the first Description specification in a screen format definition. Help specifications provide a means to associate help text with the entire screen format, specific areas within the screen format, and individual fields within the screen format. Help screens can be unique to a data screen or shared among several data screens.

Each Help specification references a Help screen. A Help screen is defined as a normal data screen containing only constant data. Migration RPG Help screens do not support input and output data fields and do not exchange data records with the program. Help screen processing occurs entirely within the screen management section of the Migration RPG Runtime System. Help screens must be defined within the screen format source file in which they are referenced.

When a user presses the Help key, the Migration RPG Runtime System looks for a Help specification associated to the portion of the screen in which the cursor is located. The Help specification must be defined within the current screen format. If a Help specification matches the cursor location, the associated Help screen is displayed. The user can then use the Roll-Up and Roll-Down keys to page to other Help screens, if they are available. Pressing a field exit key (Enter, Tab) or record entry key (Enter/Record Adv) will terminate Help mode and place the user back on the data screen.

4.1 Columns 1 - 5 (Line Number)

Columns 1 - 5 can be blank, specify a line number to assist in sequencing the screen specifications, or can be used as a comment. Columns 1 - 5 are ignored by the compiler.

4.2 Column 6 (Specification Identification)

Column 6 must always contain an H to identify a Help specification.

4.3 Column 7 (Comment)

An asterisk (*) in column 7 indicates that the line is a comment line. The line will be ignored by the compiler. Column 6 can be blank on a comment line.

4.4 Columns 7 - 14 (Help Screen Format Name)

Columns 7 - 14 are used to identify the Help screen that is associated to the help area defined in columns 34 - 42. The format name must be exactly eight characters in length and must be formatted as follows:

Table 4–1 Columns 7 - 14 (Help Screen Format Name)

Position	Definition
1	Must be an alphabetic character from A - Z.
2 - 6	Must be 5 alphanumeric characters. The first six characters in the help screen format name should match if more than one screen is used to provide help information with a help area.
nn	Must be a numeric value from 00 to 99. Up to 100 help screens can be associated with a single help area.

Section 4.16, Help Screen Processing, provides detailed information on how help screen naming conventions determine the order in which help screens are processed.

4.5 Columns 15 - 33 (Reserved)

Columns 15 - 33 are not used at present. They are reserved for future use.

4.6 Columns 34 - 37 (Upper Left Boundary)

Columns 34 - 37 contain the line number and the column number of the upper left corner of the help area covered by the Help specification. When this is combined with the lower right corner defined in columns 39 - 42, a rectangular area on the screen is defined which describes the help area covered by this Help specification. A help area can range from a single character position to the entire screen. When the cursor is within this area and the Help key is pressed, the program will call up the help screen associated with this Help specification.

The upper left boundary location is specified by **rrcc**, where **rr** is the row number (1 - 48) and **cc** is the column number (1 - 80 for an 80-column display, 1 - D2 for a 132-column display). Numbers 100 - 132 on a 132-column display are represented as follows:

Table 4–2 100 to 132 Two Digit Representation

Column	Entry	Column	Entry
100	A0	117	B7
101	A1	118	B8
102	A2	119	B9
103	A3	120	C0
104	A4	121	C1
105	A5	122	C2
106	A6	123	C3
107	A7	124	C4
108	A8	125	C5
109	A9	126	C6
110	B0	127	C7
111	B1	128	C8
112	B2	129	C9
113	B3	130	D0
114	B4	131	D1
115	B5	132	D2
116	B6		

4.7 Column 38 (Reserved)

Column 38 is not used at present. It is reserved for future use.

4.8 Columns 39 - 42 (Lower Right Boundary)

Columns 39 - 42 contain the line number and the column number of the lower right corner of the help area covered by the Help specification. When this is combined with the upper left corner defined in columns 34 - 37, a rectangular area on the screen is defined which describes the help area covered by this Help specification. A help area can range from a single character position to the entire screen. When the cursor is within this area and the Help key is pressed, the program will call up the help screen associated with this Help specification.

The lower right boundary location is specified by **rrcc**, where **rr** is the row number (1 - 48) and **cc** is the column number (1 - 80 for an 80-column display, 1 - D2 for a 132-column display). The two digit representation of the numbers 100 - 132 on a 132-column display are listed in Table 4–2, 100 to 132 Two Digit Representation.

4.9 Column 43 (Reserved)

Column 43 is not used at present. It is reserved for future use.

4.10 Columns 44 - 45 (Suppress Selection Indicator)

Columns 44 - 45 can be used to specify an indicator to suppress the display of the help screen. In certain instances, it may be desirable to suppress the display of select help screens when the user accesses help. Placing an indicator in columns 44 - 45 and turning it on in the program accomplish this.

Table 4–3 Columns 44 - 45 (Suppress Selection Indicator)

Code	Definition
blank	This Help specification is available when the Help key is pressed and the cursor is within the boundary of this help area.
01 - 99	This Help specification is available when the indicator is off. When the indicator is on, the Help specification is not available and the associated help screen will not be displayed if the Help key is pressed while the cursor is within the boundary of this help area.

4.11 Column 46 (Reserved)

Column 46 is not used at present. It is reserved for future use.

4.12 Columns 47 - 48 (Restore Application Format)

Columns 47 - 48 specifies if the data screen is to be redisplayed if the user exits help using a command key. A command key can be used to exit a help screen if the command key is enabled in the key mask of the help Screen specification.

Table 4–4 Columns 47 - 48 (Restore Application Format)

Entry	Definition
Y or Blank	The data screen is redisplayed. Control is returned to the program along with the command key and any data input by the user before help mode was entered.
N	The data screen is not redisplayed. Control is returned to the program along with the command key. Any data input by the user before help mode was entered is lost.
01 - 99	If the program indicator is on, the data screen is redisplayed. Control is returned to the program. The command key and any data input by the user before help mode was entered is also returned to the program. If the indicator is off, the data screen is not redisplayed. Control is returned to the program along with the command key. Any data input by the user before help mode was entered is lost.

4.13 Column 49 (Reserved)

Column 49 is not used at present. It is reserved for future use.

4.14 Columns 50 - 51 (Boundary Indicator)

By default, once a user has entered help mode, they can scroll through all of the help screens defined within a screen format file using the Roll-Up and Roll-Down keys. Using columns 50 - 51 to set a boundary on a Help specification limits the help screens a user can view. Section 4.16, Help Screen Processing, and Section 4.17, Help Specification Example, describe boundaries in more detail.

Table 4–5 Columns 50 - 51 (Boundary Indicator)

Entry	Definition
N or Blank	This Help specification does not act as a boundary.
Y	This Help specification acts as a boundary.
01 - 99	If the program indicator is on, this Help specification acts as a boundary. If the indicator is off, this Help specification does not act as a boundary.

4.15 Columns 52 - 80 (Reserved)

Columns 52 - 80 are not used at present. They are reserved for future use.

4.16 Help Screen Processing

The naming format used with help screens determines the order in which the screens are processed when the Roll keys are used in help mode. The first six characters in the help format name determine the group to which the help screen belongs. The two numeric digits at the end of the help format name determine the order in which the help screen is displayed while in help mode.

Assume that two groups of help screens are defined for a customer information entry screen. The first group of screens provides help on customer address fields. The second group of screens provides information on customer credit fields. The help screens are named as follows:

- Address Help Screens: ADDRSS10, ADDRSS20, ADDRSS30, ADDRSS40
- Credit Help Screens: CREDIT10, CREDIT20, CREDIT30

When the screen format file is compiled, the help screens are stored in the following sequence: ADDRSS10, ADDRSS20, ADDRSS30, ADDRSS40, CREDIT10, CREDIT20, CREDIT30.

Assume that the user is in the customer city field and presses the Help key. This brings up the ADDRSS30 help screen. If the user then presses the Roll-Down key, the ADDRSS40 help screen is displayed. If the user presses the Roll-Down key again and boundaries are in place, the ADDRSS10 help screen is displayed. If boundaries are not in place, the CREDIT10 help screen is displayed.

Once the user enters help mode, the screens are displayed in sequential order by format name. The boundary setting in columns 50 - 51 determine whether the user can view more than one group of help screens for a given help area.

4.17 Help Specification Example

The following example shows a Help specification defining the Help screen HLPTXT04 for the ADD screen. This example can be found in the TEMPATE_MAINTFM.FRM screen format file supplied with the Migration RPG Compiler Kit. The Help specification example is followed by the definition of the HLPTXT04 help screen.

Example 4–1 Help Specification Example

```
         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
      *
     SADD        124      YY                                    AEFG
     HHLPTXT04                     1 1 2480       Y
     D          66 1 1Y               Y                CFILE MAINTENANCE UTILITX
     DY - ADD RECORD
   .
   .
   .
     SHLPTXT04   124      YY
     D          23 1 1Y               Y                CADD RECORD SCREEN
     D          79 3 1Y                                 CThe Add Record screen aX
     Dllows new records to be added to the file.  Fill in the
     D          79 4 1Y                                 Cdata fields and then prX
     Dess the [ENTER/REC ADV] key ([F4], [PF4], or [-] on
     D          79 5 1Y                                 CPC key pad) to save theX
     D changes.  If you would like to leave the record
     D          79 6 1Y                                 Cwithout saving any chanX
     Dges, enter [CMD1].
     D          79 8 1Y               Y                CCOMMAND & FUNCTION KEYS
     D          79 9 1Y                                 C [ENTER/REC ADV] Save cX
     Dhanges, return to key screen
     D          7910 1Y                                 C [CMD1]          Quit rX
     Decord without saving changes, return to key screen
     D          7911 1Y                                 C [CMD5]          SwitchX
     D to Find mode, do not save changes
     D          7912 1Y                                 C [CMD7]          Exit pX
     Drogram and return to menu, do not save changes
     D          7923 1Y                                 CPress [Page Up] or <PagX
     De Down] to view more help text and keypad layouts.
     D          7924 1Y               Y                CPress [ENTER/REC ADV], X
     D[F4], or [PF4] to exit help screen.
```

- The HLPTXT04 help screen name indicates that the help screen is part of the HLPTXT group of help screens.
- The entries in columns 34 - 42 of the Help specification indicate that the HLPTXT04 screen can be called from anywhere within the ADD screen.
- The Y in column 50 of the Help specification indicates that the HLPTXT04 help specification is a boundary specification. Help screens outside of the HLPTXT group cannot be viewed using the Roll keys when this screen is accessed.

