HEWLETT PACKARD

RTE Driver DVM72 Universal Interface Driver

Reference Manual

Software Technology Division
11000 Wolfe Road
Cupertino, CA 95014-9804

Manual Part No. 09580-93027
U1283

Printed in U.S.A. June 1993

Printing History

The Printing History below identifies the edition of this manual and any updates that are included. Periodically, update packages are distributed that contain replacement pages to be merged into the manual, including an updated copy of this printing history page. Also, the update may contain write-in instructions.

Each reprinting of this manual will incorporate all past updates; however, no new information will be added. Thus, the reprinted copy will be identical in content to prior printings of the same edition with its user-inserted update information. New editions of this manual will contain new information, as well as all updates.

To determine which manual edition and update is compatible with your current software revision code, refer to the Manual Numbering File or the Computer User's Documentation Index. (The Manual Numbering File is included with your software. It consists of an "M" followed by a five digit product number.)

Third Edition	Aug 1981	To change RTE time-out information
Update 1	Dec 1983	
Reprint	Dec 1983	Update 1 incorporated

Table of Contents

Chapter 1
General Information

Introduction 1-1
Operating Environment 1-1
    Software 1-1
    Hardware 1-1
Components of DVM72 1-2
    Initiation Segment 1-2
    Completion Segment 1-2
Driver Limitations 1-3

Chapter 2
Applications

Introduction 2-1
Standard Read/Write Requests 2-2
    Calling Sequence 2-6
    Control Word for Standard Read/Write Requests 2-6
Read/Write Requests with Subfunction 2-7
    Calling Sequence 2-8
    Control Word for Read/Write Request with Subfunction 2-10
Control Requests 2-10
Standard Control Functions 2-11
    Calling Sequence 2-11
    Standard Control Functions 2-12
Special Control Requests 2-13
    Calling Sequence 2-13
    Special Control Request Calling Sequence Parameters 2-14
    Control Word for Special Control Requests 2-15
Special Read/Write Requests 2-16
    Calling Sequence 2-16
    Special Read/Write Request Calling Sequence Parameters 2-18
    Control Word for Special Read/Write Requests 2-19
DMA Programming 2-20
    Calling Sequence 2-20
    Calling Sequence Parameters 2-21
Error and Status Information 2-22

Chapter 3
Configuration Data

Introduction 3-1
Driver Considerations 3-1
Program Input Phase 3-1
Table Generation Phase 3-2
    Equipment Table Entry (EQT Table) 3-2
    Device Reference Table (DRT Table) 3-2
    Interrupt Table (INT Table) 3-2

Appendix A
Utility Program DSCHD

List of Illustrations

Figure 2-1 General Format of DVM72 Calling Sequences 2-4
Figure 2-2 DVM72 Timeout Processing 2-5
Figure 2-3 Control Word Format, Standard Read/Write Requests 2-7
Figure 2-4 Control Word Format, Read/Write Requests with Subfunction 2-10
Figure 2-5 Control Word Format, Special Control Requests 2-15
Figure 2-6 Control Word Format, Special Read/Write Requests 2-19

Tables

Table 2-1 Standard I/O Sequences 2-2
Table 2-2 Valid Combinations of Function and Subfunction Codes 2-9
Table 2-3 DVM72 Error and Status Information (EQT 5) 2-22
Table 3-1 EQT Table for DVM72 3-3

1

General Information

Introduction

This manual is a programmer's guide to DVM72, the RTE Universal Interface Driver. The driver is callable from HP FORTRAN or HP Macro Assembler Language programs, using RTE EXEC Read, Write, and Control requests. DVM72 is accessible from Multi-User Real-Time BASIC only through Device Subroutines written in FORTRAN or assembler language.

Operating Environment

The following paragraphs define the software and hardware for which the RTE Universal Interface Driver, DVM72, was designed.

Software

DVM72 may be operated within the HP Real-Time Executive Operating System (RTE-IV, RTE-IVB, RTE-6/VM, and RTE-M). The driver requires only one external subroutine, $LIST, the scheduling routine provided by the RTE Operating System.

Hardware

DVM72 can be used with any computer/controller that is capable of supporting the specified software operating system. The memory requirements of DVM72 are approximately 525 (decimal) words.

The following interface cards may be used with DVM72 to control the operation of a variety of programmable instruments. This list is by no means all-inclusive; through experimentation the user may find that a number of other I/O interface cards can be used with this driver.

Model Number	Description
HP 12556B	40-Bit Output Register
HP 12566B	Microcircuit Interface
HP 12604A	General Purpose Data Source Interface
HP 12661A	Digital Voltage Source Program Interface

DVM72 also provides Direct Memory Access (DMA) for use with instruments whose I/O protocol is compatible with HP 21xx DMA requirements.

Components of DVM72

DVM72 is coded as one driver with two entry points: IM72 and CM72. Entry point IM72 provides access to the Initiation Segment; CM72 begins the Continuation Segment of the driver. Both segments share common subroutines and constants contained within the driver. A description of DVM72 follows.

Initiation Segment

The Initiation Segment of DVM72 performs the following functions:

1. Configures I/O instructions with the select code (provided by RTE from the logical unit number in the Control Word, *conwd*).
2. Makes validity checks on passed parameters.
3. Analyzes bits 6 through 10 of the *conwd* to determine the requested function. If bit 9 is set, DVM72 examines the Subfunction Code (Word 1 of IDBUF).
4. Either initiates and completes I/O operations (return to RTIOC with A=4), or initiates only (return with A=0). The latter indicates that further I/O is expected, pending an interrupt from the device.

Completion Segment

The Completion Segment performs the following functions:

1. Configures I/O instructions with select code.
2. If a check for an I/O operation in progress proves false, a check is made for an available interrupt processing routine. If one is found, the routine is scheduled. If no interrupt processing routine is found, the interrupt is ignored and DVM72 makes a continuation return (p + 2) to RTIOC.
3. If an I/O operation is already in progress, the cause of the interrupt and the Function Code determine the course of action. One of the following may occur:
    a. If the entry is due to a timeout, the driver will clear the timeout bit and continue normal I/O operations.
    b. If a DMA operation was in progress, the DMA channel in use is returned to the system (bit 15 of A-Register set).
    c. If the interrupt was expected, normal I/O operations (read or write) continue as specified by Function Code and Subfunction Code.

