M21 Devices

M21 Devices

Table of Contents

Chapter 1. Overview

This chapter describes the physical and logical peripheral devices that are supported by M21. It describes how to use each type of device and how to control the features and functions of the device.

M21 supports a variety of physical and logical devices. **Physical devices** include terminals, modems, printers, and sequential devices such as floppy disks and tape drives. **Logical devices** are used to read and write host operating system files, access networking, etc.

Every M system user is able to access any of the system devices. To access a device, the job must first obtain ownership of the device by issuing an `OPEN` command for the device. If the device is available, it is added to the list of devices owned by the job. When a device is `OPEN` ed, it is possible to specify, as part of the `OPEN` command, the amount of time to wait if the device is not immediately available. If the device is unavailable, then execution of the job will be suspended until the device becomes available if no timeout is specified, or will return after the timeout expires if one is specified as part of the `OPEN` command. The `OPEN` mechanism permits a job to gain ownership of more than one device at a time.

Since each job can access only one device at a time, the job needs to direct input and output operations to a specific device with the `USE` command. The `USE` command makes the specified device the current device, which will be used for all subsequent input or output operations until explicitly changed by another `USE` command or until the job terminates. After ownership and use of a device are established, the job can read from or write to the device using the M(UMPS) `READ` and `WRITE` commands. When a device is no longer needed by a job, a `CLOSE` command issued to the device will return it to the system and make it available for other jobs (if it is a shared rather than a private device).

After a device is returned to the system with a `CLOSE` command, it must be acquired with an `OPEN` command before it can be `USE` d again. When a terminal-type device is `OPEN` ed, device characteristics established through previous `OPEN` or `USE` commands are automatically reset to the default values specified by the default system configuration.

Devices can either be private to an M process or shared across a running M21 system. When a device is shared it means that only one job can open the device identifier and any other job trying to open the device will be suspended until the device becomes available or any specified timeout expires. In contrast, any job can open a private device identifier, even if the same identifier has already been opened by other active jobs.

Device Identifiers

Every device in M21 is assigned a unique numeric device identifier, which is used as an argument to the `OPEN` , `USE` and `CLOSE` commands to identify a specific device. The following table lists the device identifiers for each device type:

Table 1-1. Table of Device Identifiers

Identifiers	Description	Private to M Proces
1-6999	Terminal devices	

Identifiers	Description	Private to M Process
7000-7049	Magnetic tape devices	
7100	VIEW device for disk operations	
8000-8049	Host file devices	Yes
8050	Null device	Yes
8070-8079	Inter job communication/pipe devices	Yes
8100-8499	Host system spooling device	Yes
9050-9099	TCP/IP and UDP socket devices	Yes
9200-9299	Routine interlock devices	

Accessing M21 Devices

When a user logs on to M21, a new job is created and assigned a unique number. This number is stored in the `$JOB` Special Variable. M21 then assigns ownership of the log on terminal to the job as if it had been explicitly `OPEN` ed. The device identifier that corresponds to the terminal is stored in the `$IO` Special Variable.

The **$IO** Special Variable will always contain the device identifier of the current device. All input or output operations (for example: **READ**, **WRITE**, **(Z)PRINT**) will be directed to the current device. The initial value assigned to the **$IO** Special Variable at logon is referred to as the principal device and is stored in the **$PRINCIPAL** Special Variable. Any reference to a device identifier of 0 (zero) is interpreted to mean the principal device.

A job can own more than one device. Ownership of each device is achieved by issuing an `OPEN` command with one or more device identifiers as the operand of the command. After a job obtains ownership of a device, the device is no longer available to other jobs if it is a device that is shared across the system. Devices which are private to a process and not shared as indicated in the table above will still be available to other jobs. An exception to this rule are terminal devices, which can be used for output only by another process by means of the `ZUSE` command.

The operand of the `USE` command is the identifier of the device that is to be used as the current device. The device must have been previously `OPEN` ed. The device identifier will then become the new value of the `$IO` Special Variable and will remain the current device until changed by another `USE` command. The `$IO` Special Variable can be referenced to determine the current device, but can only be changed with the `USE` command.

If M21 detects an error during routine interpretation, it will internally issue an `OPEN` and `USE` command with a value of zero before displaying the error message. Output will then be directed to the principal device to ensure that the error message is displayed on the device that initiated execution of the job. This will also happen when a program finishes normally and returns to the programmer prompt.

When a job no longer needs to use a specific device, it should issue a `CLOSE` command to the device. The `CLOSE` command will return the device to the system and allow it to be assigned to other jobs if necessary. When a job finishes execution with a `HALT` command, then M21 will automatically `CLOSE` all devices that are still open.

In M21, all of the devices, except the VIEW and Routine Interlock devices, appear like sequential devices to the process using them. For this reason, the same input and output commands can be used regardless of the device type. M21 supports the following input and output commands:

`READ` - Requests input from a device. The information returned by the device may be stored in a local or global variable. Optionally, the command can include a prompt string and formatting characters.

`WRITE` - Sends information to a device. It can include format characters, device mnemonics, ASCII values, literals, and local or global variables.

`ZLOAD` - When issued with no operands, transfers a routine from the current device to the partition.

`ZPRINT` or `PRINT` - Writes the routine currently loaded in the partition to the current device.

`ZWRITE` or `WRITE` - Writes the contents of all or part of the local variable table to the current device.

Special Variables

For each device, M21 maintains the following special variables to provide feedback information to the process about the status of the current device.

