RX01 (M8357) Omnibus Floppy Disk Interface

**Abstract**

This document describes the RX01 (M8357) omnibus floppy disk interface for the PDP-8/E, /F, /M, and /A computers. The RX02 and VT78 diskette interfaces are largely upwards compatible from this, as is the DSD-210 third-party RX01 compatible drive.

The RX02 supported double-density diskettes, and the RX03 added support for double-sided diskettes.

Table of Contents

[Device Registers 2](#_Toc524870495)

[Command Register 2](#_Toc524870496)

[Interface Register 3](#_Toc524870497)

[Track Address Register 3](#_Toc524870498)

[Sector Address Register 3](#_Toc524870499)

[Error Status Register 4](#_Toc524870500)

[Error Code Register 5](#_Toc524870501)

[The Buffer 5](#_Toc524870502)

[The Transfer Request Flag 6](#_Toc524870503)

[The Error Flag 6](#_Toc524870504)

[The Done Flag 6](#_Toc524870505)

[The Interrupt Enable Bit 6](#_Toc524870506)

[Instruction Format and Device Addresses 7](#_Toc524870507)

[Instructions (device x) 8](#_Toc524870508)

[6750 - SEL Select device pair 8](#_Toc524870509)

[67x1 - LCD Load Command register 8](#_Toc524870510)

[67x2 - XDR transfer Data Register 9](#_Toc524870511)

[67x3 - STR Skip if Transfer Request 9](#_Toc524870512)

[67x4 - SER Skip if Error 10](#_Toc524870513)

[67x5 - SDN Skip if Done 10](#_Toc524870514)

[67x6 - INTR Interrupt enable 10](#_Toc524870515)

[67x6 - INIT Initialize 10](#_Toc524870516)

[Programming Conventions 11](#_Toc524870517)

# Device Registers

The RX01 interface consists of two coupled components, the interface itself and the microprogrammed disk controller. Here, we will not distinguish between these. The computer interacts with this device through 8 IOT instructions; these instructions, in turn, reference the registers described below.

Note: Most of these registers are accessed via the accumulator; such registers are illustrated as their contents appears when loaded in the accumulator.

## Command Register

 00 01 02 03 04 05 06 07 08 09 10 11

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 | | | | | | | | | | | | |

 |\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|

 | | | | | | | | |

 | | H| D| M| E| U | F | |

The command register holds the current diskette interface function, and the operation of loading the command register initiates these functions. The command register has the following fields:

* H -- Head Select (not RX01).
For the RX02 and RX03, this selects the head to use. On an RX02, it must be zero (because only one head but the selection function is implemented). On an RX03, it is used to determine which side of the disk is to be used.
* D -- Density (not RX01).
For the RX02, this selects the recording density to be used.
* M -- Maintenance.
If this bit is set, the interface operates in maintenance mode.
* E -- Eight bit mode.
If this bit is set, the interface loads and stores sectors as streams of 8 bit bytes. If reset, sectors are loaded and stored as streams of 12 bit words.
* U -- Unit select.
The RX01 through RX03 dual diskette drives use only the low order bit of the unit select field. The DSD210 allows up to 4 drives and uses the full unit select field.
* F -- Function select.
	+ 0 -- Fill Buffer.
	+ 1 -- Empty Buffer.
	+ 2 -- Write Sector.
	+ 3 -- Read Sector.
	+ 4 -- No Op.
	+ 5 -- Read Status.
	+ 6 -- Write Deleted Data Sector.
	+ 7 -- Read Error Register.

## Interface Register

 00 01 02 03 04 05 06 07 08 09 10 11

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 | | | | | | | | | | | | |

 |\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|

 | | |

 | |<--- 8-bit mode --->|

 | |

 |<------ 12-bit mode ------>|

The interface register is used for transfers of data between the accumulator and the RX01 controller. As such, it is used in filling and emptying the controller's buffer, for transferring track and sector addresses to the controller and for transferring error reports from the controller.

## Track Address Register

 00 01 02 03 04 05 06 07 08 09 10 11

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 | | | | | | | | | | | | |

 |\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|

 | | |

 | Ignored | Track Address |

Valid track addresses range from 0 to 76 (decimal), but standard software never uses track 0 of a disk.