Driver Limitations

Since it is a “universal” interface for many devices, DVM72 exhibits the following device-independent characteristics:

1. DVM72 does not verify that the device to be programmed is actually connected.
2. DVM72 does not analyze device-dependent status information.
3. DVM72 does not verify that the device is functioning properly.

These and other device-dependent considerations must be handled by device subroutines external to DVM72. The driver can be used, however, to collect the necessary device status and other information for analysis by device subroutines.

Note RTE-6/VM does not support a control request from an extended background program to set up an interrupt-handler program.

In the case of configuring and arming an interrupt-handler program, the request passes the name of the program to the driver through a designated user buffer. However, if the calling program is an extended-background type, the page in memory that contains the user buffer may be remapped by the operating system before entry into the driver.

The call to set up an interrupt handler is supported from any program type other than extended background, and all other driver functions are supported from any program type.

2

Applications

Introduction

This section describes how DVM72 should be called from Assembler and FORTRAN programs. In its general form, this driver uses two buffers declared by the calling program: IDBUF and ICBUF, a data buffer and a control buffer, respectively.

In this section, a number of "codes" are described that control the operation of DVM72. These "codes" are as follows:

Request Code	this is a standard RTE EXEC Call Request Code as described in the Programming and Operating Manual and/or Programmer's Reference Manual for the RTE Operating System. For DVM72, this code may be 1, 2, or 3 for Read, Write, and Control requests, respectively.
Function Code	a general description of the format and purpose of the Function Code, contained in the Control Word (*conwd*) of the EXEC Call, is found in the Programming and Operating Manual and/or Programmer's Reference Manual for the RTE Operating System. Function Codes for DVM72 produce pre-defined I/O sequences, which are described in detail in this section.
Subfunction Code	whenever the Function Code itself cannot specify the I/O sequence of the driver in sufficient detail, a Subfunction Code will be found in Word 1 of the Data Buffer. Subfunction Codes are described in detail within this section.
Command Code	where the standard DVM72 I/O sequences (specified by Function Code only) will not meet the requirements of a device to be programmed, the user may generate his or her own I/O sequence with Command Codes in the Command Buffer (ICBUF). Refer to "Special Control Functions" for further details on Command Codes.

The general format of the DVM72 calling sequences is shown in Figure 2-1.

Standard Read/Write Requests

If the device to be programmed can operate with one of the standard I/O sequences shown in Table 2-1, the user may be able to use one of the standard DVM72 functions to read data from or output data to the device. A Control request may be needed to alter timeout processing or schedule an interrupt program, but the function of the Read/Write request itself can be established in bits 10 through 6 of the control word. Bits 8 and 7 of the control word (function code bits 10 - 6) are shown in column one of Table 2-1. See Figures 2-2 and 2-3 for DVM72 Timeout Processing and Control Word Format, respectively.

Table 2-1. Standard I/O Sequences

Control Word Bits 8/7	Mode	Driver Part	I/O Sequence	Function
00	Read			Interrupt on
		INIT	CLC	arm device
			LIA	read one word after first interrupt
			STC,C	arm device for next interrupt
		CONT	LIA	read one word } for each word
			STC,C	arm device again }
		FINI	CLC	set device and
			STF	I/O card to rest
00	Write			Interrupt on
		INIT	CLC	arm device
			OTA	output first word
			STC,C	arm device for next interrupt
		CONT	OTA	output one word } for each word
			STC,C	arm device again }
		FINI	CLC	set device and
			STF	I/O card to rest
01	Read			Non-interrupt
		INIT	CLC	arm device for data transfer
			LIA	} read into IDBUF until Data Buffer is filled
			LIA	
			LIA	
		FINI	CLC	set device and
			STF	I/O card to rest
				(Returns with A-Reg=4 to indicate completion.)
01	Write			Non-interrupt
		INIT	CLC	arm device for data transfer
			OTA	} write from IDBUF until Data Buffer is empty
			OTA	
		FINI	CLC	set device and
			STF	I/O card to rest
				(Returns with A-Reg=4 to indicate completion.)

Table 2-1. Standard I/O Sequences (continued)

Control Word Bits 8/7	Mode	Driver Part	I/O Sequence	Function
10	Read			Interrupt on
		INIT	CLC STC,C	arm device for data transfer
		CONT	LIA STC,C	read one word arm device again repeat for each word in IDBUF
		FINI	CLC STF	set device and I/O card to rest
10	Write			Interrupt on
		INIT	CLC STC,C	arm device for data transfer
		CONT	OTA STC,C	output one word arm device again repeat for each word in IDBUF
		FINI	CLC STF	set device and I/O card to rest
11	Read			Interrupt initially on
		INIT	CLC STC,C	arm device for initial interrupt
		CONT	LIA LIA LIA STC,C	read into IDBUF (without interrupts) until buffer is full arm device for “done” flag
		FINI	CLC STF	clear device and I/O card
11	Write			Interrupt initially on
		INIT	CLC OTA OTA OTA STC,C	arm device for data transfer output to device from IDBUF (without interrupts) until Data Buffer is empty arm device for “done” flag
		FINI	CLC STF	set device and I/O card to rest

```
CALL EXEC (ICODE,ICNWD,IDBUF,IDBL,ICBUF,ICBL)

          JSB     EXEC
          DEF     RTN          Return Address
          DEF     ICODE        Request Code
          DEF     ICNWD        Control Word
          DEF     IDBUF        Data Buffer Address
          DEF     IDBL         Data Buffer Length
          DEF     ICBUF        Command Buffer Address
          DEF     ICBL         Command Buffer Length
RTN   return point

ICODE     DEC     2    Request Code 2 is WRITE Request
ICNWD     OCT            Function Code plus Logical Unit Number
IDBL      DEC     1      Data Buffer Length is one word only
ICBL      DEC     6      Command Buffer contains six words
IDBUF     DEC     11     Data Buffer contains Subfunction Code 11
ICBUF     DEC     1      Command Code  1 = LIA CHAN Instruction
          DEC     2                    2 = OTA CHAN
          DEC     3                    3 = STC CHAN,C
          DEC     4                    4 = CLC CHAN
          DEC     5                    5 = CLF CHAN
          DEC     6                    6 = STF CHAN
```