`$X` Contains the cursor position relative to the left margin. M21 only counts ASCII characters between the decimal values of 32 and 126, inclusive, and all 8-bit ASCII values. If terminal-specific escape sequences are used to move the cursor, `$X` will no longer represent the true cursor position unless it is explicitly updated using the `SET` or `USE` commands. When the value of `$X` exceeds 255 characters, it will be automatically reset to zero.

`$Y` Contains the number of line-feed characters that were written to the device since the last form feed. When the value of `$Y` exceeds 255 characters, it will be automatically reset to zero.

`$ZA` Contains information specific to the device being accessed. It can contain error flags, status information, or general information, dependent on the device type.

`$ZB` Contains additional information specific to the device being accessed. It can contain block numbers, buffer offsets, or other information, dependent on the device type.

`$ZC` Contains additional information that is common to all devices in M21. It indicates whether an end-of-file condition or error condition occurred.

Mnemonic Namespaces

The Mnemonic Namespaces feature allows the development of applications that are independent of the type of device that they are being run on and the device's implementation of common functionality e.g. positioning the cursor or clearing the screen.

Mnemonic Namespaces can be used when accessing Terminals, Host System Spooling devices, TCP/IP and UDP Sockets and the Null device.

Mnemonics are specified on the `READ` or `WRITE` command in the same way as format arguments like `?` and `#` as follows:

```
WRITE /MnemonicName(Param1,Param2, ... ,ParamN)
```

The slash (`/`) is required, but the parameters can be optional or mandatory dependent upon the way that the individual mnemonic is set up.

Mnemonics are arranged into groups to form a Mnemonic Namespace and identified by a domain name. These domains are either system-supplied e.g. `TCP` for sockets, or user-defined. When a device is opened a list of domain names can be specified as a parameter to the `OPEN` command and the first of these will become the default domain for the device. The `USE` command will subsequently switch between the domain names specified as part of the `OPEN` command. When an M job is started the `OPEN` on the principal device is implicit and hence a domain name list cannot be specified. Since `USE` can only switch between domain names that are specified on the `OPEN` command a way is needed to specify domains to the principal device as if they were specified on the `OPEN` . This is achieved using the configuration file parameter `DEFAULT_DOMAINS` , which is use to specify a list of domain names accessible by the job's principal device.

User-defined domains and mnemonics are specified in an external configuration file (**xxxxx.mnemonicnamespaces**), where **xxxxx** is the database name. This file needs to be placed in the directory where the database files reside. It consists of two types of entries, the first (keyword `domain`) specifies a domain name and the second (keyword `mnemonic`) specifies a mnemonic within a domain, its parameters and the M code that will be executed when this mnemonic is specified. Mnemonics need to be specified in the configuration file with their names in ascending alphabetical order. To include comments in the configuration file prefix them with " `##` ". The syntax for the two keywords is as follows:

```
domain Name
```

Name is the identifier for this domain

```
mnemonic Name(min=val1 max=val2) M Code
```

Name is the identifier for this mnemonic

val1 is the minimum number of parameters that must be specified when using this mnemonic

val2 is the maximum number of parameters that can be specified when using this mnemonic

M Code is the M code to execute when this mnemonic is used. Parameters to the mnemonic are specified as `$1` , `$2` etc. and the supplied values are substituted at execution time.

The following example configuration file illustrates how domains and mnemonics are specified:

```
## Specify test domain
domain test
## mnemonics in alphabetic order
```

```
mnemonic bl(min=0 max=0)  w *27,"[8m"
mnemonic cub(min=0 max=0) w *8
mnemonic cud(min=0 max=0) w *10
mnemonic cuf(min=0 max=0) w *27,"[1C"
mnemonic cuoff(min=0 max=0) w *27,"[?25l"
mnemonic cuon(min=0 max=0) w *27,"[?25h"
mnemonic cup(min=2 max=2) w *27,"[",$1,";",$2,"H"
mnemonic devt(min=0 max=1) u 0:::$1
mnemonic dummy(min=0 max=0) w ""
mnemonic test(min=0 max=0) w "Hello World"
mnemonic ul(min=0 max=0)  w *27,"[33m"
```

Configuration File Parameters

Associated with domains and mnemonics are a series of parameters. These are specified in the external configuration file (**xxxxx.cfg**), where **xxxxx** is the database name. This file can be found in the directory where the database files reside and will have been created by the database initialisation program **dbinit** . None of the domains and mnemonic parameter values are available using the `$ZINFO()` command since they are not needed to load the externally defined domains. For more information on these parameters see the documentation relating to the configuration file. Domains and mnemonics are loaded when the M system is started up if the appropriate parameter is specified in the configuration file.

AVG_DOMAIN_NAME_SIZE

To minimize wasted memory space for variable length domain names there is a Domain Name Array which is sized at

MAX_DOMAINS * AVG_DOMAIN_NAME_SIZE

Each domain name is entered as a NULL terminated string at the end of the used portion of the array.

AVG_MNEMONIC_NAME_SIZE

To minimize wasted memory space for variable length mnemonic names there is a Mnemonic Name Array which is sized at:

MAX_DOMAINS * AVG_MNEMONICS_PER_DOMAIN * AVG_MNEMONIC_NAME_SIZE

Each mnemonic name is entered as a NULL terminated string at the end of the used portion of the array.

AVG_MNEMONIC_VALUE_SIZE

To minimize wasted memory space for variable length mnemonic values there is a Mnemonic Value Array which is sized at

MAX_DOMAINS * AVG_MNEMONICS_PER_DOMAIN * AVG_MNEMONIC_VALUE_SIZE

The mnemonic values will be a null terminated M(UMPS) routine entry point string.