## Sector Address Register

 00 01 02 03 04 05 06 07 08 09 10 11

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 | | | | | | | | | | | | |

 |\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|

 | | |

 | Ignored | Sector Address |

Valid sector addresses range from 1 to 26 (decimal), but note! The sector address for each sector is recorded on the diskette when that diskette is formatted. Nonstandard formatting software could, in theory, assign other sector numbers! The RX01 interface cannot be used to format a diskette, but the DSD-210 supports formatting.

## Error Status Register

 00 01 02 03 04 05 06 07 08 09 10 11

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 | | | | | | | | | | | | |

 |\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|

 DD DER ID CRC

 RDY DEN WP PAR

 RX2 DS

The Error Status Register contains a large number of one-bit flags; for some of these, the meaning depends on which drive type is in use.

* RDY -- Device ready.
* DD -- Deleted data encountered on read.
* DEN -- not RX01: Density of sector just read or written.
* DER -- not RXO1: Density error in read.
* WP -- RX01 and DSD210 only: Write protect or not-ready.
* RX2 -- not RX01: always 1, reports RX02 attached.
* ID -- initialization done.
* PAR -- RX01 and DSD210 only: Parity error detected.
* DS -- RX03 only: Double sided media detected.
* CRC -- A CRC Error was detected.

Note that the parity error bit reports on internal errors and not on errors encountered in the actual disk I/O operation, and that the same bit is used on the RX03 to indicate the presence of double-sided media. The write protect bit is not supported by most RX01 drives, but the option was supported on some and was standard on the DSD-210. On both the modified RX01 and on the DSD-210, this bit also indicates drive-not ready. On the DSD-210, it reports both the state of the write-protect switch and of the notch in the diskette. Unfortunately, on the RX02, this bit was used to indicate that the drive was an RX02, and a fair amount of software used this bit to sense the drive type, making it incompatible with the DSD-210 and the few RX01s that supported write protect sensing.

The correct way to distinguish, in software, between an RX01 and an RX02/03 is to do a read-status command with the double density bit set. On an RX01, the double-density bit will be ignored. On an RX02 or RX03, the result will be either a density error or a report that double-density media has been detected, so the error status and 0060 (octal) will be nonzero.

## Error Code Register

 00 01 02 03 04 05 06 07 08 09 10 11

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 | | | | | | | | | | | | |

 |\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|

 | | |

 | | Error Code |

The Error Code Register contains a numeric error code indicating the cause of the most recently detected error. Codes are given in octal.

* 0010 -- Drive 0 failed to see home on initialize.
* 0020 -- Drive 1 failed to see home on initialize.
* 0030 -- Found home prematurely on init (RX01 only).
* 0040 -- Bad track address.
* 0050 -- Home was found prematurely during seek.
* 0060 -- Self diagnostic error during init (RX01 only).
* 0070 -- Seek failed (bad sector address).
* 0100 -- Attempt to write a protected disk (RX01 only).
* 0110 -- No diskette in the requested drive.
* 0120 -- No preamble found (disk not formatted).
* 0130 -- Preamble but no ID mark (bad format).
* 0140 -- CRC error on a header (RX01 only).
* 0150 -- Wrong track address in header.
* 0160 -- Too many tries to find header.
* 0170 -- No data following header.
* 0200 -- CRC error on read from disk.
* 0210 -- Internal parity error (not RX01).
* 0220 -- R/W electronics failed self-test (not RX01).
* 0230 -- Word count overflow (not RX01).
* 0240 -- Density error (not RX01).
* 0250 -- Wrong key for set media density (not RX01).

## The Buffer

The RX01 controller buffer holds one sector (128 bytes), either the most recent buffer read from the diskette or the most recent buffer transferred from the PDP-8 to the controller. This buffer is sometimes referred to as the silo. On the RX02 and RX03, the buffer holds 256 bytes, with the extra capacity used only in double-density mode.

In 12-bit mode, two consecutive 12-bit words are packed into three consecutive 8-bit bytes most significant bit first, so the first byte holds the top 8 bits of the first word, the high end of the second byte holds the bottom 4 bits of the first word followed by the top 4 bits of the second word, and the third byte holds the bottom 8 bits of the second word. In 12 bit mode on an RX01 only the first 96 bytes (64 words) of the buffer are used, and on ouptut, the final 8 bits written gets repeated to fill the buffer.

## The Transfer Request Flag

The Transfer Request Flag bit is set when most recently requested transfer to or from the interface register has been completed.