Figure 2-1. General Format of DVM72 Calling Sequences

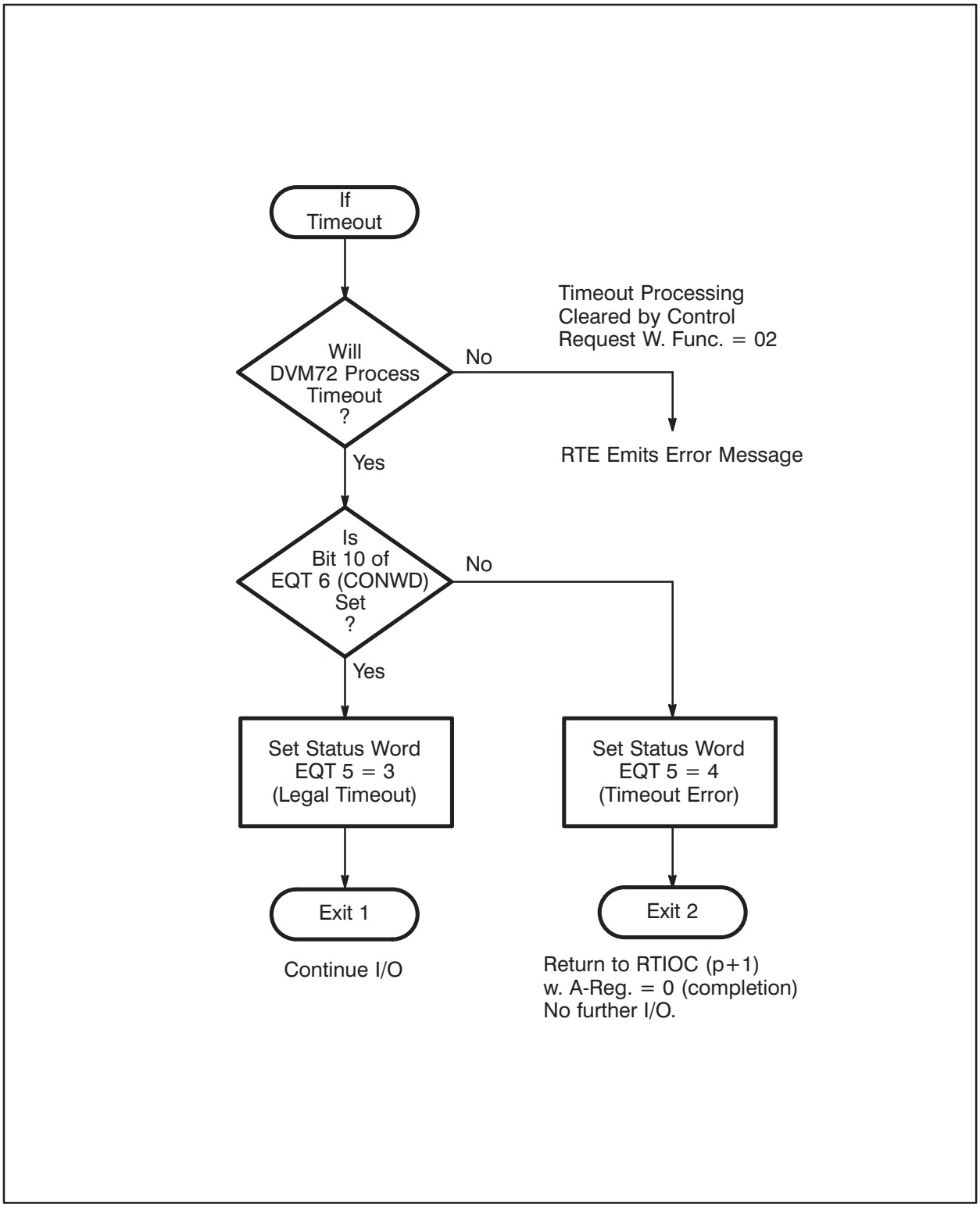


Figure 2-2. DVM72 Timeout Processing

Calling Sequence

The following are general models of standard Read/Write requests.

Assembler Language

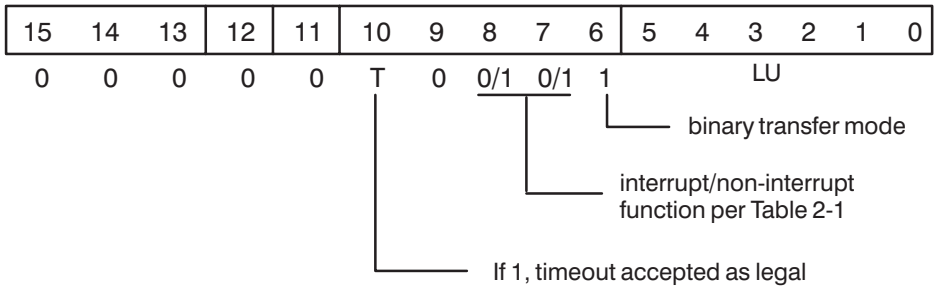
```
            EXT     EXEC
             .
             .
             .
            JSB     EXEC        transfer control to RTE
            DEF     RTN         return address
            DEF     ICODE       request code
            DEF     ICNWD       control information
            DEF     IDBUF       data buffer address
            DEF     IDBL        data buffer length
RTN   return point after execution
             .
             .
             .
ICODE       DEC     1 (or 2)    1 = READ, 2 = WRITE
ICNWD       OCT     conwd       described in following section
IDBUF       BSS     n           buffer of n words
IDBL        DEC     n           same n; number of words in buffer
```

FORTRAN

```
DIMENSION IDBUF(n)              set up data buffer
IDBL=n                          buffer length
ICODE=2 (or 1)                  request code
ICNWD=conwdB                    set control word (B = octal)
REG = EXEC(ICODE,ICNWD,IDBUF,IDBL)
```

Control Word for Standard Read/Write Requests

Figure 2-3 shows the format of the control word (*conwd*) required in the calling sequence for DVM72 driven devices. Several fields defining the nature of the data transfer are shown.



Control Word Bits	Function
Bits 5 - 0	Logical Unit number, which RTE changes to select code before passing to driver.
Bits 10 - 6	Function Code, as defined below:
10	Timeout bit; if set to 1, DVM72 will handle timeout as a legal wait between transfers. If set to 0, DVM72 will flag a timeout as an error. For interrupt operation with wait between transfers, bit 10 must be 1.
9	Request Subfunction bit; if set to 1, Word 1 of the data buffer contains a subfunction code. If 0, no subfunction code exists.
8 - 7	Interrupt/Function bits; defined in Table 2-1.
6	Transfer Mode must be binary, so bit 6 = 1.
Bit 12 (Z-bit)	Used only on special Read/Write requests; for standard requests, this bit should be set to zero at all times.