AVG_MNEMONICS_PER_DOMAIN

Used to size the Mnemonic Table in memory to:

MAX_DOMAINS * AVG_MNEMONICS_PER_DOMAIN

DEFAULT_DOMAINS

A comma separated list of domains that will be available to the partition's principal device. This is necessary because the `OPEN` on a principal device is implicit and hence the domain list cannot be specified. The first entry in the list will be the current domain for the principal device.

IPL_LoadMnemonicNamespaces

Load the mnemonic namespaces specified in the configuration file **database.mnemonicnamespaces** when the M system is first started.

MAX_DOMAINS

This gives the maximum number of domains that may be defined (internal + external).

MAX_DOMAINS_PER_DEVICE

Gives the maximum number of domains that may be associated with a single device in an `OPEN` command. Any further domains in the domain list of the `OPEN` are ignored.

Using Domains and Mnemonics

Specifying Domains On the OPEN Command

When **OPEN** ing a device, a list of domain names to be available for use by the device can be supplied as the fourth parameter. For example:

OPEN 8100:::"VT220,ANSI,PRINTER"

will open a host spool device and make available for use the domains **VT220** , **ANSI** and **PRINTER** . **VT220** will be the currently selected domain following the **OPEN** .

Selecting a Domain With the USE Command

When **USE** ing a device, the name of translation table to be enabled can be supplied as the third parameter. For example:

USE 0::"ANSI"

will make the domain **ANSI** the current domain for the principal device.

Device Translation

Device translation is a facility that allows both input and output characters to and from any device to be mapped to other values to compensate for the variations in national character sets and terminal devices.

To facilitate device translation, a group of named device translation tables can be loaded into memory when starting the M system. Subsequently, in order to enable device translation, the name of a pre-loaded translation table can be supplied as a parameter to either the OPEN or USE commands. The exact syntax of this is detailed below.

Device translation tables are specified in an external configuration file (**xxxxx.devicetranslations**), where **xxxxx** is the database name. This file needs to be placed in the directory where the database files reside. It consists of a name for the table with a maximum length of eight characters followed by two lists, the first list specifies the character translations on output (ASCII code followed by " -> "

followed by ASCII code to translate to) and the second list specifies the character translations on input (ASCII code followed by " <- " followed by ASCII code to translate to). To include comments in the configuration file prefix them with " ## ".

The following example configuration file illustrates how device translations are specified:

```
## Test Device Translation Table
## Name up to 8 characters
name test

## Translations on output
81 -> 59
113 -> 59
65 -> 193
66 -> 194
71 -> 195
68 -> 196
69 -> 197
90 -> 198
72 -> 199
85 -> 200

## Translations on input
81 <- 59
65 <- 193
66 <- 194
71 <- 195
68 <- 196
69 <- 197
90 <- 198
72 <- 199
85 <- 200
```

Configuration File Parameters

MAXDEVTRANTABS

This parameter is used to specify the maximum number of device translation tables that can be held in memory. If this parameter is not defined in the database.cfg file, a default value of 8 will be used. A maximum value of 16 is also imposed.

IPL_LoadDeviceTranslations

Load the device translation tables specified in **database.mnemonicnamespaces** when the M system is first started.

Using Device Translation

Enabling a Table With the OPEN Command

When OPEN ing a device, the name of translation table to be used can be supplied as the fifth and last parameter. For example:

OPEN 9050:(:"localhost":1000):::"SOCKTAB"

will open a listening socket device and use the translation table SOCKTAB.

Enabling a Table With the USE Command

When USE ing a device, the name of translation table to be enabled can be supplied as the fourth and last parameter. For example:

```
USE 0:::"PRINTAB"
```

will enable the use the translation table PRINTAB for the principal device.

Disabling Device Translation

To disable device translation, simply issue a USE command with an empty string in the device translation table name field. For example:

```
USE 9050:::""
```

will disable any device translation on device 9050.

Chapter 2. Terminal Devices

Terminal devices in the M21 system are numbered from 1 to 6999. These devices can be either dynamically allocated or set up as part of system generation.

Table 2-1. Valid M commands for use with terminal devices

OPEN	WRITE	READ
CLOSE	ZPRINT	ZLOAD
USE	ZWRITE	ZUSE

It is not possible to use all of the above commands with all types of terminals. For example, issuing a `READ` or `ZLOAD` command to a printer device would not make sense since printers are output-only devices.

OPEN, USE, and CLOSE Parameters

When a terminal device is opened, or input or output is directed to the device with the `USE` command, additional information can be specified to change the characteristics of the terminal. These options will remain in effect until they are either explicitly changed by a subsequent `OPEN` or `USE` command or until the device is `CLOSE` d. The format of the `OPEN` , `USE` , and `CLOSE` commands is:

`OPEN` **{:Condition} Device{:{(Option1:...:Option9)}{:Timeout}{:Domain}{:Translate}}**

`USE` **{:Condition} Device{:{(Option1:...:Option9)}{:Domain}{:Translate}}**

`CLOSE` **{:Condition} Device**

Condition is a post-conditional statement used to control execution of the command. If the post-condition is true then the command is executed, otherwise the command is not executed.

Device is the device to be operated upon.

Option1-Option9 are the terminal device options described in the following paragraphs.

Timeout on the `OPEN` command indicates the maximum amount of time in seconds to wait for the specified device to become available. If **Timeout** is specified, then after the command completes, the `$TEST` Special Variable will contain true (1) if the `OPEN` was successful; otherwise, it will contain false (0). `$TEST` remains unchanged if a timeout value is not specified.