5 Description Specification

The Description specifications follow the Screen specification and optional Help specifications. Each screen format must contain at least one Description specification.

Each Description specification completely defines a field for the screen format. The specification provides the field position on the screen, field attributes, field length, and other information about each field in the screen format. A Description specification can be used to control how and when a field is displayed and when it will accept input.

5.1 Columns 1 - 5 (Line Number)

Columns 1 - 5 can be blank, specify a line number to assist in sequencing the screen specifications, or can be used as a comment. Columns 1 - 5 are ignored by the compiler.

5.2 Column 6 (Specification Identification)

Column 6 must always contain a D to identify a Description specification.

5.3 Column 7 (Comment)

An asterisk (*) in column 7 indicates that the line is a comment line. The line will be ignored by the compiler. Column 6 can be blank on a comment line.

5.4 Columns 7 - 14 (Field Name)

The field name is optional and does not need to be entered.

Columns 7 - 14 can contain the name of an input or output field. The field name has a maximum of 8 characters with no embedded blanks. The name must be left-justified, starting in position 7. The field name may contain any combination of alphanumeric characters along with the $, #, and @ characters. The name must begin with an alphabetic character (A - Z) or the characters #, $, or @. The field name is not required and is treated as a comment by the compiler. It is provided for documentation purposes at present, but may be used for cross-reference or program generation purposes in the future.

5.5 Columns 15 - 18 (Field Length)

Columns 15 - 18 are used to define the length of the field described by the Description specification. The field length can be from 1 to 1919. The field length must be right-justified; leading zeros are not required.

Signed numeric fields should be defined one digit larger than they are defined within the RPG program to accommodate display of the sign character.

Fields defining MIC message members for display must be a minimum of 6 characters in length. See Section 5.31.1, Using MIC Message Members in Output Fields, for more information on using MIC message members in output fields.

5.6 Columns 19 - 20 (Line Number or Row)

Columns 19 - 20 define the line number or row that the field will start on. The field can contain 01 - 48. The field must be right-justified; a leading zero is not required.

The line number is an offset from the start line number, which is defined in columns 17 - 18 of the Screen specification. The actual line number can be determined using the following formula:

```
actual line = starting line (S spec) + line number (D spec) - 1
```

As an example, a starting line number of 3 and a line number of 5 gives an actual line number of 7 = (3 + 5 - 1).

The line number cannot exceed the number of lines on the screen. In the above example, the maximum line number would be 46. The overall maximum line number is 48.

5.7 Columns 21 - 22 (Horizontal Position or Column)

Columns 21 - 22 define the horizontal position or column in which the field starts. The field can contain 01 - 80 or 01 - D2, depending on whether the screen is defined as an 80 or 132 column display. The field must be right-justified; a leading zero is not required. Two digit values representing positions 100 to 132 are listed in the following table:

Note: When designing screen formats, it is recommended that the screen width on all screens used by the program be the same, either 80 column or 132 column. Switching screen widths between formats within a program changes the display characteristics of the display terminal and may produce undesireable results. Maintaining a consistent display width produces consistent output.

Table 5–1 Two Digit Representation of 100 to 132

Column	Entry	Column	Entry
100	A0	117	B7
101	A1	118	B8
102	A2	119	B9
103	A3	120	C0
104	A4	121	C1
105	A5	122	C2
106	A6	123	C3
107	A7	124	C4
108	A8	125	C5
109	A9	126	C6
110	B0	127	C7
111	B1	128	C8
112	B2	129	C9
113	B3	130	D0
114	B4	131	D1
115	B5	132	D2
116	B6		

Note: Display fields cannot be overlaid within a screen format. Overlaid fields within a format will generate a compilation error message.

5.8 Columns 23 - 24 (Output Data)

Columns 23 - 24 determine if the field will be output to the screen. Combining the output entry with a Y in column 26 defines an input/output field. An entry must be present in the Output Data field to display program data, constant data, and MIC message member data.

Table 5–2 Columns 23 - 24 (Output Data)

Entry	Definition
N or blank	No output.
Y	Output based on entries in other columns of the Description specification. Output options are described in Table 5–3, Output Control Based on Y in Columns 23 - 24.
01 - 99	Output based on entries in other columns of the Description specification. Output options are described in Table 5–4, Output Control Based on Indicator in Columns 23 - 24 .

Table 5–3 Output Control Based on Y in Columns 23 - 24

Output Data Col 23 - 24	Constant Type Col 56	Constant Data Col 57 - 79	Result - Data displayed
Y	Blank	Blank	Output data is supplied by program. Pointer in program output record buffer is moved the length defined in columns 15 - 18.
Y	Blank	Data	Data in columns 57 - 79 is displayed.
Y	C	Blank	Blanks are displayed.
Y	C	Data	Data in columns 57 - 79 is displayed.
Y	M	Blank	The Runtime System looks for a MIC message member number in the program output record. A MIC message member number is a 6-character field. If the output record contains blanks or the MIC number is invalid, blanks are displayed. If the output record contains a valid MIC number, the associated MIC message is displayed. In either case, the pointer in the program output buffer will be advanced 6 characters. See Section 5.31.1, Using MIC Message Members in Output Fields, for more information on using MIC message members in output fields.
Y	M	Data	The MIC message identified by the MIC number in columns 57 - 62 is displayed. If the MIC number is invalid, blanks are displayed. See Section 5.31.1, Using MIC Message Members in Output Fields, for more information on using MIC message members in output fields.

Table 5–4 Output Control Based on Indicator in Columns 23 - 24

Output Data Col 23 - 24	Constant Type Col 56	Constant Data Col 57 - 79	Result - Data displayed
01 - 99	Blank	Blank	If Output Data indicator is on, output data is supplied by program. If Output Data indicator is off, blanks are displayed. In either case, the pointer in program output record buffer is moved the length defined in columns 15 - 18.
01 - 99	Blank	Data	If Output Data indicator is on, data in columns 57 - 79 is displayed. If Output Data indicator is off, blanks are displayed.
01 - 99	C	Blank	This combination is not allowed.
01 - 99	C	Data	This combination is not allowed.
01 - 99	M	Blank	If Output Data indicator is on, the MIC message member specified by the program is displayed. If Output Data indicator is off, blanks are displayed. See Section 5.31.1, Using MIC Message Members in Output Fields, for more information on using MIC message members in output fields.
01 - 99	M	Data	If Output Data indicator is on, the MIC message member specified by the program is displayed. If Output Data indicator is off, the MIC message identified by the MIC number in columns 57 - 62 is displayed. See Section 5.31.1, Using MIC Message Members in Output Fields, for more information on using MIC message members in output fields.

If the field is defined as an input/output field, data within the field can be modified by the user. If the field is defined as an output only field, the field contents cannot be modified by the user.

If an override is in progress, field output is controlled by an indicator, and the indicator is on, then data supplied by the user program is displayed in the field. If the field output indicator is off, the field contents remain unchanged. See Section 3.16, Columns 33 - 34 (Override Fields), for a complete description of the override operation.

5.9 Column 25 (Reserved)

Column 25 is not used at present. It is reserved for future use.

5.10 Column 26 (Input Data)

Column 26 determines if a field will accept input. When combined with columns 23 - 24, a field can be defined as an input/output field. This column determines whether a field is allowed to accept input. Each input field writes data to the program input record.

Table 5–5 Column 26 (Input Data)

Entry	Definition
N or Blank	The field will not accept input.
Y	The field will accept input.

Input can be provided to an input field via data entered by the user or as a constant. A constant input field is defined by entering a Y in column 26, a C in column 56, and the constant data in columns 57 - 79.

5.11 Column 27 (Data Type)

Column 27 specifies the type of data that can be input into the field. Data typing can be used to provide rudimentary edit checking of input data. With the exception of the signed numeric data type, entries in the Data Type field are ignored for output fields.

Table 5–6 Column 27 (Data Type)

Entry	Definition
A	Alphabetic data type: The field can contain only alphabetic characters (A - Z). This is a useful data type for names.
Blank or B	Alphanumeric data type: The field can contain alphameric data. No edit checking is done on the field's contents. This is a good data type to use for general data such as customer address information.
D	Decimal data type: The field can contain only the digits 0 - 9. This data type is useful with fields like dates and amounts.
N	Numeric data type: The field can contain only numeric data, including 0 - 9, commas, periods, and plus and minus signs. This data type can be used for numeric fields such as pricing information.
	Note: Numeric fields within RPG do not use embedded commas or decimal points. If a numeric field in a screen format permits this type of data to be entered, a parsing routine will need to be implemented within the program to extract the formatting characters before the data is placed in a numeric field within the program.
S	Signed numeric data type: Only the digits 0 - 9 can be entered into a signed numeric field. A signed numeric field must be a minimum of 2 digits long. Signed numeric fields used with RPG can be from 2 - 16 digits long. The last position of the field is reserved for the sign. This means the field length in the screen (columns 15 - 18) must be one larger than the actual field size in the program. For example, a 6-digit numeric field displayed on a screen with a signed numeric data type would be defined within the field Description specification with a length of 7. To enter a negative value in a signed numeric field, the user presses the Field - key. The Field - is defined as Cmd Enter in Migration RPG.
Z	Right-to-left data type: The cursor moves from right to left within the input field. This data type is useful when writing software for cultures that use a language that is read and written right to left. Column 40 in the Screen specification controls cursor movement between fields within a screen format.