Figure 2-3. Control Word Format, Standard Read/Write Requests

Read/Write Requests with Subfunction

A slightly more complex form of Read/Write Request uses Word 1 of the Data Buffer (IDBUF) to pass a subfunction code to DVM72. Subfunction codes are used to specify DMA transfers with and without terminating STC CHAN,C instructions (refer to DMA Programming), control functions without data transfers (refer to Special Control Functions), and control functions with data transfers (refer to Special Read/Write Requests). Refer to Table 2-2 for valid combinations of Function and Subfunction Codes.

The least complex form of the Read/Write Requests described above uses DVM72 in interrupt mode to obtain I/O transfers with a fixed time delay between l/O transfers.

Calling Sequence

The calling sequence for DVM72 Read/Write Requests with subfunctions 9, 10, and 15 is shown below. For subfunctions 9 and 10, also refer to the DMA Programming section.

Assembler Language

```
            EXT     EXEC
             .
             .
             .
            JSB     EXEC        transfer control to RTE
            DEF     RTN         return address
            DEF     ICODE       request code
            DEF     ICNWD       control information
            DEF     IDBUF       data buffer address
            DEF     IDBL        data buffer length
RTN  return point
             .
             .
             .
ICODE       DEC     1 (or 2)    1 = READ, 2 = WRITE
ICNWD       OCT     conwd       described in following section
IDBUF       BSS     n           data buffer of n words
IDBL        DEC     n           same n; number of words in data buffer
```

FORTRAN

```
DIMENSION IDBUF(n)              set up data buffer
IDBL=n                          define length of data buffer
ICODE=1 (or 2)                  define request code
ICNWD=conwdB                    set up control word (B = octal)
             .
             .
             .                  establish contents of data buffer
REG = EXEC(ICODE,ICNWD,IDBUF,IDBL)  used as Function
    or
CALL EXEC(ICODE,ICNWD,IDBUF,IDBL)   used as Call
```

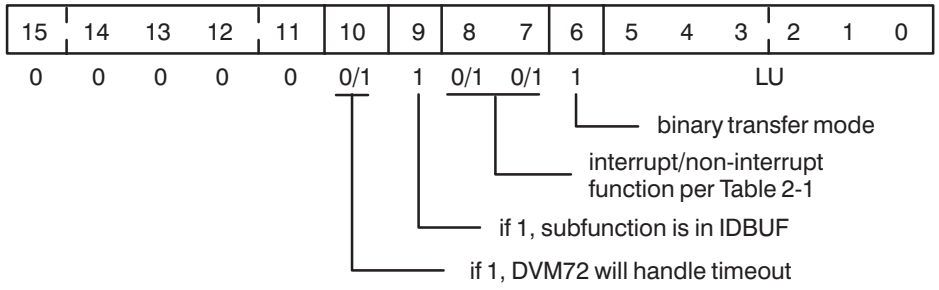
Table 2-2. Valid Combinations of Function and Subfunction Codes

This Subfunction Code	When Used with these Function Code Bits							Produces this Function
	12	11	10	9	8	7	6	
9*	0	0	0	1	0	0	1	DMA Read/Write without STC.
10*	0	0	0	1	0	0	1	DMA Read/Write with STC.
15*	0	0	1	1	0	0	1	Standard Read/Write Request with approximately 500μs delay between transfers.
					1	0	1	
	0	0	0	1	0	1	1	Not valid; desired sequence will be produced only if interrupt function is selected (bit 7 = 0).
					1	1	1	
11†	1	0	0	1	x	x	1	Command sequence in ICBUF with finish in INIT: bits 7 and 8 are ignored.
12†	1	0	0	1	x	x	1	Command sequence in ICBUF with interrupt expected before return to RTIOC, bits 7 and 8 are ignored. Timeout handling as shown in Figure 2-2.
	1	0	1	1	x	x	1	Same except timeout handled by driver.
13†	1	0	0	1	0	0	1	Execute command sequence in ICBUF, wait for interrupt, then transfer data to/from IDBUF. CLC and STC are suppressed; timeout handling as shown in Figure 2-2.
	1	0	0	1	1	0	1	
	1	0	0	1	0	1	1	Same as above, but without waiting for interrupt after command sequence.
	1	0	0	1	1	1	1	
	1	0	1	1	0	0	1	Execute command sequence in ICBUF, wait for interrupt, then transfer data to/from IDBUF. Suppress CLC and STC.
	1	0	1	1	1	0	1	
	1	0	1	1	0	1	1	Same as above, but without waiting for interrupt after command sequence.
	1	0	1	1	1	1	1	
14†	1	0	0	1	0	0	1	Execute command sequence in ICBUF, wait for interrupt, then transfer data with an interrupt at each transfer to/from IDBUF. At the end, issue CLC and STC commands to the device. Timeout is handled by RTE.
	1	0	0	1	1	0	1	
	1	0	0	1	0	1	1	Same as above, but without waiting for interrupt after command sequence.
	1	0	0	1	1	1	1	Same, but wait only for the initial interrupt between command sequence and first data transfer. Thereafter, transfer data without waiting for an interrupt each time. Timeout is handled by RTE.
	1	0	1	1	0	0	1	Same as for corresponding bits 8 - 6 above, except timeout is handled by driver instead of RTE.
	1	0	1	1	1	0	1	
	1	0	1	1	0	1	1	
	1	0	1	1	1	1	1	

x = Don't Care; bits are ignored.
* Does not require ICBUF.
† Requires ICBUF.

Control Word for Read/Write Request with Subfunction

Figure 2-4 shows the format of the control word (*conwd*) required for Read/Write Requests with Subfunction.



Control Word Bits	Function
Bits 10 - 6	I/O Request Function Code, as described below:
10	Timeout bit; must be set to 1 to allow DVM72 to handle the timeout that is expected after each data transfer when subfunction 15 is being used.
9	Request Subfunction bit; set to 1 to indicate that Word 1 of the data buffer contains one of the following subfunction codes: 9 = DMA transfer without STC to start device* 10 = DMA transfer with STC to start device* 15 = Interrupt mode I/O transfer with fixed-time delay between transfers * Refer to DMA Programming

Figure 2-4. Control Word Format, Read/Write Requests with Subfunction

Control Requests

DVM72 Control Requests are available in two formats: Standard Control requests, which perform a limited set of predefined functions, and Special Control requests, which permit the user to set up custom I/O sequences. The latter group of "Control requests" are actually a set of complex Read/Write requests. They are described in detail under "Special Read/Write Requests" (see the Special Read/Write Requests section).