## The Error Flag

The Error Flag bit is set when any error is detected. The cause of the error is reported in the Error Status Register and the Error Code Register.

## The Done Flag

The Done Flag bit is set when the controller completes a command.

## The Interrupt Enable Bit

 00 01 02 03 04 05 06 07 08 09 10 11

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 | | | | | | | | | | | | |

 |\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|

 | | |

 IE

The Interrupt Enable Bit allows the RX01 interface to request an interrupt when the Done or Error Flag is set. Note that, unlike essentially all other PDP-8 peripherals, the interface will not request an interrupt to signal a transfer request! The working assumption is that transfer requests are sufficiently high speed that an interrupt on transfer request would be inappropriate. Indeed, in an interrupt driven system, the transfer of blocks of data between the controller and CPU can be accelerated by writing the transfer loops without any termination test, relying on the done interrupt to terminate the loop.

# Instruction Format and Device Addresses

 00 01 02 03 04 05 06 07 08 09 10 11

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 | 1| 1| 0| 1| 1| 1| x| x| x| | | |

 |\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|\_\_|

 | | | |

 | IOT | Device | Op |

The RX01 interface is almost always connected as device 75, however, the (M8357) interface allows for any address from 70 to 77. If a second device is attached, it should usually be connected as device 76. On the VT78 and later interfaces, the SEL command is used for drive selection and the interface is always through device 75.

# Instructions (device x)

## 6750 - SEL Select device pair

 SEL

For the VT78 and later interfaces for the RX01/RX02, this command loads the least significant bit of AC into the drive pair select register. These machines supported two drive pairs. On the earlier RX01/RX02 interfaces, the drive pair was selected by using a different device address for each pair of disk drives.

## 67x1 - LCD Load Command register

 LCD

The command register is loaded from the accumulator and the accumulator is cleared. Note that on an RX02 or RX03, the command has 10 defined bits and that only the low 8 bits are loaded if the interface is in 8-bit mode. As a result, on an RX02 or RX03, when 8 bit mode is selected, the high bits of the command register must be loaded with an extra STR-XDR sequence immediately following the LCD. The XDR sends the high bits of the command register from the least significant bits of the accumulator.

Once the command register is loaded, the controller starts executing the indicated command. The interpretation of following instructions depends on the command loaded, as follows:

0 -- Fill Buffer.

The following sequence of 128 (in 8-bit mode) or 64 (in 12-bit mode) XDR instructions transfer data from the accumulator via the interface register to the buffer, until the buffer is filled, at which point the done flag will be set.

1 -- Empty Buffer.

The following sequence of 128 (in 8-bit mode) or 64 (in 12-bit mode) XDR instructions transfer data from the buffer via the interface register to the accumulator, until the buffer has been emptied, at which point the done flag will be set.

2 -- Write Sector.

3 -- Read Sector.

The first XDR following either of these instructions will transfer the sector address from the accumulator via the interface register. The second XDR will transfer the track address from the accumulator via the interface register and initiate a seek for the indicated track and sector. After the seek, the indicated transfer between the buffer and the diskette will take place, and then the done flag will be set to signal completion of the transfer.
(On the DSD-210, if the sector address given is 154 (octal), this is a format command, and it should be followed by a sequence of 26 additional XDR instructions giving the 26 sector numbers to be written to the specified track, in whatever interleaved order is desired.)

4 -- Set Media Density (RX02 only).

The following XDR instruction transfers the key (0111 octal), verifying that this is indeed a set media density command and not an RX01 no-op. The desired density is taken from the command register.

5 -- Read Status.

The controller will check to see that a disk is in the drive and spinning properly, and then set the done flag. The drive test is done by reading the header on sector one of the current track, so this command may take as much as one full disk revolution.

6 -- Write Deleted Data Sector.

Same as Write Sector, except that the sector is marked as deleted. After reading such a sector, the deleted data bit will be set in the error status register. No known PDP-8 software other than diagnostics uses this feature, which rests on one bit per sector allocated in the standard IBM 8-inch diskette format to mark sectors as having been deleted.

7 -- Read Error Register.

This instruction moves the error register into the interface register and then sets the done flag. This is the recommended way to set the done flag if no device operaton is required.