5.12 Column 28 (Mandatory Fill)

Column 28 specifies that if at least one character is entered in the field, the entire field must be filled before it can be entered and the user can move on to the next field. Filling the field with spaces is valid.

Table 5–7 Column 28 (Mandatory Fill)

Entry	Definition
N or Blank	Field is not required to be filled.
Y	If one character is entered into the field, the entire field must be filled.

5.13 Column 29 (Mandatory Entry)

Column 29 specifies that data must be entered in the field before control can be returned to the program. The only way control can be returned to the program without entering data in a mandatory entry field is if no data has been entered in any input field on the screen.

Table 5–8 Column 29 (Mandatory Entry)

Entry	Definition
N or Blank	Field does not require that data be entered.
Y	Field requires data. The user must enter at least 1 character of data into this field before control can be returned to the program.

5.14 Column 30 (Self-Check)

Column 30 can be used to specify modulus 10 or modulus 11 self-checking for the field. If self-checking is specified, the rightmost digit in the field is used as the self-check digit (0 - 9). Self-checking applies to input capable fields only.

If a self-check fails, the cursor remains in the field. The user must enter a valid value before the program will continue.

Table 5–9 Column 31 (Self-Check)

Entry	Definition
Blank	No self-checking is done.
E	Modulus 11 self-checking is done.
T	Modulus 10 self-checking is done.

A right-to-left field (data type Z in column 27) can be a self-check field. However, the rightmost digit of the field is still used as the check digit. Thus, in a right-to-left field, the first character entered is the check digit.

Section 5.33, Using Self-Check Fields, contains more information on how to use self-check fields.

5.15 Column 31 (Adjust/Fill)

Column 31 specifies how a field is adjusted and filled after it is entered. If Adjust/Fill is specified, a field can be right-adjusted and zeros or blanks placed in the unused portion of the field. Adjust/Fill entries only affect input capable fields.

Table 5–10 Column 31 (Adjust/Fill)

Entry	Definition
Blank	Signed numeric fields are right adjusted and blank filled. No adjust or fill is used for any other field type.
Z	The field is right adjusted and zero filled.
B	The field is right adjusted and blank filled.

5.16 Columns 32 - 33 (Position Cursor)

Normally the cursor is positioned in the leftmost, top-most, input capable field on a screen (rightmost, top-most in the case of a right-to-left display). This default cursor position is referred to as the home position on the screen. Columns 32 - 33 specify that the cursor should be positioned at the beginning of this field, rather than the home position, when the screen is displayed. Cursor control is useful when attempting to draw a user's attention to a field. For example, if a data entry error occurs, a program indicator can be used to position the cursor in the field containing the invalid data. The same indicator can be used to highlight the field, further drawing the user's attention. If more than one field in a display has cursor control enabled, the cursor will be placed in the field closest to the home position.

Table 5–11 Columns 32 - 33 (Position Cursor)

Entry	Definition
N or Blank	The cursor will only be positioned in the field if the field is input capable, occupies the home position, and no other field in the display has Position Cursor enabled.
Y	The cursor will be positioned in this field unless another field also has Position Cursor enabled. If more than one field have position cursor enabled, the field closest to the home position will contain the cursor.
01 - 99	If the indicator is on, the cursor will be positioned in this field unless another field also has Position Cursor enabled. If more than one field have position cursor enabled, the field closest to the home position will contain the cursor. If the indicator is off, the entry is treated like a blank or N.

5.17 Column 34 (Enable Dup)

Column 34 controls whether the DUP key can be used in the field. The DUP key fills the field from the current cursor position to the end of the field with the DUP character. The DUP character is displayed on the screen as an asterisk (*) and recorded in the field as a hex 1C.

The purpose of the DUP key is to permit fields to be easily duplicated as records are entered. Once an initial value has been placed in a field, the user can press the DUP key in subsequent records to signify that the field contents are to be duplicated. However, field duplication is not an automatic process. When control returns to the program, it must check the field for DUP characters and make the appropriate replacements.

Table 5–12 Column 34 (Enable Dup)

Entry	Definition
N or Blank	The DUP key is not available for the field.
Y	The DUP key is available for the field.

5.18 Column 35 (Controlled Field Exit)

Column 35 specifies that a valid field termination key must be pressed to exit the field. Valid field termination keys include field exit keys, the DUP key, the Record Enter/Adv key, command keys, and function keys. See Section 1.3.1, Workstation Key Assignments, for a complete description of Migration RPG key definitions.

Specifying a controlled field exit prevents the cursor from automatically moving to the next field when the current field has been filled. This can be useful when it is desirable to have the user explicitly enter the field. It is also useful when paired with the Record Enter/Advance option. When the Controlled Field Exit and Record Enter/Advance option are used together, the record is automatically entered and control is returned to the program when the user presses a field exit key to enter the field.

Table 5–13 Column 35 (Controlled Field Exit)

Entry	Definition
N or Blank	The cursor will advance to the next field when this field is filled with data.
Y	The cursor will only leave this field when one of the field termination keys are pressed.

5.19 Column 36 (Auto Record Enter/Advance)

Column 36 can be used to specify that entering the field also serves to enter the record, returning control to the program. The Auto Record Enter/Advance option treats field termination like the entry of the Record Enter/Adv key.

The Record Enter/Advance option is useful when it is desirable to have the program automatically take action when a field is filled. The option can be paired with the Controlled Field Advance option to force the user to press a field exit key to terminate the field. If the Controlled Field Advance option is not used, filling the field will automatically enter the record.

Table 5–14 Column 36 (Auto Record Enter/Advance)

Entry	Definition
N or Blank	Control is not returned to the program when the field is entered.
Y	Control is returned to the program when the field is entered.

5.20 Columns 37 - 38 (Protect Field)

Columns 37 - 38 specify that an input capable field is to be protected from user input. When field protection is enabled, the cursor skips the protected field.

If an override operation is in effect, the protect field indicator is ignored.

Table 5–15 Columns 37 - 38 (Protect Field)

Entry	Definition
N or Blank	The field is not protected from input.
Y	The field is always protected from input.
01 - 99	If the program indicator is on, the field is protected from input. If the indicator is off, the field is not protected from input.

5.21 Columns 39 - 40 (High Intensity)

Columns 39 - 40 specify that the field be displayed with high intensity or in bold type. Some terminal emulators may use colors instead of bolding to emphasize a field.

Table 5–16 Columns 39 - 40 (High Intensity)

Entry	Definition
N or Blank	Normal intensity is always used to display the field.
Y	High intensity is always used to display the field.
01 - 99	If the program indicator is on, high intensity is used to display the field. If the indicator is not on, normal intensity is used to display the field.

5.22 Columns 41 - 42 (Blink Field)

Columns 41 - 42 specify that the field be displayed as blinking. Some terminal emulators may use colors instead of blinking to emphasize a field.

Table 5–17 Columns 41 - 42 (Blink Field)

Entry	Definition
N or Blank	The field never blinks when displayed.
Y	The field always blinks when displayed.
01 - 99	If the program indicator is on, the field blinks when displayed. If the indicator is not on, the field does not blink when displayed.

5.23 Columns 43 - 44 (Nondisplay Field)

Columns 43 - 44 specify that the contents of the field do not appear on the display. This option is useful when building displays that request passwords or other sensitive data.

Table 5–18 Columns 43 - 44 (Nondisplay Field)

Entry	Definition
N or Blank	The field contents are always displayed.
Y	The field contents are never displayed.
01 - 99	If the program indicator is on, the field contents are not displayed. If the indicator is not on, the field contents are displayed.

5.24 Columns 45 - 46 (Reverse Image)

Columns 45 - 46 specify that the field be displayed in reverse image. Some terminal emulators may use colors instead of reverse image to emphasize a field.

Table 5–19 Columns 45 - 46 (Reverse Image)

Entry	Definition
N or Blank	The field is never displayed with reverse image.
Y	The field is always displayed with reverse image.
01 - 99	If the program indicator is on, the field is displayed with reverse image. If the indicator is not on, the field is not displayed with reverse image.

5.25 Columns 47 - 48 (Underline)

Columns 47 - 48 specify that the field be underlined when displayed. Some terminal emulators may use colors instead of underlining to emphasize a field.

Table 5–20 Columns 47 - 48 (Underline)

Entry	Definition
N or Blank	The field is never underlined when displayed.
Y	The field is always underlined when displayed.
01 - 99	If the program indicator is on, the field is underlined when displayed. If the indicator is not on, the field is not underlined when displayed.

5.26 Column 49 (Column Indicators)

Column 49 specifies that a field should contain column indicators when displayed. When column indicators are specified, an underscore character (_) appears in each *blank* character position of the field. Column indicators are not transferred to the program input record buffer when the field in entered.

Table 5–21 Column 49 (Column Indicators)

Entry	Definition
N or Blank	The field does not display column indicators.
Y	Column indicators appear in each blank position of the field.

5.27 Column 50 (Reserved)

Column 50 is not used at present. It is reserved for future use.

5.28 Column 51 (Lowercase)

Column 51 controls the entry of lowercase characters (a - z) into an input capable field. If lowercase entry is not enabled, lowercase characters typed into a field are automatically converted to uppercase when the field is entered.

The Lowercase option on the Description specification overrides the Lowercase option in column 21 of the Screen specification.

Table 5–22 Column 51 (Lowercase)

Entry	Definition
Blank	Field case will be controlled by the lowercase Screen specification entry in column 21.
N	Lowercase characters typed into the field will be converted to uppercase when the field is entered.
Y	The field will accept lowercase characters.

5.29 Columns 52 - 55 (Reserved)

Columns 52 - 55 are not used at present. They are reserved for future use.

5.30 Column 56 (Constant Type)

Column 56 specifies the type of data that is to be displayed in an output or input/output field. Constant data can be coded directly into the Description specification in columns 57 - 79 or it can be obtained from a MIC message member.