Standard Control Functions

The Standard Control Functions of DVM72 are listed below. Each Control Function is described in detail in the Standard Control Functions section that follows.

Code	Function
01	Set or reset timeout value
03	Define alarm program name*
04	Disarm the alarm program*
05	Arm the alarm program*

* The "alarm program" may be set up from BASIC by using the utility program DSCHD. Refer to Appendix A for further details.

Calling Sequence

The calling sequence for Standard Control Requests is as follows:

Assembler Language

```
              EXT     EXEC
               .
               .
               .
              JSB     EXEC        transfer control to RTE
              DEF     RTN         return address
              DEF     ICODE       request code
              DEF     ICNWD       control information optional parameter
                                    (see Function Code descriptions on next page)
              DEF     IPRAM       optional parameter
                                    (see Function Code descriptions on next page)
RTN   return point                returns here after execution
               .
               .
               .
ICODE         DEC     3           3 = CONTROL request
ICNWD         OCT     conwd       described in following section
IPRAM         DEF     *+1         define address of interrupt program name
              ASC     3,PROGN     set name of interrupt program
                 or
IPRAM         DEC     time        set new timeout value
```

FORTRAN

```
*IPRAM = address of interrupt          via user-written Assembler
           program name                Language routine
    or
IPRAM = timeout value                  set new timeout value
   .
   .
   .
ICODE = 3                              request code
ICNWD = conwdB                         set control word (B = octal value)
   .
   .
   .
*REG=EXEC(ICODE,ICNWD,IPRAM)           used as Function
    or
*CALL  EXEC(ICODE,ICNWD,IPRAM)         used as Call
```

* The interrupt program may be scheduled from FORTRAN using the utility program DSCHD. This is done by deleting the IPRAM statement and replacing the EXEC calls with: "`CALL DSCHD (LU,3,IPROG)`" where IPROG is the interrupt program name. Refer to Appendix A for further details.

Standard Control Functions

DVM72 will accept the following Function Codes in bits 10 through 6 of the Control Word:

01 Reset the established timeout value for the driver. This function sets a new timeout value, as defined by IPRAM in tens of milliseconds, into EQT14. When bit 10 of the *conwd* is set during subsequent I/O requests, this function may be used to implement a software wait between I/O transfers.

03 Set up the interrupt program defined by IPRAM for scheduling by a subsequent interrupt. This function does not schedule the program; it only places the ID segment address of the program into EQT13. IPRAM must contain the address of the alarm program's name. This function is not supported from an RTE-6/VM extended-background program.

04 Disarm the defined alarm program. If an alarm program has been set up and armed (Control Function 05), it must be disabled before further I/O to the controlled device may take place. If the defined alarm program is not first disarmed, any interrupt caused by normal I/O operations could schedule the program for execution. IPRAM is not used with this function code.

05 Arm the defined alarm program. Before a program defined by Control Function 03 can respond to an interrupt, it must be enabled or "armed". This function is also used to re-arm a program that was temporarily disarmed for I/O operations (Control Function 04). IPRAM is not used with this function code.

Special Control Requests

To meet the special code sequence requirements of some instruments' I/O protocol, DVM72 provides the user with a means of generating custom I/O instruction sequences. These custom instruction sequences do not transfer any data to/from the programmed device. The only instruction capable of sending data (OTA) has a zero in the A-Register at the time of execution. The configured I/O instructions provided by DVM72 are defined as follows:

1	LIA CHAN	Load into A-Register from I/O channel CHAN
2	OTA CHAN	Output from A-Register from I/O channel CHAN
3	STC CHAN,C	Set I/O control bit and clear I/O flag on CHAN
4	CLC CHAN	Clear the I/O control bit of select code CHAN
5	CLF CHAN	Clear the I/O flag on select code CHAN
6	STF CHAN	Set the I/O flag on select code CHAN

Each configured I/O instruction is assigned an integer command code in the range of 1 to 6. Configured I/O instructions may be called in any sequence and as often as required by placing the proper command codes (not the I/O instructions themselves) into the command buffer in the desired sequence. Note that the command buffer will not contain any data words or subfunction codes; it should contain only integers between 1 and 6, one integer per word.

Calling Sequence

The calling sequence for Special Control requests is as follows:

Assembler Language

```
            EXT     EXEC
              .
              .
              .
            JSB     EXEC        transfer control to RTE
            DEF     RTN         return address
            DEF     ICODE       request code
            DEF     ICNWD       control information
            DEF     IDBUF       data buffer address
            DEF     IDBL        data buffer length
            DEF     ICBUF       command buffer address
            DEF     ICBL        command buffer length
RTN   return point
              .
              .
              .
ICODE       DEC     2 (or 1)    2 = WRITE, 1 = READ
ICNWD       OCT     conwd       described in following section
IDBUF       BSS     n           data buffer of n words
IDBL        DEC     n           same n; number of words in data buffer
ICBUF       BSS     m           command buffer of m words
ICBL        DEC     m           same m; number of words in command buffer
```

FORTRAN

```
DIMENSION IDBUF(n),ICBUF(m)     set up both buffers
IDBL=n                          define length of data buffer
ICBL=m                          define length of command buffer
ICODE=2 (or 1)                  request code
ICNWD=conwdB                    set up control word (B = octal)
  .                             establish contents of data buffer (see next section)
  .                             establish contents of command buffer
  .                             (see next section)
REG = EXEC(ICODE,ICNWD,IDBUF,IDBL,ICBUF,ICBL)
```

Special Control Request Calling Sequence Parameters

ICODE RTE EXEC Request Code; must be one of the following: 1 = READ request, or 2 = WRITE request

ICNWD Control Word, see Figure 2-5.

IDBUF Data Buffer containing one of the following subfunction codes in Word 1 of the buffer:

11 = Execute the command sequence stored in ICBUF, then finish in the Initiation Section of DVM72.

12 = Execute the command sequence stored in ICBUF, then wait for an interrupt before returning to RTIOC.