Domain when specified on the `OPEN` command is a comma-separated list of Mnemonic Namespace **domain** names that will be available to this device after the `OPEN` has completed. The first **domain** in the list will be the default domain and will be used for subsequent mnemonic operations. When specified as part of a `USE` command, there should only be one domain specified and this will become the current domain for future mnemonic operations.

Translate is the name of a translation table to use for this device in order to translate characters on input and output to cope with **variations in national character sets and terminal devices** .

Options on the `OPEN` and `USE` commands are evaluated in left-to-right order, with one exception: **Option6** is evaluated before **Option5** . The device characteristics specified by **Option6** are reset first, then the characteristics specified by **Option5** are set.

Right Margin - (Option1)

The right margin option controls when M21 automatically inserts a carriage return/line feed sequence during output operations. When the value of the `$X` Special Variable is equal to the right margin, the system inserts a carriage return/line feed before it outputs the next character to the device. Insertion of the carriage return/line feed is suppressed if the character being output is a carriage return. If the right margin is set to 0 (zero), then automatic insertion of a carriage return/line feed sequence is suppressed. The right margin value can be any number in the range 0 through 255.

Reserved for Future Usage - (Option2)

This option is reserved for future use; and, if present, will be ignored. To ensure compatibility with future releases of M21, do not use this option.

Fixed-length READ - (Option3)

This option is used to set the maximum number of characters to be accepted by the next `READ` command. The `READ` command will be automatically terminated when the specified number of characters has been received, even if there is no carriage return. The system resets this value to 256 after the completion of each `READ` command. The syntax `USE device:(::10) READ var` is equivalent to `USE device READ var#10` .

Reserved for Future Usage - (Option4)

This option is reserved for future use and, if present, will be ignored. To ensure compatibility with future releases of M21, do not use this option.

Device Characteristics to be Set - (Option5)

This option is used to enable terminal device functions. When converted to binary, the value of this option represents a bit string, where each bit specifies a particular characteristic. If the bit is on, then the characteristic is set. If the bit is off, then the characteristic is left unchanged. To enable characteristics, use the `OPEN` or `USE` command to turn on the appropriate bit or bits as described in the following table. Either the value can be specified or an expression that evaluates to the value.

The following table lists the bit numbers and describes the functions associated with each bit. It also includes the decimal values that correspond to the bit value. Remember that the characteristics associated with **Option6** are applied before the **Option5** values.

Table 2-2. Device Characteristics to be Set Values

Bit	Characteristic	Usage
0	NO ECHO	If set, characters received from the terminal are not to be echoed back to the terminal. If not set, printable characters received from the terminal are written back to the terminal. (Decimal value = 1)
1	OUTPUT ONLY	If set, the terminal is an output-only device. If not set, the device supports `READ` operations. (Decimal value = 2)
2	CRT	If set, the terminal in use is a CRT, and characters are to be erased from the screen on backspace. If this bit is not set, the terminal is a hard-copy device and \ is echoed on backspace. (Decimal value = 4)
3	RESERVED	
4	RESERVED	
5	NO BUFFER	If set, output is not buffered and is flushed immediately. If not set, output is buffered. (Decimal value = 32)
6	ESCAPE	If set, the Escape processing feature is enabled. If not set, Escape processing is disabled, and the Escape character is interpreted as a carriage return. Refer to **Escape Processing** for a description of this option. (Decimal value = 64)
7	CURSOR POSITIONING	If set, a VT type cursor positioning escape sequence is sent when the value of `$X` or `$Y` is altered or if **Option7** is specified on an `OPEN` or `USE` command. If this bit is not set, cursor positioning sequences are not sent. (Decimal value = 128)

Bit	Characteristic	Usage
8	RESERVED	
9	RESERVED	
10	RESERVED	
11	RESERVED	
12	TAB CONTROL	If set, the tab character (`$C(9)`) is passed directly through to the port; otherwise, the tab character is translated to the number of spaces needed to move the cursor to the next tab position. Tabs are multiples of eight positions on output. (Decimal value = 4096)
13	LINEFEED	If set, M21 sends only a carriage return for the exclamation point (!) format character in `WRITE` commands. If not set, a carriage return/line feed sequence is sent. This bit also applies to the control sequence sent for margin processing. (Decimal value = 8192)
14	LOWERCASE	If set, all lowercase characters sent by the terminal are converted to uppercase. If not set, lowercase characters are passed through as they are received. (Decimal value = 16384)
15	RESERVED	
16	RESERVED	
17	UPDATE $X and $Y	If set, M21 does not update the values of the `$X` and `$Y` variables on a `WRITE *N` operation. If not set, the values of `$X` and `$Y` are updated when a `WRITE *N` sequence is used. (Decimal value = 131072)

Bit	Characteristic	Usage
18	DATA LENGTH	If set, M21 allows input of full 8-bit characters. If not set, the high-order bit of each character is set to 0 (zero) as it is read, and input is restricted to the 7-bit ASCII character set. (Decimal value = 262144)
19	EMPTY LINE DELETE	If set, when the system BACKSPACE character (Backspace or Delete, or both) is received and the input buffer is empty, the `READ` operation is terminated and the low-order byte of `$ZB` is set to 127. If not set, then the system ignores the BACKSPACE character that is received when the input buffer is empty. (Decimal value = 524288)
20	CONTROL CHARACTERS	If set, control characters not supported by M21 are ignored on a normal `READ` (i.e., are not passed as input and are not echoed). If not set, they are treated as normal input characters. For `READ *` operations, control characters are not echoed, but their value is passed to the variable associated with the `READ *` , regardless of the setting of this characteristic. (Decimal value = 1048576)
21	RESERVED	
22	NO TRANSLATION	If set, any device translations in force for the device are ignored. If not set, device translations are honoured. (Decimal value = 4194304)