Table 5–23 Column 56 (Constant Type)

Entry	Definition
Blank	If columns 57 - 79 contain data, it is displayed. If columns 57 - 79 are blank, data from the program output record is displayed.
C	The data in columns 57 - 79 is displayed. Use the C option if blanks are to be displayed in the field.
M	A MIC message member is displayed. The message member code can be placed in columns 57 - 79 or provided by the program. See Section 5.31.1, Using MIC Message Members in Output Fields, for more information on using MIC message members in output fields.

5.31 Columns 57 - 79 (Constant Data)

Columns 57 - 79 specify information about the data to be placed in an output or input/output field. Data in columns 57 - 79 and the Constant Type specified in column 56 determine how the data is interpreted.

Table 5–24 Columns 57 - 79 (Constant Data)

Constant Type	Constant Data	Result
C	Data	Constant data appears in the field.
C	Blanks	Blanks appears in the field.
Blank	Data	Constant data appears in the field.
Blank	Blanks	Data from program output record appears in the field.
M	Data	The entry in the Constant Data field should be a 6-character, MIC message member number. The Migration RPG Runtime System uses the MIC number to locate and display the associated MIC message member. See Section 5.31.1, Using MIC Message Members in Output Fields, for more information on using MIC message members in output fields.
M	Blanks	The Migration RPG Runtime System will expect to find a 6-character, MIC message member number in the program output record. The Runtime System uses the MIC number to locate and display the associated MIC message member. See Section 5.31.1, Using MIC Message Members in Output Fields, for more information on using MIC message members in output fields.

5.31.1 Using MIC Message Members in Output Fields

MIC message member output fields are identified by an M in column 56 (Constant Type) of the Description specification. A MIC number consists of 4 digits followed by one of the following Message ID codes:

Table 5–25 MIC Message Member ID Codes

Entry	Definition
U1	User Level 1 message member.
U2	User Level 2 message member.
S1	System Level 1 message member.
S2	System Level 2 message member.

If the Message ID code is left blank, the User Level 1 message member (U1) is assumed.

The amount of the MIC message that is displayed is determined by the field length entry in columns 15 - 18. If the field length is shorter than the MIC message, the message text is truncated. If the field length is longer than the message text, the field is padded with blanks.

When the MIC number is obtained from the program output record, only 6 characters need to be reserved in the buffer for the number. The Runtime System will always advance 6 characters in the program buffer when it encounters a program-supplied MIC number, regardless of the field length specified in the Description specification.

Information on creating and maintaining MIC Message Member files is available in Chapter 12 of the *Migration RPG User's Guide, Maintaining Message Files (RPGMSG Utility).*

5.32 Column 80 (Continuation)

Column 80 allows constant data to be continued on the next line. If the constant data placed in column 57 - 79 exceeds 23 characters, place an X in column 80 and continue the constant data in column 7 of the next line. Constants can be continued over multiple lines using the Continuation option.

Example 5–1 Constant Using Continuation Option

```
         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
      *
     D           37 1 1Y               Y                CFILE MAINTENANCE UTILITX
     DY - ADD RECORD
```

This example shows a constant output field that is 37 characters in length. The constant is too long to fit in columns 57 - 79, so column 80 is marked with an X, enabling the Continuation option. The remainder of the constant data appears on the next Description specification starting in position 7.

5.33 Using Self-Check Fields

Self-check fields are defined by placing a Y in column 25 (Input) and a T or E in column 30 (Self-Check) of an input field Description specification. A self-check field provides a means to validate special codes like account numbers.

A self-check field works by running the contents of a field through a arithmetic formula, then comparing the results to a check digit. The check digit will be a value of 0 - 9. The check digit is always the rightmost digit in the field and is not used in the self-check calculations.

Self-check fields are generally used with numbers which have been generated in advance so that their self-check digits will be valid. For example, a bank assigning account numbers that can pass the modulus 10 self-check would choose the numbers from a generated list of modulus 10 numbers, thus ensuring that they assign valid account numbers to new accounts.

Migration RPG provides modulus 10 and modulus 11 self-check algorithms. Modulus 10 self-checking is specified by entering a T in column 30 of the input field Description specification. Modulus 11 self-checking is specified by entering an E in column 30. The following sections describe how these self-check algorithms work.

5.33.1 Modulus 10 Self-Check

The following steps are used to determine the modulus 10 check digit. The rightmost digit in the input string is not used in the calculations. It is considered the check digit and is compared to the results of the calculations. The field 134825 will be used as an example.

1 Disregard the rightmost digit, 5, leaving 13482.

```
13482
```

2 Multiply the left most digit, 1, and each alternate digit, 4 and 2, by 2, giving the results 2, 8, 4. Add these digits together:

```
2 + 8 + 4 = 14
```

3 Add in the remaining digits, 3 and 8:

```
14 + 3 + 8 = 25
```

4 Subtract the result, 25, from the next highest number ending in zero, in this case 30. The result is the calculated check digit:

```
30 - 25 = 5
```

The calculated check digit is compared to the check digit at the end of the string. If they match, the field is valid.

5.33.2 Modulus 11 Self-Check

The following steps are used to determine the modulus 11 check digit. The rightmost digit in the input string is not used in the calculations. It is considered the check digit and is compared to the results of the calculations. The field 8940711209 will be used as an example.

1 Disregard the rightmost digit, 9, leaving 894071120.

2 Assign each digit in the number a weighting factor in the sequence 2, 3, 4, 5, 6, 7, 2, 3, 4, 5, 6, 7, 2, 3, and so on. Work from right to left. In the example, 894071120 would be assigned the weighting factor 432765432.

```
8 9 4 0 7 1 1 2 0
4 3 2 7 6 5 4 3 2
```

3 Multiply each digit by its weighting factor:

```
 8  9  4  0  7  1  1  2  0
 4  3  2  7  6  5  4  3  2
-- -- -- -- -- -- -- -- --
32 27  8  0 42  5  4  6  0
```

4 Add the results of the digits multiplied by their weighting factor:

```
32 + 27  + 8  + 0 + 42  + 5  + 4  + 6  + 0 = 124
```

5 Divide the result by 11.

```
124/11 = 11; Remainder = 2
```

6 Subtract the remainder from 11:

```
11 - 2 = 9
```

The result, 9, is the calculated check digit.

Note: If the remainder is 1, the number does not have a check digit. Be sure numbers generated for modulus 11 self-checks do not have a remainder of 1. Valid remainders are 0 and 2 - 9.

The calculated check digit is compared to the check digit at the end of the string. If they match, the field is valid.

6 WORKSTN File Processing

WORKSTN files can be defined as combine primary or combine demand files. If a WORKSTN file is defined as the primary file, no secondary files are allowed in the program.

6.1 Primary WORKSTN File

Primary WORKSTN files are processed very much like a primary update DISK file. A screen is output during the output phase of the RPG logic cycle. Input is requested from the screen during the input phase of the RPG logic cycle.

Care must be taken during program startup with a primary WORKSTN file. If no screen is displayed during the first cycle of the RPG program startup phase, then a blank record is returned from the WORKSTN device during first cycle input. If a WORKSTN screen is displayed during the output phase of program startup, then input is obtained from the WORKSTN screen during first cycle input. When defining a WORKSTN device as the primary file, it is customary to display a WORKSTN screen during program startup using the first page indicator (1P).

Example 6–1 WORKSTN Combine Primary Output Specifications

```
         1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901234567890123456789
      *
     OWORKSTN D         1P
     O       OR         71
     O                                      K5 'SCRNA'
     O                           UDATE Y    10
     O                           TYPE       11
     O                  98       MSG,E      86
     O                                      93 'PRT-OFF'
     O                           WWWAR#     99
     O                           WWSPLT    102
```

This example shows WORKSTN file Output specifications for a combine primary file. Note the use of the 1P indicator to output this screen during program startup.

After the first cycle, all primary input is assumed to be coming from the WORKSTN device. Screen output can be done using the *EXCPT* opcode or during the output phase of each program cycle.

6.2 Demand WORKSTN File

It is a common programming practice to define WORKSTN files as combine demand files. This gives the programmer more control over when screens are displayed and input is retrieved. Screen output is done using the *EXCPT* opcode and screen input is retrieved using the *READ* opcode.

When using a demand WORKSTN file, a screen must always be output before input can be retrieved. If a *READ* operation is done against a WORKSTN file without first displaying a screen, a blank record will be returned.

Example 6–2 WORKSTN Demand File Calculation Specifications

```
         1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901234567890123456789
      *
     C                     EXCPTKEY
     C                     READ WORKSTN                90
```

This example shows WORKSTN demand file Calculation specifications. Note that the EXCPT operation appears before the READ operation to ensure that a screen is being displayed from which to retrieve input.

The indicator in columns 56 - 57 turns on if the WORKSTN read is terminated by an exception. An exception occurs when the user presses one of the function keys (Roll-Up, Roll-Down, Clear, Print, Home, and Help).

6.3 WORKSTN End-of-File

To indicate end-of-file on a WORKSTN file, the last-record (LR) indicator must be set on.

7 WORKSTN Command and Function Keys

Command and function keys provide a way for the user to interact with an interactive WORKSTN program. Command and function keys terminate entry on the current input screen and return control to the program. The program can determine which command or function key was entered by the user and take action based on the information.

7.1 Command Keys

This section describes how to define and use command keys and command key indicators. The following table defines all of the Migration RPG command keys, the keystrokes used to execute each command key, and each command key's associated K indicator:

Table 7–1 Command Keys

Command Key	Keys to Press	K Indicator	Key Mask Entry
Cmd1	PF1 1	KA	A
Cmd2	PF1 2	KB	B
Cmd3	PF1 3	KC	C
Cmd4	PF1 4	KD	D
Cmd5	PF1 5	KE	E
Cmd6	PF1 6	KF	F
Cmd7	PF1 7	KG	G
Cmd8	PF1 8	KH	H
Cmd9	PF1 9	KI	I
Cmd10	PF1 0	KJ	J
Cmd11	PF1 -	KK	K
Cmd12	PF1 =	KL	L
Cmd13	PF1 !	KM	M
Cmd14	PF1 @	KN	N
Cmd15	PF1 #	KP	P
Cmd16	PF1 $	KQ	Q
Cmd17	PF1 %	KR	R
Cmd18	PF1 ^	KS	S
Cmd19	PF1 &	KT	T
Cmd20	PF1 *	KU	U
Cmd21	PF1 (	KV	V
Cmd22	PF1)	KW	W
Cmd23	PF1 _	KX	X
Cmd24	PF1 +	KY	Y

7.1.1 Defining Command Keys

The command keys that can be used in a program are defined in the Screen specification. Chapter 3, Screen Specification, describes the Screen specification in detail. A programmer has the option of making all command keys available, no command keys available, or selected command keys available. The following examples show how this is accomplished.