IDBL Data Buffer Length; for subfunction codes 17 and 12 (no data transfer, only I/O instructions), IDBL must be set to one. Only Word 1 is used for the subfunction code.

ICBUF Command Buffer containing integer command codes that correspond to the following I/O instructions. Each I/O instruction is configured with the select code (CHAN) of the device specified by LU in the Control Word.

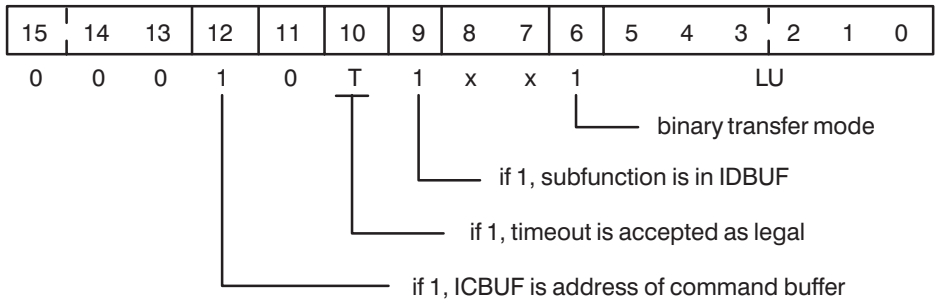
Command Word	I/O Instruction
1	LIA CHAN
2	OTA CHAN
3	STC CHAN,C
4	CLC CHAN
5	CLF CHAN
6	STF CHAN

Command codes in ICBUF may be used in any order and as many commands as may be required can be executed at any one time. It is the user's responsibility to know and understand the programming requirements of the instrument to be controlled.

ICBL Command Buffer Length in words; one word per command code in ICBUF.

Control Word for Special Control Requests

Figure 2-5 shows the format of the control word (*conwd*) required in the calling sequence for DVM72 special Control requests. Note that although the requests in this category are classified as "Control requests", they are in fact special types of Read/Write requests (Request Code is 1 or 2). Although no data is transferred, the data buffer IDBUF is required to contain the subfunction code that defines this type of request to the driver.



Control Word Bits	Function
Bits 5 - 0	Logical Unit Number, changed to select code by RTE before it is passed to the driver.
Bits 10 - 6	Function Code, as defined below:
10	Timeout bit; if set to 1, DVM72 will handle timeout as a legal wait between commands. If set to 0, DVM72 will flag a timeout as an error. For interrupt operation (e.g., subfunction = 12) bit 10 must be set to 1.
9	Request Subfunction bit; if set to 1, Word 1 of the data buffer contains a subfunction code. For this type of request, one of the subfunction codes described below must be provided.
8 - 7	Interrupt/Function bits; for this type of request, these bits are ignored.
6	Transfer Mode must be binary, so bit 6 = 1.
12	Parameter/Address bit for optional parameter; if set to 1, as it must be for this type of request, it indicates to RTE and DVM72 that the entry ICBUF is an address instead of a parameter. In this case, ICBUF is the address of the command buffer containing a special I/O instruction sequence.

Figure 2-5. Control Word Format, Special Control Requests

Special Read/Write Requests

For instruments with special I/O sequence requirements, command sequences in ICBUF may be combined with data transfers to/from the Data Buffer, IDBUF. This custom commands plus data transfer capability of DVM72 can be combined with the extended function codes of the Control Word (*conwd*) to satisfy the I/O requirements of practically any programmable instrument.

Using Special Read/Write Requests to DVM72, the user can combine any of the following capabilities:

1. Read or Write data in interrupt or non-interrupt mode
2. Transfer data only after an initial interrupt
3. Programmed delay between I/O transfers
4. Custom I/O instruction sequences before or after data transfers.

The calling sequences for Special Read/Write Requests are, on the surface, very similar to the calling sequences for Special Control Requests. The differences are that the Data Buffer Length (IDBL) is greater than one, since the I/O request includes data transfer, that different subfunction codes are expected in Word 1 of IDBUF (13 and 14 instead of 11 and 12), and several bits in the CONWD (bits 7 and 8) are no longer ignored as they were with Special Control Requests.

Calling Sequence

The calling sequence for Special Read/Write Requests is as follows. Note that this sequence could be described as the "general form" of the sequence introduced by Special Control Requests.

Caution Custom I/O instruction combinations, timing and data transfer sequences can be issued in practically any combination defined by the user. It is the programmer's responsibility to become thoroughly familiar with the I/O instruction and timing requirements of the device to be programmed to avoid loss of data or possible damage to the device.

Assembler Language

```
            EXT     EXEC
             .
             .
             .
            JSB     EXEC        transfer control to RTE
            DEF     RTN         return address
            DEF     ICODE       request code
            DEF     ICNWD       control information
            DEF     IDBUF       data buffer address
            DEF     IDBL        data buffer length
            DEF     ICBUF       command buffer address
            DEF     ICBL        command buffer length
RTN  return point
             .
             .
             .
ICODE       DEC     1 (or 2)    1 = READ, 2 = WRITE
ICNWD       OCT     conwd       described in following section
IDBUF       BSS     n           data buffer of n words, including subfunction code
IDBL        DEC     n           same n; number of words in data buffer
ICBUF       BSS     m           command buffer of m words
ICBL        DEC     m           same m; number of words in command buffer
```

FORTRAN

```
DIMENSION IDBUF(n),ICBUF(m)     set up both buffers
IDBL=n                          define length of data buffer
ICBL=m                          define length of command buffer
ICODE=1 (or 2)                  request code
ICNWD=conwdB                    set up control word (B = octal)
  .                             establish contents of data buffer (see next section)
  .                             establish contents of command buffer
  .                             (see next section)
REG = EXEC(ICODE,ICNWD,IDBUF,IDBL,ICBUF,ICBL)
  or
CALL  EXEC(ICODE,ICNWD,IDBUF,IDBL,ICBUF,ICBL)
```

Special Read/Write Request Calling Sequence Parameters

ICODE RTE EXEC Request Code; must be one of the following: 1, for READ requests, or 2 for WRITE requests

ICNWD Control Word, see Figure 2-6.

IDBUF Data Buffer containing one of the following subfunction codes in Word 1 of the buffer:

13 = Special command codes in ICBUF are executed, then data is to be transferred from/to the controlled device to/from IDBUF. Upon completion of I/O, in either the Initiator or the Continuator, the normal sequence of CLC,STF will be suppressed.