Bit	Characteristic	Usage
23	PASS ALL	If set, M21 does not interpret any characters received other than user-specified `READ` terminators (see **Option9**) or ESC if Escape processing is enabled. If not set, the system-defined control characters are interpreted. (Decimal value = 8388608)
24	ZUSE	If set, all `WRITE` commands directed to the device by a `ZUSE` command are ignored. The `ZUSE` command can still be used to check the value of `$ZA` or the other Special Variables. If not set, the system allows `ZUSE` commands to write data to the terminal. (Decimal value = 16777216)
25	TYPEAHEAD	If set, the system flushes the input buffer of all characters received at the start of each `READ` . If not set, the system uses any characters in the input buffer as a result of typeahead. (Decimal value = 33554432)
26	RESERVED	
27	PASS ALL FLOW CONTROL	If set, XON/XOFF is honoured when PASS ALL is enabled. If not set, then XON/XOFF flow control is disabled when PASS ALL is enabled. (Decimal Value = 134217728)
28	ENABLE INTERRUPT	If set, CTRL-C is enabled for the device. If not set, CTRL-C is disabled. (Decimal Value = 268435456)

Bit	Characteristic	Usage
29	BLOCK ON OPEN	If set, an `OPEN` on the device will block for a given time. This time is either the specified timeout value if greater than 2 seconds, the system default timeout if set or for ever. (Decimal Value = 536870912)
30	RESERVED	
31	RESERVED	

Device Characteristics to be Cleared - (Option6)

This option is used to turn off the bits associated with various terminal functions. The definition for the bits is the same as described for the **Device Characteristics to be Set - (Option5)** .

To disable functions, use the `OPEN` or `USE` command to turn off the appropriate bit or bits, as indicated by this option field. Specify the value whose binary representation contains the bits to be turned off. Either the value can be specified or an expression that evaluates to the value.

Value of $X and $Y Special Variables - (Option7)

This option is used to set the value of the `$X` and `$Y` Special Variables. The new values are specified as a single number in the form `$Y` *256+ `$X` (the new value of `$Y` times 256, plus the new value of `$X`). For example, to set `$X` to column 20 and `$Y` to row 6, specify a value of 1556.

Alternatively, it is possible to use a normal `SET` command to change the values of `$X` and `$Y` directly.

Initialisation Parameters - (Option8)

This option may be used to specify the baud rate, parity, data bits, stop bits and flow control characteristics for the terminal device. The parameters are stored as a 32 bit number. The following table lists each bit position's meaning and possible values:

Table 2-3. Initialisation Parameter Options

0-3	BAUD RATE	1	50 baud
		2	75 baud
		3	110 baud
		4	134.5 baud

		5	150 baud
		6	200 baud
		7	300 baud
		8	600 baud
		9	1200 baud
		10	1800 baud
		11	2400 baud
		12	4800 baud
		13	9600 baud
		14	19200 baud
		15	38400 baud
4	1 STOP BIT Set for 1 stop bit. (Decimal value = 16)		
5	2 STOP BITS Set for 2 stop bits. (Decimal value = 32)		
6	7 DATA BITS Set for 7 data bits. (Decimal value = 64)		
7	8 DATA BITS Set for 8 data bits. (Decimal value = 128)		
8	ENABLE PARITY Set to enable parity. (Decimal value = 256)		
9	ODD PARITY Set for odd parity. (Decimal value = 512)		
10	EVEN PARITY Set for even parity. (Decimal value = 1024)		
11	INSTRUMENT DEVICE (Decimal value = 2048)		
12	MODEM DEVICE (Decimal value = 4096)		
13	PRINTER DEVICE (Decimal value = 8192)		
14	SPECIAL DEVICE (Decimal value = 16384)		
15	TERMINAL DEVICE (Decimal value = 32768)		
16	DYNAMIC DEVICE (Decimal value = 65536)		
17	PHYSICAL DEVICE (Decimal value = 131072)		
18	VIRTUAL DEVICE (Decimal value = 262144)		

19	SOFTWARE DEVICE (Decimal value = 524288)
20	RESERVED DEVICE (Decimal value = 1048576)
21	TCP/IP DEVICE (Decimal value = 2097152)
22	SPECIAL I/O OPTIONS (Decimal value = 4194304)
23	NETWORK DEVICE (Decimal value = 8388608)
24	LAT DEVICE (Decimal value = 16777216)
25	ETHERNET DEVICE (Decimal value = 33554432)
26	HOST SPOOL DEVICE (Decimal value = 67108864)
27	INCOMING VIRTUAL DEVICE (Decimal value = 134217728)
28	UNIVERSAL DEVICE (Decimal value = 268435456)
29	TERMINAL TYPE Terminal type is set in the DDB. (Decimal value = 536870912)
30	KEYBOARD TYPE Keyboard type is set in the DDB. (Decimal value = 1073741824)
31	FLUSH ON OPEN If set input is flushed when the device is opened (Decimal value = 2147483648)

READ Terminators - (Option9)

This option specifies which ASCII control characters (values 0 through 31 and 127) are to be recognised as `READ` terminators. When a `READ` command is issued to the device and one of the specified characters is encountered, the `READ` terminates. The value of the termination character is stored in the low byte of `$ZB` . When the device is first opened, **CARRIAGE RETURN, LINE FEED** and **ESCAPE** are set as the default terminators (`$C(13,10,27)`).