Example 7–1 Defining all command keys as enabled

```
         1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901234567890123456789
      *
     SKEY        124
```

In this example, column 28 in the S specification is left blank. This indicates that all command keys are enabled for the KEY screen.

Example 7–2 Defining all command keys as disabled

```
         1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901234567890123456789
      *
     SKEY        124       Y
```

In this example, column 28 in the S specification is Y. This indicates that only the command keys defined in the key mask in columns 64 - 79 are enabled. Since no keys are defined in the key mask, no command keys are enabled for the KEY screen.

Example 7–3 Defining selected command keys as enabled

```
         1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901234567890123456789
      *
     SKEY        124       Y                                       DEL
```

In this example, column 28 in the S specification is Y. This indicates that only the command keys defined in the key mask in columns 64 - 79 are enabled. Command keys 4, 5, and 12 are enabled for the KEY screen.

Note: Command keys are represented in the key mask on the Screen specification by the character portion of the K indicator. In the previous example, Cmd4 corresponds to the indicator KD, which corresponds to the key mask value D; Cmd5 = KE = E; Cmd12 = KL = L.

7.1.2 Command Key Indicators

Command keys are associated to K indicators. Migration RPG provides 24 K indicators: KA - KN, KP - KY. These can be used as general purpose indicators within an RPG program. When a command key is used to exit a screen, the associated K indicator is turned on. All other K indicators are turned off. For example, if the user enters Cmd7, the indicator KG is turned on in the program and all other K indicators are turned off.

7.1.3 Command Keys and the INFDS Data Structure

If an INFDS data structure is defined in a WORKSTN program, the *STATUS field will be set to 00002 when a screen is terminated using a command key. The INFDS data structure is discussed in more detail in Chapter 8, INFDS Data Structure.

7.2 Function Keys

This section describes how to define and use function keys. Function keys terminate a screen and return control to the program by creating an exception condition. This condition can be trapped in one of two ways: by specifying an indicator in columns 56 - 57 of a *READ* operation or by using an INFSR subroutine.

The INFDS data structure and *STATUS field must be used to determine the specific function key entered if more than one function key is enabled on a screen.

Function keys are defined in the Screen specification key mask using the numbers 1 - 6. The following table defines each function key, its key strokes, and its key mask definition:

Table 7–2 Function Keys

Function	Function Key	Key Mask Entry
Print	PF1+P	1
Roll-Up	PF1+U or Next Screen	2
Roll-Down	PF1+D or Prev Screen	3
Clear	PF1+C or	4
Help	PF1+H or PF2 or Help	5
Home	PF1+T	6

7.2.1 Defining Function Keys

The function keys that can be used in a program are defined in the Screen specification. Chapter 3, Screen Specification, describes the Screen specification in detail. A programmer has the option of making all function keys available, no function keys available, or selected function keys available. The following examples show how this is accomplished.

Example 7–4 Defining all function keys as enabled

```
          1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901234567890123456789
      *
     SKEY        124
```

In this example, column 27 in the S specification is left blank. This indicates that all function keys are enabled for the KEY screen.

Example 7–5 Defining all function keys as disabled

```
          1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901234567890123456789
      *
     SKEY         124       Y
```

In this example, column 27 in the S specification is Y. This indicates that only the function keys defined in the key mask in columns 64 - 79 are enabled. Since no keys are defined in the key mask, no function keys are enabled for the KEY screen.

Example 7–6 Defining selected function keys as enabled

```
          1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901234567890123456789
      *
     SKEY         124       Y                                      23
```

In this example, column 27 in the S specification is Y. This indicates that only the function keys defined in the key mask in columns 64 - 79 are enabled. The Roll-Up and Roll-Down function keys are enabled for the KEY screen.

7.2.2 Function Keys and the INFDS Data Structure

If an INFDS data structure is defined in a WORKSTN program, the *STATUS field will be set to one of the following values when a screen is terminated using a function key:

Table 7–3 INFDS Function Key Codes

Code	Definition
01121	Print key was pressed
01122	Roll-Up key was pressed
01123	Roll-Down key was pressed
01124	Clear key was pressed
01125	Help key was pressed
01126	Home key was pressed

The INFDS data structure is discussed in more detail in Chapter 8, INFDS Data Structure. The INFDS data structure and the *STATUS field must be used to identify the function entered on screens which define more than one function key.

8 INFDS Data Structure

The INFDS data structure is used to obtain return information from a WORKSTN device. The *STATUS field in the INFDS data structure can be used to determine if a function key, command key, or the Enter/Record Adv key was used to exit a screen.

The Migration RPG INFDS data structure implementation includes limited compatibility with the IBM RPG II INFDS function. This is discussed further in Section 8.1.2, INFDS IBM Compatibility.

8.1 Coding the INFDS Data Structure

The INFDS data structure is coded using a continuation line following the WORKSTN file definition in the File specifications and a data structure definition in the Input specifications.

Example 8–1 INFDS Data Structure Coding Example

```
         1         2         3         4         5         6
1234567890123456789012345678901234567890123456789012345678901234567890123456789
      *
     FWORKSTN CD  F    1300            WORKSTN
     F                                              KINFDS $INFDS
   .
   .
   .
      * INFDS Data Structure
      *
     I$INFDS      DS
     I                                      *STATUS  $STAT
```

- The continuation line following the WORKSTN device definition defines the INFDS data structure $INFDS.
- The INFDS definition is completed with the $INFDS data structure definition in the Input specifications.

Note: The INFDS data structure is not a general-purpose data structure. Fields other than the *STATUS field should not be defined within the INFDS data structure.

8.1.1 *STATUS Keyword

The *STATUS keyword identifies a 5-digit, numeric subfield with zero decimal positions within the INFDS data structure. This subfield contains a code that identifies the method used to exit the last WORKSTN screen. Possible return codes are:

Table 8–1 *STATUS Codes

Code	Definition
00000	Enter/Record Advance key pressed
00002	A command key was pressed
01121	Print key was pressed
01122	Roll-Up key was pressed
01123	Roll-Down key was pressed
01124	Clear key was pressed
01125	Help key was pressed
01126	Home key was pressed
1100	A WORKSTN read operation was attempted before a WORKSTN write operation has taken place.

8.1.2 INFDS IBM Compatibility

Migration RPG provides limited compatibility with the IBM RPG II INFDS data structure. The IBM INFDS data structure provides several additional keywords. These keywords and the data they return are listed in the following subsections.

8.1.2.1 *OPCODE Keyword

The *OPCODE keyword identifies a 5-character, alphanumeric subfield within the INFDS data structure. This subfield contains a value that identifies the WORKSTN I/O operation that most recently completed execution. Possible return values are *READ* or *WRITE*.

8.1.2.2 *RECORD Keyword

The *RECORD keyword identifies an 8-character, alphanumeric subfield within the INFDS data structure. If *OPCODE contains WRITE, then *RECORD contains the name of the screen to which output was written. Otherwise, *RECORD contains blanks.

8.1.2.3 *SIZE

The *SIZE keyword identifies a 4-digit, numeric subfield within the INFDS data structure. The subfield contains the display size. The *SIZE subfield under Migration RPG always returns 1960.

8.1.2.4 *MODE

The *MODE keyword identifies a 2-digit, numeric subfield within the INFDS data structure. The *MODE subfield under Migration RPG always returns 00.

8.1.2.5 *INP

The *INP keyword identifies a 2-digit, numeric subfield within the INFDS data structure. The *INP subfield under Migration RPG always returns 00.

8.1.2.6 *OUT

The *OUT keyword identifies a 2-digit, numeric subfield within the INFDS data structure. The *OUT subfield under Migration RPG always returns 00.

8.1.2.7 Return Code (Positions 23 - 26)

Positions 23 - 26 of an IBM INFDS data structure can be defined to accept a 4-character return code. Migration RPG always places 0000 in these positions of the INFDS data structure with the following two exceptions:

1. If the screen is the first screen displayed for the program, the return code is set to 0100.
2. If a WORKSTN read operation is attempted before a WORKSTN write operation has taken place, the return code is set to 1100.

8.1.2.8 IBM Compatible INFDS Data Structure

The following example shows how an IBM-compatible INFDS data structure is coded in Migration RPG. The * subfields can be coded in any order. The return code field must always be defined in positions 23 - 26.

Example 8–2 IBM Compatible INFDS Data Structure

```
         1         2         3         4         5         6
123456789012345678901234567890123456789012345678901234567890123456789012345
      *
     FTERMINALCD  F     120            WORKSTN
     F                                                KINFDS EXCPTN
   .
   .
   .
     IEXCPTN      DS
     I                                       *STATUS  STATUS
     I                                       *OPCODE  OPCODE
     I                                       *RECORD  RECORD
     I                                       *SIZE    SIZE
     I                                       *MODE    MODE
     I                                       *INP     INP
     I                                       *OUT     OUT
     I                                         23  26 RCODE
```

8.2 Using an INFDS Data Structure

The INFDS data structure is used in conjunction with the resulting indicator in columns 56 - 57 of a *READ* operation or the INFSR subroutine. The INFSR subroutine is described in detail in Chapter 9, INFSR Subroutine. The INFDS data structure is generally used to identify the method used to exit a screen. The data structure must be used to identify function keys if multiple function keys are enabled on a screen.