## 67x2 - XDR transfer Data Register

 XDR

This operation of this instruction depends on the current contents of the command register and the state of the controller. It either:

Loads the high bits of the command register:

This applies to the first XDR immediately following an LCD that set the interface into 8-bit mode on an RX02 or RX03 only.

Loads the accumulator into the interface register:

This applies to all XDRs after a Fill Buffer command, and to the two XDRs used to load the track and sector registers after a Read Sector, Write Sector or Write Deleted Data Sector command.

Ors the interface register with the accumulator:

This applies to all XDRs after an Empty Buffer command in 8-bit mode, and to the XDR after a Read Error Register command, and to the XDR after a Read Status command, and to the XDR after a SDN command in 8-bit mode.

Loads the interface register into the accumulator:

This applies to all XDRs after an Empty Buffer command in 12-bit mode. and to the XDR after a SDN command in 12-bit mode.

## 67x3 - STR Skip if Transfer Request

 STR

Skip if the Transfer Request Flag is set, and reset the Transfer Request. This instruction must precede every XDR between the issuing of a command and the setting of the done flag! Data transfers between the interface register and the controller are done over a high-speed serial link, and take 18 to 23 microseconds, depending on whether 8 or 12 bits are being transferred.

## 67x4 - SER Skip if Error

 SER

Skip if the Error Flag is set, and reset the Error Flag.

## 67x5 - SDN Skip if Done

 SDN

Skip if the Done Flag is set, and reset the Done Flag. After any command other than a read error register command, the interface register is loaded with the contents of the error status register. In the exceptional case, the interface register is loaded with the error code. In either case, an XDR following the setting of the done flag will load the accumulator from the interface register.

## 67x6 - INTR Interrupt enable

 INTR

The least significant bit of the accumulator is loaded into the interrupt enable bit.

## 67x6 - INIT Initialize

 INIT

The device is initialized, a time-consuming process that involves a sequence of seeks and then a full rotation of the disk to verify that a formatted disk is present and read track 1 sector 1 into the interface buffer.

# Programming Conventions

Typical PDP-8 assembly language programs begin by defining the necessary IOT instructions. In the case of the RX01 interface, these instructions are rarely predefined and must be defined as follows:

 LCD= 6751 / Load Command register

 XDR= 6752 / transfer Data Register

 STR= 6753 / Skip if Transfer Requested

 SER= 6754 / Skip if Error

 SDN= 6755 / Skip if Done

 INTR= 6756 / Interrupt enable

 INIT= 6757 / Initialize

In addition, it is common to give symbolic definitons that simplify formulation of command words:

 MAINT= 0200 / the maintenance bit

 BITS8= 0100 / select 8 bit mode

 BITS12= 0000 / select 12 bit mode

 DRIVE0= 0000 / select drive 0

 DRIVE1= 0020 / select drive 1

 FNFILL= 0000 / fill buffer from cpu

 FNEMPT= 0002 / empty buffer to cpu

 FNWRIT= 0004 / write buffer to disk

 FNREAD= 0006 / read buffer from disk

 FNRDST= 0012 / read status to cpu

 FNWRDE= 0014 / write deleted buffer to disk

 FNRDER= 0016 / read error register to cpu

Given these, the following routine will initialize an RX01 drive and test for media present; a more sophisticated initialization routine would return a more useful error report.

 RXINIT, .-.

 / Initialize a drive

 / expects DRIVE0 or DRIVE1 in AC

 / returns error status word in AC

 / on return, skips if no error and ready.

 TAD (FNRDST) / make read stat command

 LCD / and issue it

 SDN / wait for completion

 JMP .-1

 XDR / get error-status codes

 SER

 JMP RXINOE / skip if no errors

 JMP I RXINIT / error return

 RXINOE, AND (0200) / test drive ready bit

 SZA

 ISZ RXINIT / normal return bump PC

 JMP I RXINIT / normal or not-ready

The following service routines fill and empty the RX8E internal buffer in 12-bit mode. As with the above code, these only make minimal provisions for errors! These two routines are not usually directly called by user code (other than diagnostics), but are used as part of higher level read and write routines.

 RXFILL, .-.