A maximum of 32 different `READ` terminator characters may be specified for each terminal device. When the null string is specified as the terminator, no `READ` terminator characters are in effect for the device. The program is then required to use either fixed-length `READ` s (for example: `READ var#30`) or single-character `READ` s (for example: `READ *var`) to process terminal input.

If the **CTRL + S** character (`$C(19)`) is specified as a terminator, it is interpreted as the **DEL** character (`$C(127)`) instead. Therefore the 32 possible `READ` terminator characters consist of the ASCII values 0 through 18, 20 through 31, and 127.

Escape Processing

Escape processing allows the handling of input escape sequences generated by a terminal with a minimum amount of application coding. Many terminals transmit an escape sequence when a program function key or arrow key is pressed. The escape sequence consists of the escape character (`$C(27)`) followed by one or more characters.

When Escape processing is enabled, the system reads additional characters after the escape character is received. These characters are looked up in a table of defined escape sequences and, if a match is found, a value denoting the escape sequence is stored in the high-order byte of the `$ZB` Special Variable (`$ZB\256`). If an escape sequence is not matched, then the high-order byte of the `$ZB` Special Variable is 0 (zero). In either case, the low-order byte of `$ZB` contains the ASCII value of the character that terminated the `READ` .

If an escape sequence is received while a single-character `READ` command is active, the variable specified as the argument to the `READ` command will contain 0 (zero).

M21 supports both standard VT100 function keys and VT220 extended function keys. The function key codes are listed in the following table, together with the high-order byte `$ZB` values:

Table 2-4. Function Key Values

Function Key	Escape Sequence	$ZB\256
PF1	EscOP	32
PF2	EscOQ	33
PF3	EscOR	34
PF4	EscOS	35
F5	EscOT	36
F6	Esc [17~	37
F7	Esc [18~	38
F8	Esc [19~	39
F9	Esc [20~	40
F10	Esc [21~	41
F11	Esc [23~	43
F12	Esc [24~	44
F13	Esc [25~	45
F14	Esc [26~	46

Function Key	Escape Sequence	$ZB\256
HELP	Esc [28~	48
DO	Esc [29~	49
F17	Esc [31~	51
F18	Esc [32~	52
F19	Esc [33~	53
F20	Esc [34~	54
UP ARROW	Esc [A	17
DOWN ARROW	Esc [B	18
RIGHT ARROW	Esc [C	19
LEFT ARROW	Esc [D	20
FIND	Esc [1~	21
INSERT HERE	Esc [2~	22
REMOVE	Esc [3~	23
SELECT	Esc [4~	24
PREVIOUS SCREEN	Esc [5~	25
NEXT SCREEN	Esc [6~	26

$ZA, $ZB, and $ZC Feedback Information

After each terminal input or output command, status information about the terminal and the last operation performed is stored in the `$ZA` , `$ZB` , and `$ZC` Special Variables. For terminal devices, these variables have the following meanings:

`$ZA` contains the 32-bit value of the device characteristics for the terminal. These are the characteristics set and cleared using **Option5** and **Option6** above.

`$ZB` contains information that describes how the input command was terminated. If Escape processing is disabled, then the value of `$ZB` is as follows:

`$ZB` = 13 If the `READ` command was terminated with a carriage return.

`$ZB` = 10 If the `READ` command was terminated with a line feed.

`$ZB` = 27 If the `READ` command was terminated with the escape character.

`$ZB` = nn If the `READ` command was terminated by one of the user-specified termination characters (where `$C(nn)` is the termination character).

`$ZB` = 0 If the `READ` command was terminated because the number of characters specified by the field length option was received.

If Escape processing is enabled, then `$ZB` contains a two-byte value with the low-order byte containing the same information as described above. The high-order byte contains the escape sequence indicator value as shown in the table above under escape processing.

`$ZC` Contains 0 if the operation completed normally, -1 if end-of-file was reached, or 1 if a device error occurred.

The values of `$ZA` , `$ZB` , and `$ZC` are always returned for the current device. A `USE` command must be issued to make the terminal device the current device.

Terminal Device Examples

Table 2-5. How to access terminal type devices using the OPEN , USE , and CLOSE commands

Command	Description
U 0:(0::::16384)	Converts all lower case characters to upper case for the principal device.
O 300:(::::1):15	**OPEN s device 300, turns off echo, and waits for up to fifteen seconds for the device to become available.**
U $I:(::::8388608)	Enables pass-all mode for the current device.
U 0:(::::64:$ZA)	For the principal device, turns off all device currently set device characteristics, then turns on escape processing.
U 1000:25	Sets line width for device 1000 to 25 characters.
U 0:(:::::::$C(13,10,8))	**Sets READ terminators for the principal device to carriage return, line feed, and backspace.**
U $P(::::1)	Turns off echo for the principal device.

Chapter 3. Magnetic Tape Devices

M21 allows access to tape devices configured on the system using device identifiers in the range 7000 to 7049. The following commands are valid for tape devices:

Table 3-1. Valid M commands for use with tape devices

OPEN	WRITE	READ
CLOSE	(Z)PRINT	ZLOAD
USE	(Z)WRITE	

To use a magnetic tape device, an M21 job opens one of the tape devices specifying the necessary parameters. The mapping between the physical tape device and the device number within the M environment is specified in two external configuration files (**xxxxx.ports and system.conf**), where **xxxxx** is the database name. The **system.conf** file needs to be placed in the **etc** sub-directory of the base installation directory and the **xxxxx.ports** file needs to be placed where the database files reside. The **system.conf** file specifies the mapping between the physical device and a logical device number, whilst the **xxxxx.ports** file maps the logical device number to the specific M device number. To include comments in these configuration files prefix them with " ## ".