The following example shows how the INFDS data structure can be used to identify the roll keys. The code used in this example is located in the TEMPLATE_INQ.RPG program included with the Migration RPG Compiler Kit.

Example 8–3 INFDS Example: Roll Key Identification

```
         1         2         3         4         5         6
123456789012345678901234567890123456789012345678901234567890123456789
      *
     FWORKSTN CD  F    1300            WORKSTN
     F                                              KINFDS $INFDS
   .
   .
   .
      * INFDS Data Structure
      *
     I$INFDS      DS
     I                                     *STATUS  $STAT
   .
   .
   .
     C           INQ01     TAG
     C           'INQ01   'DEBUGDEBUG
     C                     EXCPTINQUIR
     C                     MOVE *BLANKS   ERRMSG
     C                     SETOF                    90
     C                     READ WORKSTN               60
      *
      * Check for function keys.
      *
     C  N60                GOTO INQ02
CAS  C           $STAT     CASEQ01123     PRVREC            Roll-Down
     C           $STAT     CASEQ01122     NXTREC            Roll-Up
     C                     CAS            BADFNC
END  C                     END
     C                     GOTO INQ01
   .
   .
   .
```

9 INFSR Subroutine

The INFSR subroutine is used to handle WORKSTN screen exception processing. An exception occurs if an enabled function key is pressed while reading from a combine primary WORKSTN file or while using the *READ* opcode on a combine demand file without a resulting indicator specified in columns 56 - 57.

An INFSR subroutine is written like a normal subroutine within the Calculation specifications using the *BEGSR* and *ENDSR* statements. It can also be called like a normal subroutine during processing using the *EXSR* opcode. Where an INFSR subroutine is called using the *EXSR* opcode, it returns as a normal subroutine. Entries in factor 2 of the INFSR subroutine *ENDSR* statement are ignored.

9.1 INFSR Exception Processing

When an INFSR subroutine is activated by an exception, it first processes the Calculation specifications coded between its *BEGSR* and *ENDSR* statements. When the subroutine has completed operations, it uses the entry in factor 2 of the *ENDSR* statement to determine where to resume program execution. Possible factor 2 entries are:

Table 9–1 INFSR Return Options

ENDSR Factor 2 Entry	Description
*GETIN	Program execution resumes at the beginning of a new cycle.
*DETC	Program execution resumes at the beginning of detail calculations within the same cycle.
*CANCL	Program execution is terminated. Files are closed and the program exits.

9.1.1 Coding INFSR Exception Return Options

Exception return control from an INFSR subroutine is determined by the entry in factor 2 of the INFSR subroutine *ENDSR* statement. Valid entries are *GETIN, *DETC, and *CANCL. The action taken for each one of these entries is listed in Table 9–1, INFSR Return Options.

The INFSR return option can be listed in factor 2 as a literal, field, or array element. Using a field or array element allows the programmer to change where the INFSR subroutine resumes execution based on program parameters.

Example 9–1 INFSR Subroutine Code Example

```
         1         2         3         4         5         6
123456789012345678901234567890123456789012345678901234567890123456789
      *
     FKEYIN   CP  F    1920            WORKSTN
     F                                              KINFDS $INFDS
     F                                              KINFSR $INFSR
   .
   .
   .
     I$INFDS      DS
     I                                    *STATUS  $STAT
   .
   .
   .
     CSR          $INFSR    BEGSR
     CSR          $STAT     COMP 01121                    21 Print
     CSR          $STAT     COMP 01122                    22 Roll-Up
     CSR          $STAT     COMP 01123                    23 Roll-Down
     CSR          $STAT     COMP 01124                    24 Clear
     CSR          $STAT     COMP 01125                    25 Help
     CSR          $STAT     COMP 01126                    26 Home
     CSR                    ENDSR'*DETC'
   .
   .
   .
```

This example depicts the INFSR subroutine and INFDS data structure used together to handle a program exception. When a function key is pressed, program control is turned over to the INFSR subroutine $INFSR. The subroutine used the *STATUS field, $STAT, in the INFDS data structure to determine which function key was used to exit the WORKSTN screen. When the INFSR subroutine completes, the *DETC entry in factor 2 of the INFSR *ENDSR* statement directs program control back to the beginning of detail calculations for the current program cycle.

If the INFSR return option in factor 2 is left blank, the user is prompted to enter one of the return options after an INFSR exception occurs and the INFSR subroutine has executed.

10 WORKSTN Program Examples

This chapter provides examples of RPG WORKSTN programs and screens. The source code and data files used in these examples are provided with the Migration RPG Compiler kit. The files are located in the S3X$EXAMPLES directory. It is strongly recommended that the files be copied to a different location before any experimentation is done with them.

10.1 Combine Demand WORKSTN Program Example

This section provides an example of a simple combine demand WORKSTN program, INVENT.RPG. The program accesses the screens defined in the INVENTFM.FRM screen format file. The following three screen formats are defined for the program:

- PARTNUMB
- UPDATE
- ERROR

The purpose of the program is to allow the user to select parts from an inventory file and display and update the inventory data. The PARTNUMB screen is used to retrieve the three-character part number that is the key into the inventory file. The UPDATE screen is used to display the inventory information if the specified part number is found. The UPDATE screen also allows updates to any inventory field except the part number. The ERROR screen displays an error message if an invalid part number is entered on the PARTNUMB screen. The program is terminated by entering a Cmd7 from any of the screens.

The program uses the following indexed file INVENT.DAT:

Example 10–1 INVENT.DAT Data file

```
         1         2         3
12345678901234567890123456789
-----------------------------
P01NUT    W2RED  0300135
P02BOLT   W2GREEN0880167
P03SCREW  W1BLUE 0180048
P04SCREW  W2RED  0150143
P05CAM    W1BLUE 0130205
P06COG    W3RED  0200215
P07GEAR   W3GREY 0100234
P08BEARINGW2GREY 0260136
P09BOLT   W1WHITE0060015
```

10.1.1 INVENT Example Components

The INVENT program is comprised of the following files. These files can be found in the S3X$EXAMPLES directory. It is strongly recommended that the files be copied to another directory before they are used.

Table 10–1 INVENT Program Files

File	Description
INVENT.COM	Procedure used to recreate the INVENT.DAT file.
INVENT.DAT	Indexed inventory data file.
INVENT.FDL	Description file used to create INVENT.DAT.
INVENT.RPG	Inventory maintenance program source code.
INVENT.TXT	Sequential version of INVENT.DAT. This file is used by the INVENT.COM procedure to recreate the INVENT.DAT file. It is recommended that this file not be modified.
INVENTFM.FRM	Inventory maintenance program screen source code.

10.1.2 INVENT.RPG Source Code

This is the Migration RPG source code that comprises the INVENT.RPG program:

Example 10–2 Combine Demand WORKSTN Program Example - INVENT.RPG

```
         1         2         3         4         5         6
123456789012345678901234567890123456789012345678901234567890123456789
     H
      * INVENT.RPG
      * Program to review and modify parts inventory.
      * Program uses a combine demand WORKSTN file.
      *
0010 FPARTS   CD  F      25            WORKSTN
0020 FINVENT  UC  F      24R 3AI     1 DISK
      *
0030 IINVENT
0040 I                                       1  24 DATA
      *
      * WORKSTN input descriptions.  Note the use
      * of position 1 in the input records to identify
      * the input screen.
      *
0050 IPARTS       01   1 CA
0060 I                                       2   4 PN
0070 I            02   1 CB
0080 I                                       2  25 DATA
0085 I            03
      *
0090 IDATA        DS
0100 I                                       1   3 PN
0110 I                                       4  10 PNAME
0120 I                                      11  12 WHOUSE
```

Example 10–2 Cont'd on next page

Example 10–2 (Cont.) Combine Demand WORKSTN Program Example - INVENT.RPG

```
0130 I                                        13  17 COLOR
0140 I                                        18  200WEIGHT
0150 I                                        21  240QTY
      *
      * Output Part # screen.
      *
0160 C                     EXCPTP_NUM
      *
      * Read Part # screen.  If screen terminated with
      * a Cmd7, exit program.
      *
0170 C                     READ PARTS
0180 C   KG                SETON                     LR
0190 C   KG                GOTO END
      *
      * Look for Part in Inventory file.  If Part not
      * found, display the Error screen.
      *
0200 C           PN        CHAININVENT               99
0210 C   99                EXCPTP_ERR
0215 C   99                READ PARTS
0217 C   99 KG             SETON                     LR
0220 C   99                GOTO END
      *
      * If Part was found, display the update screen
      * and retrieve data from it.  Place the updated
      * record in the Inventory file.
      *
0230 C                     EXCPTP_UPD
0240 C                     READ PARTS
0242 C   KG                SETON                     LR
0244 C   KG                GOTO END
0250 C                     EXCPTINV
      *
0260 C           END       TAG
      *
      * WORKSTN output descriptions.  Note that the first
      * output field specification always lists the name
      * of the screen the output is to be sent to.
      *
0270 OPARTS   E                P_NUM
0280 O                                    K8 'PARTNUMB'
0290 O                                     1 'A'
0300 O                         PN          4
0310 O        E                P_UPD
0320 O                                    K6 'UPDATE'
0330 O                                     1 'B'
0340 O                         DATA       25
0350 O        E                P_ERR
0360 O                                    K5 'ERROR'
0370 O                                    22 'No part by that number'
      *
0380 OINVENT  E                INV
0390 O                         DATA       24
```

— The WORKSTN file is defined on line **10** as a combined demand file. The record length of 25 is chosen based on the longest WORKSTN input buffer required.

— Line **20** shows the definition for the inventory file.

— The data layout for the inventory file is given in the data structure on lines **90** through **150** and referenced in the input record on lines **30** through **40**.