14 = Identical to subfunction 13, except terminating I/O sequence of CLC,STF will be executed upon completion of data transfer.

IDBL Data Buffer Length; must be one word longer than the number of data words to be transferred, since Word 1 of the Data Buffer contains the subfunction code.

ICBUF Command Buffer containing integer command codes that correspond to the following I/O instructions. Each I/O instruction is configured with the select code (CHAN) of the device specified by LU in the Control Word.

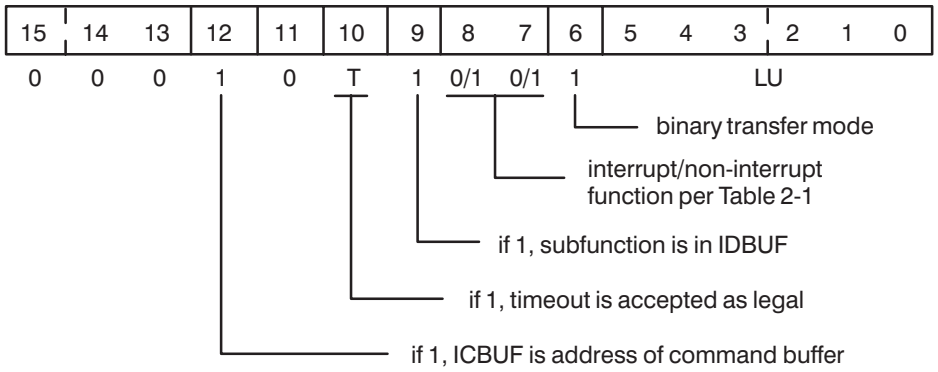
Command Code	I/O Instruction
1	LIA CHAN
2	OTA CHAN
3	STC CHAN,C
4	CLC CHAN
5	CLF CHAN
6	STF CHAN

Command codes in ICBUF may be used in any order and as many commands as may be required can be executed at any one time. It is the user's responsibility to know and understand the programming requirements of the instrument to be controlled.

ICBL Command Buffer Length in words; one word per command code in ICBUF.

Control Word for Special Read/Write Requests

The format of the control word (*conwd*) for Special Read/Write Requests is shown in Figure 2-6. A command buffer, ICBUF, is required to store any special I/O sequences that may be needed; the data buffer, IDBUF, is required for the subfunction code and all data to be transferred to/from the programmed device.



Control Word Bits	Function
Bits 5 - 0	Logical Unit Number, changed by RTE to select code before passing to driver.
Bits 10 - 6	Function Code, as defined below:
10	Timeout bit; if set to 1, DVM72 will handle timeout as a legal wait between transfers. If set to 0, DVM72 will flag a timeout as an error. For interrupt operation with a wait between transfers, bit 10 must be 1.
9	Request Subfunction bit; to get subfunction code from Word 1 of IDBUF, bit 9 must be 1.
8 - 7	Interrupt/non-interrupt Function bits; refer to Table 2-1.
6	Transfer Mode must be binary; bit 6 must be 1.
Bit 12 (Z-bit)	Parameter/Address bit for optional parameter; if set to 1, as it must be for this type of I/O function, bit 12 indicates to RTE and DVM72 that the entry ICBUF is an address instead of a parameter. In this case, ICBUF is the address of the command buffer containing a special I/O instruction sequence.

Figure 2-6. Control Word Format, Special Read/Write Requests

DMA Programming

DVM72 provides the user with the capability to handle I/O transfers via Direct Memory Access (DMA) channels if the device to be programmed offers the I/O protocol necessary for DMA processing.

Calling Sequence

The calling sequences for DMA Read/Write requests are as follows:

Assembler Language

```
            EXT     EXEC
            JSB     EXEC            transfer control to EXEC
            DEF     RTN             return address
            DEF     ICODE           request code
            DEF     CONWD
            DEF     IDBUF
            DEF     IDBL
RTN  return point
ICODE       DEC     1 (or 2)
ICNWD       OCT     11LU
IDBL        DEC     n+2
IDBUF       DEF     *+1
WD1         DEC     9 or 10         first DMA control word
WD2         OCT     0 or 100000B    second DMA control word
```

WD*n*

FORTRAN

```
CALL  EXEC(ICODE,CONWD,IDBUF,IDBL)
```

Calling Sequence Parameters

ICODE Request Code: 1 = Read Request, 2 = Write Request.

CONWD Control Word with Function Code plus Logical Unit number. For DMA transfers, the Function Code is 11B (bits 9 and 6 must be set to 1).

IDBL Data Buffer Length in number of 16-bit words. If *n* words are to be transferred, IDBL must be set to *n* + 2 to include the first two words containing subfunction code and DMA control word.

IDBUF Data Buffer of length specified by IDBL.

WD1 The first two words contain a subfunction code and DMA control code as defined below.

Word 1

9	DMA transfer without STC CHAN,C to start the device.
10	DMA transfer with STC CHAN,C to start the device.

WD2 The second word in the Data Buffer must contain the first DMA control word (CW1).

Word 2

0	No STC CHAN,C is to be issued after each DMA transfer.
100000B (Bit 15 set)	DMA is to issue an STC CHAN,C at the end of each DMA cycle, except on the last cycle of a Read Request.

Error and Status Information

Status information on DVM72 is contained in Words 4 and 5 of the Equipment Table Entry and, upon return from the driver, in the A- and B-Registers. This status data can be obtained through an RTE I/O Status Request (EXEC 13) or by using the EXEC Function as shown in the example below.

```
DIMENSION IREG(2)
EQUIVALENCE (REG,IREG,IA), (IREG(2),IB)
   .
   .
   .
REG = EXEC(ICODE,p2,...,pn)
IA = IAND(IREG,377B)
```

Upon return from this EXEC function, the B-Register (IB or IREG(2)) will contain the transmission log. The status information contained in bits 7 through 0 of the A-Register (EQT word 5) is defined by Table 2-3.

Table 2-3. DVM72 Error and Status Information (EQT 5)

Value in Bits 7 - 0	Meaning
0	No error.
1	Illegal subfunction code.
2	Timeout after 500-μsec delay (subfunction = 15).
3	Legal timeout (end of programmed delay) between I/O transfers.
4	Timeout error; device failed to interrupt during allowed time and bit 10 of CONWD was not set.
5	ID segment does not exist. Interrupt program defined by IPRAM could not be located.
6	Illegal instruction code in command buffer.
7	An alarm or interrupt program is currently scheduled.