 / given buffer address in AC

 / transfer buffer to RX8E internal buffer

 / wipes out X0

 / skips if no error

 / returns AC = 0

 CMA IAC

 CMA / decrement buffer address

 DCA X0 / setup to index through buffer

 TAD (BITS12+FNFILL+DRIVE0)

 LCD / (note: this clears ac)

 RXFILP, STR / skip if ready to transfer

 SKP

 JMP RXFIDO / go fill buffer if ready

 SDN / if not, see if done

 JMP RXFILP / not done, not ready, go poll

 SER / see if errors

 ISZ RXFILL / skip if no errors

 JMP I RXFILL / return

 RXFIDO, / assert ac = 0

 TAD I X0 / get a word from buffer

 XDR / put it out

 CLA

 JMP RXFILP / await next step

 RXEMPT, .-.

 / given buffer address in AC

 / transfer RX8E internal buffer to buffer

 / wipes out X0

 / skips if no error

 / returns AC = 0

 CMA IAC

 CMA / decrement buffer address

 DCA X0 / setup to index through buffer

 TAD (BITS12+FNEMPT+DRIVE0)

 LCD / (note: clears AC)

 RXEMLP,

 STR / see if ready to transfer

 SKP

 JMP RXEMDO / if so, go transfer word

 SDN / see if done

 JMP RXEMLP / not done, go poll

 SER / done, see if error

 IXZ RXEMPT / skip if no errors

 JMP I RXEMPT / return

 RXEMDO, / assert AC = 0

 XDR / or data word with AC

 DCA I X0 / move to buffer

 JMP RXEMLP / wait for next word

The above code makes no attempt to diagnose errors - it should use an XDR after the SDN command detects completion, and then taken appropriate actions based on the error codes reported.

The following routine for transferring the contents of the buffer to or from disk is also fairly low-level. As with the above code, it has poor error reporting and would normally be called only from higher level disk read and write routines.

 RXIO, .-.

 / read/write internal to or from disk

 / expects DRIVE0 or DRIVE1 in AC

 / other parameters are inline:

 / 1 - FNREAD, FNWRIT or FNWRDE

 / 2 - pointer to sector number

 / 3 - pointer to track number

 / on error, returns AC = error status

 / if no error, skip; AC = 0

 TAD I RXIO / make command

 ISZ RXIO / and advance to next

 LCD / issue command

 TAD I RXIO / get pointer to sector

 ISZ RXIO / and advance to next

 DCA TEMP

 TAD I TEMP / follow pointer

 STR / and then poll to wait

 JMP .-1 / for command to take

 XDR / output sector

 CLA

 TAD I RXIO / get pointer to track

 ISZ RXIO / and advance to next

 DCA TEMP

 TAD I TEMP / follow pointer

 STR / and then poll to wait

 JMP .-1 / for sector transfer

 XDR / output track

 CLA

 SDN / poll to wait

 JMP .-1 / for I/O done

 SER / check for success

 JMP RXINOE

 XDR / get error status

 JMP I RXIO / return

 RXINOE, ISZ RXIO / skip if success

 JMP I RXIO / return

This code reports errors but does not handle them! While it is tempting to issue an INIT command after errors, parity and CRC errors should be dealt with by reissuing the offending command, giving up after some number of retries. If the error is due to write protected media or device not ready, there is certainly no benefit in doing a reset.

When an RX01 application comes into execution, the state of the drive may be unknown! It may be ready to accept a new command, or busy performing a time consuming operation such as the power-on self test. Because the SDN command resets the done flag, the problem of synchronizing the application's model of the device state with the actual state is not trivial. The following sequence of instructions will suffice to solve this problem:

 SDN

 NOP

 TAD (FNRDER)

 LCD

This sequence first uses SDN to reset the done flag, if it was set, and then uses the Read Error Register command to set the done flag. Note that if a command is issued while another command is in progress, that command is ignored. Thus, in the event that a time consuming command was in progress, the above sequence is a no-op and the done flag will come up of its own accord when the previous operation completes.

Under OS/8 (prior to OS/78 and OS/278), the convention was established that applications should exit either with an operation in progress or with the done flag set. The conventional way to set the done flag prior to exit is to execute a Read Error Register command without checking or done or using an XDR to actually read the error register. Handlers written for this environment can use simplified entry sequence, simply waiting for any previous operation to terminate with SDN and then continuing normally.

Another challenge is writing code to determine if the disk drive is an RX01 or an RX02. This can be done by issuing a Read Status command with the double-density bit set (defined only for the RX02). This bit will be ignored on the RX01 and DSD-210, but on the RX02, the returned status register will indicate that double-density has been selected.