— Lines **50** through **85** show the definition of the input records provided by the WORKSTN screens. The first character in the WORKSTN file record buffer is examined to see if it is 'A' or 'B'. Note that lines **290** and **330** place an 'A' or 'B' into the first position of the WORKSTN file record buffer. This first byte corresponds to the CODE field, specified as part of the two WORKSTN input records. The CODE field is used to provide a means for the Input specifications to identify the last input screen. The CODE fields could also have been defined as input constants within the WORKSTN screens.

— Line **85** provides a catch-all input record definition to allow the program to process input records that do not have an 'A' or 'B' in position 1. This functionality is needed when returning control from the ERROR screen.

— Lines **270** through **370** show the specifications that output data to the WORKSTN screens. The three screens are listed. Each screen uses an *EXCPT* name to make it easy to select the appropriate screen for display. Note that data is placed in the WORKSTN record buffer in lines **290** through **300 330** through **340**, and line **370**. The ending positions for these fields match the end positions for the output data fields in the WORKSTN Description specifications.

— Lines **160** through **260** define the actions performed during each cycle.

1 The PARTNUMB screen is displayed on line **160**.

2 Input is requested from the WORKSTN file on line **170**. An 'A' was output in the first position of the WORKSTN output record and will be returned in the first position of the WORKSTN input record by the *READ* operation. The 'A' will identify the WORKSTN input record as belonging in the Input specifications on lines **50 - 60**

3 On line **180** the program checks for screen termination via Cmd7. If Cmd7 was entered, the program is terminated (**190**).

4 The part number obtained from the PARTNUMB screen is used on line **200** to retrieve part information from the inventory file.

5 Line **210** displays the ERROR screen if part record retrieval is unsuccessful. Line **215** pauses the program so the error message can be viewed. The user must press Record/Enter Adv or Cmd7 to leave the ERROR screen and return control to the program.

6 A check is again made for the use of Cmd7 to terminate the program on line **217**.

7 Line **230** displays the part data on the UPDATE screen.

8 Input is requested from the WORKSTN file on line **240**. A 'B' was output in the first position of the WORKSTN output record and will be returned in the first position of the WORKSTN input record by the *READ* operation. The 'B' will identify the WORKSTN input record as belonging in the Input specifications on lines **70 - 80**.

9 A check is again made for the use of Cmd7 to terminate the program on lines **242** and **244**.

10 The inventory file is updated on line **250**.

11 The cycle is repeated until a Cmd7 is used to terminate the program.

10.1.3 INVENTFM.FRM Screen Source Code

This is the Migration RPG source code that comprises the INVENTFM.FRM screen module:

— The Screen specifications on lines **10**, **100**, and **300**, have Y's in columns 27 and 28. This says that only the function and command keys defined in the key masks in columns 64 - 79 are enabled for these screens. Only Cmd7 (G) has been enabled on each of the screens. No function keys are enabled in this program.

— The field names specified in columns 7 - 14 on the Description specifications, such as lines **40** and **70**, are treated as comments. They are provided for information purposes only and do not compile in the screen.

— On lines **40** and **140**, the CODE field is defined as an input/output (Y's in columns 23 and 26) and protected field (Y in column 37). This ensures that the data output to the WORKSTN screen in the CODE field will be returned to the program and that it cannot be modified by the user.

— On line **70**, the Controlled Field Exit (column 35) and Auto Record Advance (column 36) functions are both enabled. A Controlled Field Exit requires that the user press a field termination key (Enter, Tab, Record Enter/Adv) to leave the field. The Auto Record Advance terminates the screen and returns control to the program as soon as the field is entered.

Cmd7 will also terminate the screen and return control to the program.

— On line **160** in the UPDATE screen, the part number (PN) is marked as a protected field (Y in column 37). This ensures that the field value will be returned to the program and that it cannot be modified by the user. The UPDATE screen displays the part number field, but does not permit it to be modified.

Example 10–3 WORKSTN Screen Example - INVENTFM.FRM

```
         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
      *
      * INVENTFM.FRM
      * Screen for interactive inventory review and updates.
      *
0010 SPARTNUMB   124      YY                                    G
0020 D          17 132Y               Y               CINVENTORY UTILITY
0030 D          10 168Y                               CScreen ID:
0040 DCODE       1 179Y  Y          Y
0050 D          10 235Y               Y               CKEY SCREEN
0060 D          19 426Y                               CPart Number:
0070 DPN         3 446Y  Y        YY  Y       Y
0080 D          2324 1Y               Y               C[CMD7] to Exit
      *
0100 SUPDATE     124      YY                                    G
0110 D          33 1 1Y               Y               CINVENTORY UTILITY - UPDX
0120 DATE SCREEN
0130 D          10 168Y                               CScreen ID:
0140 DCODE       1 179Y  Y          Y
0150 D          19 3 1Y                               CPart Number:
0160 DPN         3 321Y  Y          Y Y
0170 D          23 5 1Y                               CPart Name:
0180 DPNAME      7 525Y  Y            Y       Y
0190 D          23 6 1Y                               CWarehouse Code:
0200 DWHOUSE     2 625Y  Y            Y       Y
0210 D          23 7 1Y                               CColor:
0220 DCOLOR      5 725Y  Y            Y       Y
0230 D          23 8 1Y                               CWeight:
0240 DWEIGHT     3 825Y  YD           Y       Y
0250 D          23 9 1Y                               CQuantity:
0260 DQTY        4 925Y  YD           Y       Y
0270 D          2324 1Y               Y               C[CMD7] to Exit
      *
0300 SERROR      124      YY                                    G
0310 D          33 1 1Y               Y               CINVENTORY UTILITY - ERRX
0320 DOR SCREEN
0330 D          22 529Y               Y Y
0340 D          7924 1Y               Y               CPress [Enter] to returnX
0350 D to the Part # screen or [CMD7] to exit the program.
```

— Lines **240** and **260** in the UPDATE screen define the numeric fields WEIGHT and QTY. The D in the Data Type field (column 27) prevents the user from entering any characters other than 0 - 9 in the field.

— The ERROR screen does not have any input fields. This screen is a display only screen. The user must use a Enter, Enter/Record Adv, or Cmd7 to terminate the screen and return control to the program.

10.1.4 Building the INVENT Program

The INVENT WORKSTN program can be compiled and linked using one of the following sets of commands:

Example 10–4 BUILD INVENT

```
$ BUILD INVENT
```

The BUILD command will compile the INVENT.RPG program, the INVENTFM.FRM screen format file, and then link the two object modules to produce the INVENT.EXE executable image.

Example 10–5 Compile and Link INVENT

```
$ RPG INVENT
$ SFG INVENTFM
$ LINK INVENT, INVENTFM
```

The RPG command compiles the INVENT.RPG program. The SFG command compiles the INVENTFM.FRM screen format file. The LINK command links the two object modules to produce the INVENT.EXE executable image.

10.2 Combine Primary WORKSTN Program Example

This section provides an example of a simple combine primary WORKSTN program, INVENT_INQ.RPG. The program accesses the screens defined in the INVENT_INQFM.FRM screen format file. The following three screen formats are defined for the program:

- PARTNUMB
- DATA
- ERROR

The INVENT_INQ program displays inventory records from the INVENT.DAT file. The program is an example of a display-only application; the user is not allowed to update inventory information. The PARTNUMB screen is used to retrieve the three-character part number that is the key into the inventory file. The DATA screen is used to display the inventory information if the specified part number is found. The ERROR screen displays an error message if an invalid part number is entered on the PARTNUMB screen. The program is terminated by entering a Cmd7 from any of the screens.

The INVENT_INQ program uses the same indexed data file as the previous example, INVENT.RPG. The data file is INVENT.DAT.

10.2.1 INVENT_INQ Example Components

The INVENT_INQ program is comprised of the following files. These files can be found in the S3X$EXAMPLES directory. It is strongly recommended that the files be copied to another directory before they are used.

Table 10–2 INVENT_INQ Program Files

File	Description
INVENT.COM	Procedure used to recreate the INVENT.DAT file.
INVENT.DAT	Indexed inventory data file.
INVENT.FDL	Description file used to create INVENT.DAT.
INVENT.TXT	Sequential version of INVENT.DAT. This file is used by the INVENT.COM procedure to recreate the INVENT.DAT file. It is recommended that this file not be modified.
INVENT_INQ.RPG	Inventory inquiry program source code.
INVENT_INQFM.FRM	Inventory inquiry program screen source code.

10.2.2 INVENT_INQ.RPG Source Code

This is the Migration RPG source code that comprises the INVENT_INQ.RPG program:

— The WORKSTN file is defined on line **10** as a combined primary file. The record length of 24 is chosen based on the longest WORKSTN output buffer required.

— Line **20** shows the definition for the inventory file.

— The data layout for the inventory file is given in the data structure on lines **140** through **200** and referenced in the input record on line **110**.

— Lines **120** through **136** show the definition of the input records provided by the WORKSTN screens. The first character in the WORKSTN file record buffer is examined to see if it is 'A' or 'B'. The 'A' and 'B' are placed in the WORKSTN input record by the WORKSTN screens. This first byte is used to provide a means for the Input specifications to determine the last input screen.

— Line **136** provides a catch-all input record definition to allow the program to process input records that do not have an 'A' or 'B' in position 1 of the input record. This functionality is needed when returning control from the ERROR screen.

— Lines **400** through **490** show the specifications that output data to the WORKSTN screens. Lines **410**, **450**, and **480** define the WORKSTN screens. Data is placed in the WORKSTN record buffer on lines **420**, **460**, and **490**. The ending positions for these fields match the end positions for the output data fields in the WORKSTN Description specifications. Output indicators in columns 23 - 31 determine which screen is to be output on each cycle.

— Note the use of the 1P indicator on line **404** to ensure that a screen is output during program startup.