3

Configuration Data

Introduction

This section provides configuration information for the Universal Interface Driver DVM72 and is intended to augment the data provided in the RTE Operating System Programming and Operating Manual and/or Programmer's Reference Manual. The software to be configured into an RTE System includes the following:

09580-16079	DVM72	Universal Interface Driver
(part of RTE)	$LIST	RTE Scheduling Routine

Driver Considerations

The RTE Operating System Programming and Operating Manuals, Programmer's Reference Manuals, and Online Generator Reference Manuals divide the process of System Installation (that is, System Generation) into phases with headings appropriate to the required operations. The following headings for DVM72 correspond to those parts of the System Installation with identical headings in the RTE manuals.

Program Input Phase

Load DVM72 into the system during this phase as you would load any other I/O Driver.

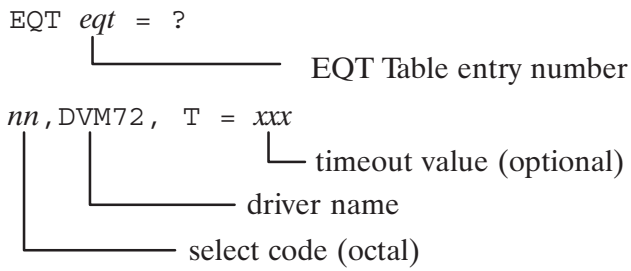
Table Generation Phase

This phase is divided into three parts: the Equipment Table, the Device Reference Table, and the Interrupt Table. DVM72 requires entries in each of these tables.

Equipment Table Entry (EQT Table)

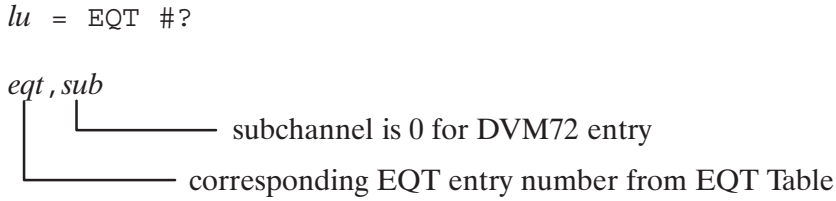
1. Determine the select code of the I/O slot for the device.
2. Unless special circumstances prevail, do not use the output buffering option "B".
3. Do not specify the "D" option (DMA required). DVM72 will request RTIOC to dynamically assign a DMA channel whenever one is required.
4. If a permanent default timeout value is desired, specify the new value with the "T=ttttt" option (where "ttttt" represents the timeout value in tens of milliseconds). This timeout value may also be temporarily reset by a Control Request or an RTE System Command.
5. DVM72 does not require an EQT extension. (Refer to Table 3-1 for DVM72's EQT Table.)

A typical response during Equipment Table generation could look like the following:



Device Reference Table (DRT Table)

The Device Reference Table contains a cross-reference of logical unit (LU) numbers to EQT entry and subchannel numbers. A typical entry for DVM72 might appear as follows:



Interrupt Table (INT Table)

This table establishes interrupt links that tie the octal select codes back to EQT numbers. For DVM72, a typical generation response follows:

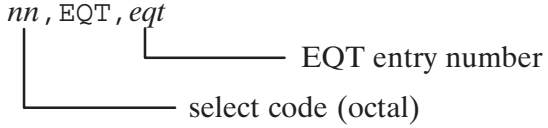


Table 3-1. EQT Table for DVM72

EQT1	Device Suspended List Pointer to ID Segment of the calling program. If REIO is used in the CALL, this pointer is to an ID Segment created by EXEC.
EQT2	Driver Initiation Section Entry Point Address.
EQT3	Driver Continuation Section Entry Point Address.
EQT4	Driver/Device Data as follows: **Bits** — **Normal Setting and Meaning** 15 = 0 DMA is dynamically assigned when needed. 14 = 0 Automatic output buffering is not used. 13 = 0 Driver will not process powerfail. 12 = 1 If driver is to process timeout. 11 = 1/0 Device did/did not time out (system sets this bit to zero before each I/O request). 10 - 6 = 0 Last subchannel addressed. 5 - 0 = I/O select code for device.
EQT5	Availability Indicator in bits 15 - 14. Equipment Type Code (72B) in bits 13 - 8. Status Code upon completion of operation in bits 7 - 0.
EQT6	CONWD = user control word from I/O EXEC call.
EQT7	Data Buffer Address (IDBUF). IPRAM if Control request.
EQT8	Data Buffer Length (IDBL).
EQT9	Temporary Storage for Optional Parameter. If bit 12 of CONWD is set, EQT9 contains the address of the Command Buffer (ICBUF).
EQT10	Temporary Storage tor Optional Parameter. EQT10 serves as a word counter for both standard and special Read/Write requests.
EQT11	Driver Storage; contains terminal interrupt flag if bit 8 of CONWD is set.
EQT12	Driver Storage; contains Subfunction Code from Word 1 of IDBUF if I/O request is a special Read/Write request.
EQT13	Driver Storage; contains address of ID Segment used for interrupt processing (Alarm Program address).
EQT14	Device Timeout Reset Value.
EQT15	Device Timeout Clock.

A

Utility Program DSCHD

DSCHD

Purpose: BASIC language interface to call EXEC 3. Selects the interrupt program defined by IPROG for scheduling by a subsequent interrupt. DSCHD itself does not schedule the program; it only places the ID segment address of the program into EQT13.

Program Type: 7

Externals: EXEC, .ENTR

Method: The BASIC and FORTRAN calls to DSCHD are converted to an assembly language call to EXEC, using the parameters of the original call.

Calling Sequence:

BASIC

```
CALL DSCHD (L,C,"PRNAM")
```

FORTRAN

```
CALL DSCHD(LU,ICODE,IPROG)
```

where:

`L = LU` = Logical Unit of device driven by DVM72.

`C = ICODE` = Control Request Code, normally = 3.

`"PRNAM" = IPROG` = Address of a buffer containing a five-character program name. If the name is less than five characters, use blanks to extend it to five. The first word of the buffer contains the number of characters in the name.

The buffer must reside in a non-swappable area of the system memory map.

In RTE-6/VM, do not call DSCHD from an extended-background program. The page containing the buffer (at IPROG) may become remapped before entry into the driver.