The following example **system.conf** configuration file illustrates how tape devices are specified. For more information look at the prototype **system.conf** file supplied with the system named **system.conf.proto** and the **system.conf** file documentation:

```
## Map tape /dev/rmt1.4 to logical device 0
## This is an 8mm variable density tape drive
TAPEDRIVE=0,V,8mm,/dev/rmt1.4
```

The following example **xxxxx.ports** configuration file illustrates how tape devices are mapped to M device numbers. For more information see the **.ports** file documentation:

```
## Tape Drive
## Map logical drive 0 to device number 7000
PORT=7000
CLASS=TD
DRIVENUMBER=0
TYPE=TAPE
END
```

OPEN, USE, and CLOSE Parameters

The format of the `OPEN` , `USE` , and `CLOSE` commands for the tape devices is as follows:

`OPEN` **{:Condition} Device{:{(Option1:...:Option4)}{:Timeout}{::Translate}}**

`USE` **{:Condition} Device{:{(Option1:...:Option4)}{::Translate}}**

`CLOSE` **{:Condition} Device**

Condition is a post-conditional statement used to control execution of the command. If the post-condition is true then the command is executed, otherwise the command is not executed.

Device is the device to be operated upon.

Option1-Option4 are the magnetic tape device options described in the following paragraphs.

Timeout on the `OPEN` command indicates the maximum amount of time in seconds to wait for the specified device to become available. If **Timeout** is specified, then after the command completes, the `$TEST` Special Variable will contain true (1) if the `OPEN` was successful; otherwise, it will contain false (0). `$TEST` remains unchanged if a timeout value is not specified.

Translate is the name of a translation table to use for this device in order to translate characters on input and output to cope with **variations in national character sets and terminal devices** .

Access Mode - (Option1)

This option is used to specify the mode in which the tape device is accessed. Each mode is denoted by a single character as described in the following table. The values of this option can be uppercase or lowercase and must be enclosed in quotation marks (for example, `"A"`). All access modes may be specified on an `OPEN` command, but the only access mode valid for a `USE` command is " `R` " for read-only.

Table 3-2. Access Mode Options

Access Mode	Description
A	Use the ASCII character set for all input or output.
B	Share the VIEW buffer with the tape device. All I/O operations are carried out to or from the VIEW buffer and are achieved by using `WRITE *` commands. The only other access modes valid with B are C, T and a tape density.
C	Use double buffering to carry out aynchronous tape operations.
D	Stream format using the delimiters specified in option 4.
E	Use the EBCDIC character set for all input or output. Translation to and from ASCII characters is carried out automatically.
F	Fixed length record format. The record length must be specified in **Option 2** .
L	Process standard tape labels.

Access Mode	Description
R	Read-only mode. If this access mode is not specified then read/write mode is assumed.
S	Stream format consisting of length followed by data with no delimiters. Records cannot span blocks.
U	Do not process standard tape labels.
V	Variable length format
W	Write enabled
Y	Stream format consisting of a length followed by data with no delimiters. Records can span blocks.
Z	Allow variable length blocks with fixed length records
3	800 bits per inch
4	1600 bits per inch
5	6250 bits per inch
6	Auto density

Logical Record Length - (Option2)

This option specifies the record length for fixed length record format. It is only valid for fixed length record format and only valid on the **OPEN** *command. This option can have a value between 0 and 32K bytes.*

Physical Block Size - (Option3)

This option specifies the size of each block that is read from or written to the tape device. This option can have a value between 1 and 32K bytes. For fixed length record format the block size must be a multiple of the logical record length specified in option 2. The physical block size defaults to the physical block size used when the M database was created.

Delimiter String - (Option4)

This option is used to specify a string of between 1 and 3 ASCII characters to be used as a delimiter when accessing the tape using mode " `D` ".

WRITE Commands

A set of write commands can be used to control the tape drive as specified in the following table:

Table 3-3. WRITE Commands

Write Command	Description
*1	Backspace the tape one physical block
*2	Forward space the tape one physical block.
*3	Write a tape mark.
*4	Write a block. Only valid in shared VIEW buffer mode
*5	Rewind the tape.
*6	Read a block. Only valid in shared VIEW buffer mode
*7	Read the tape label or the next block if it not a label.
*8	Write a tape header label. Ignored if not using label processing mode.
*9	Write the end of file label. Ignored if not using label processing mode.
*10	Update the tape status.
*11	Backward space to last tape mark.
*12	Forward space to next tape mark.
*13	Erase tape.
*14	Retension tape.
*15	Forward space to the end of data.
*16	Rewind and unload tape.

$ZA, $ZB, and $ZC Feedback Information

After each tape input or output command, status information about the results of the command is stored in the `$ZA` , `$ZB` , and `$ZC` special variables. For the tape devices, these variables have the following meanings:

`$ZA` Contains zero if the tape is positioned at the beginning of the tape or 256 if a logical tape error has occurred.

`$ZB` Contains the total number of characters either written to or read from the tape device.