Example 10–6 Combine Primary WORKSTN Program Example - INVENT_INQ.RPG

```
         1         2         3         4         5         6
123456789012345678901234567890123456789012345678901234567890123456789
     H
      * INVENT_INQ.RPG
      * Program to display parts inventory records.
      *
      * Program uses a combine primary WORKSTN file.
      *
0010 FPARTS   CP  F      24            WORKSTN
0020 FINVENT  IC  F      24R 3AI     1 DISK
      *
0100 IINVENT      02
0110 I                                        1  24 DATA
      *
0120 IPARTS       01   1 CA
0130 I                                        2   4 PN
0132 I            02   1 CB
0136 I            03
      *
0140 IDATA        DS
0150 I                                        1   3 PN
0160 I                                        4  10 PNAME
0170 I                                       11  12 WHOUSE
0180 I                                       13  17 COLOR
0190 I                                       18  200WEIGHT
0200 I                                       21  240QTY
      *
0300 C   KG               SETON                        LR
0310 C   01NLR   PN        CHAININVENT                99
      *
0400 OPARTS   D        02NLR
0402 O       OR        03NLR
0404 O       OR        1P
0410 O                                    K8 'PARTNUMB'
0420 O                          PN         4
0440 OPARTS   D        99N03NLR
0450 O                                    K5 'ERROR'
0460 O                                    22 'No part by that number'
0470 OPARTS   D        01N99NLR
0480 O                                    K6 'DATA'
0490 O                          DATA      25
```

— WORKSTN input and output are controlled by the program cycle.

- At program startup, the 1P indicator is on, so the PARTNUMB screen is displayed.
- During the first cycle input phase, input is requested from the WORKSTN file. An 'A' is included in the first position of the WORKSTN record returned by the PARTNUMB screen. This sets on the 01 input indicator.
- Line **300** checks for entry of Cmd7. If Cmd7 was entered, the last record indicator (LR) is turned on. This will terminate the program. The Cmd7 check is done every cycle because Cmd7 is a valid entry from all of the program screens.
- If the program is not being terminated (LR) and the last screen input was the PARTNUM screen (01), line **310** takes the part number (PN) entered on the PARTNUMB screen and tries to locate the associated part record in the inventory file.

- If the part is not found in the Inventory file, indicator 99 comes on. This will result in the ERROR screen being displayed during the output phase of the program cycle. If the part is found in the inventory file, indicator 99 is not turned on. This results in the part record being displayed on the DATA screen during the output phase of the program cycle.
- The program now begins a new cycle. WORKSTN input is again obtained during the input phase of the program cycle. The last screen displayed determines what the next screen displayed will be. Entry of a Cmd7 from any of the screens terminates the program.

10.2.3 INVENT_INQFM.FRM Screen Source Code

This is the Migration RPG source code that comprises the INVENT_INQFM.FRM screen module:

— The Screen specifications on lines **10**, **100**, and **300**, have Y's in columns 27 and 28. This says that only the function and command keys defined in the key mask in columns 64 - 79 are enabled for these screens. Only Cmd7 (G) has been enabled on each of the screens. No function keys are enabled in this program.

— The field names specified in columns 7 - 14 on the Description specifications, such as lines **70** and **180**, are treated as comments. They are provided for information purposes only and do not compile in the screen.

— On lines **40** and **140**, the CODE field is defined as an input constant (Y in column 23, C in column 56). The field is also defined as an output field (Y in column 26) and a protected field (Y in column 37). This ensures that the field value will be returned to the program and that it cannot be modified by the user. These fields will appear as the first character in the WORKSTN input record and are used to identify the input record within the RPG program.

— On line **70**, the Controlled Field Exit (column 35) and Auto Record Advance (column 36) functions are both enabled. A Controlled Field Exit requires that the user press a field termination key (Enter, Tab, Record Enter/Adv) to leave the field. The Auto Record Advance terminates the screen and returns control to the program as soon as field is entered.

Cmd7 will also terminate the screen and return control to the program.

— The DATA and ERROR screens do not have any input fields. These screens are display only screens. The user must use a Enter, Enter/Record Adv, or Cmd7 to terminate the screen and return control to the program.

Example 10–7 WORKSTN Screen Example - INVENT_INQFM.FRM

```
         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
      * INVENT_INQFM.FRM
      * Screen for interactive display of inventory records.
      *
0010 SPARTNUMB   124      YY                                  G
0020 D          17 132Y               Y                CINVENTORY INQUIRY
0030 D          10 168Y                                CScreen ID:
0040 DCODE       1 179Y  Y          Y                  CA
0050 D          10 235Y               Y                CKEY SCREEN
0060 D          19 426Y                                CPart Number:
0070 DPN         3 446Y  Y        YY  Y       Y
0080 D          2324 1Y               Y                C[CMD7] to Exit
      *
0100 SDATA       124      YY                                  G
0110 D          31 1 1Y               Y                CINVENTORY INQUIRY - DATX
0120 DA SCREEN
0130 D          10 168Y                                CScreen ID:
0140 DCODE       1 179Y  Y          Y                  CB
0150 D          19 3 1Y                                CPart Number:
0160 DPN         3 321Y               Y
0170 D          23 5 1Y                                CPart Name:
0180 DPNAME      7 525Y               Y       Y
0190 D          23 6 1Y                                CWarehouse Code:
0200 DWHOUSE     2 625Y               Y       Y
0210 D          23 7 1Y                                CColor:
0220 DCOLOR      5 725Y               Y       Y
0230 D          23 8 1Y                                CWeight:
0240 DWEIGHT     3 825Y               Y       Y
0250 D          23 9 1Y                                CQuantity:
0260 DQTY        4 925Y               Y       Y
0270 D          7924 1Y               Y                CPress [Enter] to returnX
0280 D to the Part # screen or [CMD7] to exit the program.
      *
0300 SERROR      124      YY                                  G
0310 D          33 1 1Y               Y                CINVENTORY INQUIRY - ERRX
0320 DOR SCREEN
0330 D          22 529Y               Y Y
0340 D          7924 1Y               Y                CPress [Enter] to returnX
0350 D to the Part # screen or [CMD7] to exit the program.
```

Example 10–8 BUILD INVENT_INQ

```
$ BUILD INVENT_INQ
```

The BUILD command will compile the INVENT_INQ.RPG program, the INVENT_INQFM.FRM screen format file, and then link the two object modules to produce the INVENT_INQ.EXE executable image.

10.2.4 Building the INVENT_INQ Program

The INVENT_INQ WORKSTN program can be compiled and linked using one of the following sets of commands:

Example 10–9 Compile and Link INVENT_INQ

```
$ RPG INVENT_INQ
$ SFG INVENT_INQFM
$ LINK INVENT_INQ, INVENT_INQFM
```

The RPG command compiles the INVENT_INQ.RPG program. The SFG command compiles the INVENT_INQFM.FRM screen format file. The LINK command links the two object modules to produce the INVENT_INQ.EXE executable image.

10.3 TEMPLATE Examples

The Migration RPG Compiler kit provides two WORKSTN template programs. These programs are intended to serve as interactive file maintenance and inquiry templates. The programs are more sophisticated than the examples described in this chapter.

The programs are TEMPLATE_MAINT and TEMPLATE_INQ. Their components are listed in the following sections. It is strongly recommended that these files be copied to another location before they are used.

The template programs are well documented with internal comments. They can be used to quickly develop interactive file inquiry and maintenance programs. MSI also provides RPG consulting services. The template programs provide an overview of the quality of our programming services.

10.3.1 TEMPLATE_INQ Example Components

The TEMPLATE_INQ program is comprised of the following files. These files can be found in the S3X$EXAMPLES directory. It is strongly recommended that the files be copied to another directory before they are used.

Table 10–3 TEMPLATE_INQ Program Files

File	Description
TEMPLATE.COM	Procedure used to recreate the TEMPLATE.DAT file.
TEMPLATE.DAT	Indexed template data file.
TEMPLATE.FDL	Description file used to create TEMPLATE.DAT.
TEMPLATE.TXT	Sequential version of TEMPLATE.DAT. This file is used by the TEMPLATE.COM procedure to recreate the TEMPLATE.DAT file. It is recommended that this file not be modified.
TEMPLATE_INQ.COM	Inquiry template command procedure.
TEMPLATE_INQ.RPG	Inquiry template program source code.
TEMPLATE_INQFM.FRM	Inquiry template program screen source code.

10.3.2 TEMPLATE_MAINT Example Components

The TEMPLATE_MAINT program is comprised of the following files. These files can be found in the S3X$EXAMPLES directory. It is strongly recommended that the files be copied to another directory before they are used.

Table 10–4 TEMPLATE_MAINT Program Files

File	Description
TEMPLATE.COM	Procedure used to recreate the TEMPLATE.DAT file.
TEMPLATE.DAT	Indexed template data file.
TEMPLATE.FDL	Description file used to create TEMPLATE.DAT.
TEMPLATE.TXT	Sequential version of TEMPLATE.DAT. This file is used by the TEMPLATE.COM procedure to recreate the TEMPLATE.DAT file. It is recommended that this file not be modified.
TEMPLATE_MAINT.COM	Maintenance template command procedure.
TEMPLATE_MAINT.RPG	Maintenance template program source code.
TEMPLATE_MAINTFM.FRM	Maintenance template program screen source code.

10.4 Address Book Interactive Program Example

The Migration RPG Compiler Kit includes an address book program example. This is an interactive RPG program that was developed using the TEMPLATE_MAINT template files. The program maintains an address book file.

The ADDRESS_BOOK program is well documented with internal comments.

10.4.1 Address Book Example Components

The Address Book program is comprised of the following files. These files can be found in the S3X$EXAMPLES directory. It is strongly recommended that the files be copied to another directory before they are used.

Table 10–5 Address Book Program Files

File	Description
ADDRESS_BOOK.COM	Procedure used to execute the Address Book program.
ADDRESS_BOOK.DAT	Indexed Address Book data file.
ADDRESS_BOOK.FDL	Description file used to create ADDRESS_BOOK.DAT.
ADDRESS_BOOK.RPG	Address Book program source code.
ADDRESS_ BOOKFM.FRM	Address Book screen format source code.

Index

A

B

C

D

E

F

H

I

K

L

M

N

O

P

R

S

T

U

V

W