`$ZC` Contains a value of 0 if the operation completed normally, a value of -1 if the end of file (tape mark) was reached or another value to denote tape status information as follows:

101 - Physical end of tape

102 - End of recorded data

103 - Tape mark detected

104 - Logical tape error

105 - Block length error

106 - Tape medium error

107 - Tape write-protected

108 - Data overrun error

109 - Hardware error

110 - Tape not open

111 - Tape read only

112 - Tape drive not online

113 - End of file (one tape mark)

The values of `$ZA` , `$ZB` , and `$ZC` are always returned for the current device. A `USE` command must be issued to make the terminal device the current device.

Tape Device Examples

The following examples illustrate how to access tape devices using the **OPEN** *, and* **USE** *commands.*

Table 3-4. Tape Device Examples

Command	Description
O 7000:("AVL4"):60	Opens the tape on drive 7000 in ASCII, variable length, labelled format, and waits up to 60 seconds for it to become available.
U 7000	Makes a tape drive device the current device.
U 7000 w *5	Rewinds the tape on device 7000.

Chapter 4. VIEW Device

M21 provides one VIEW device, which is used to read or modify information stored on disk. The VIEW device is assigned an identifier of 7100. The following commands and functions are valid for the VIEW device:

Table 4-1. Valid M commands for use with the VIEW devices

OPEN	VIEW	CLOSE
$VIEW	USE	

To use the forms of the `VIEW` command or `$VIEW` function that access the VIEW buffer, the VIEW device needs to be `OPEN` ed. It need not be the current device.

OPEN, USE, and CLOSE Parameters

Before issuing the `VIEW` command or using the `$VIEW` function, obtain ownership of the VIEW device by issuing an `OPEN` command to the device. The format of the `OPEN` , `USE` , and `CLOSE` commands is as follows:

`OPEN` **{:Condition} 7100{:(Option1:Option2:Option3:Option4)}{:Timeout}**

`USE` **{:Condition} 7100{:(Option1:Option2:Option3:Option4)}**

`CLOSE` **{:Condition} 7100**

Condition is a post-conditional statement used to control execution of the command. If the post-condition is true then the command is executed, otherwise the command is not executed.

Timeout on the `OPEN` command indicates the maximum amount of time in seconds to wait for the specified device to become available. If **Timeout** is specified, then after the command completes, the `$TEST` Special Variable will contain true (1) if the `OPEN` was successful; otherwise, it will contain false (0). `$TEST` remains unchanged if a timeout value is not specified.

To modify disk data, the VIEW device must be owned. When the VIEW device is opened, a buffer is assigned to the device. This area in memory is called the VIEW buffer. The size of the VIEW buffer is 1K bytes. Disk data is read into or written from this area. When device 7100 is `CLOSE` d, modified data in the VIEW buffer is `not` automatically written to disk.

Reserved for Future Usage - (Option1)

This option is reserved for future use and, if present, will be ignored.

Reserved for Future Usage - (Option2)

This option is reserved for future use and, if present, will be ignored.

Reserved for Future Usage - (Option3)

This option is reserved for future use and, if present, will be ignored.

Mode Switches - (Option4)

Use this option to specify the mode in which the VIEW device is used. The following modes are valid:

`C` Clear mode - resets all mode switches

`P` Protected mode - establishes exclusive access when a block is read

`T` Test mode - inhibits <DKHER> errors

`Z` Block zero access - allows block 0 to be written

The value of this option can be uppercase or lowercase and must be enclosed in quotation marks (for example, **"P"** *).n*

$ZA, $ZB, and $ZC Feedback Information

After each input or output operation using the VIEW device to disk, the `$ZA` Special Variable contains the block number of the disk block contained in the VIEW buffer.

The `$ZB` and `$ZC` Special Variables do not provide any information about the VIEW device.

VIEW Device Examples

The following examples illustrate how to access the VIEW device using the `OPEN` , `USE` , and `CLOSE` commands.

Table 4-2. VIEW Device Examples

Command	Description
O 7100:(:::"T")	Opens the VIEW device in test mode, inhibiting <DKHER> errors.
O 7100::30	**OPEN s the VIEW device and waits up to 30 seconds for it to become available.**

Chapter 5. Host File Devices

Host File devices allow the manipulation of files stored by the host operating system. Files created by a Host File device are written using the host operating system file structure and can be accessed externally just like any other file. The Host File device transfers information to and from the file on a logical record basis. Whenever a `WRITE` command is issued information is put into the host file, however buffering takes place and the data may not appear in the file immediately, or in some circumstances, until the Host File device is closed. Whenever a `READ` command is issued to a Host File device, information up to the delimiter or length of the fixed-length `READ` is read from the specified file. Host File devices are private to the M21 process accessing them so that the same device number can be opened by different jobs.

The Host File devices are assigned device designators in the range 8000 to 8049.

Table 5-1. Valid M commands for use with terminal devices

OPEN	WRITE	READ
CLOSE	(Z)PRINT	ZLOAD
USE	(Z)WRITE	

To access a host file, `OPEN` a Host File device specifying a file name and mode (read, write, mixed, or append). It is also possible to specify a byte offset into the file. Except for buffer mode, all subsequent accesses to the file will be in sequential order.

Each input or output operation begins at the byte immediately following the last byte of the previous input or output operation. It is possible to alter the byte location within the file for the next read or write operation by specifying an optional parameter on the `USE` command before accessing the file.

When accessing Host File devices, input and output operations can both be used if the file was opened for both `READ` and `WRITE` (or `MIXED` mode). Data can also be accessed randomly, by specifying a new byte location on the `USE` command before each input or output operation.

Lines written to a Host File device are terminated with an operating system-specific delimiter. When a `READ` is issued to a Host File device, M21 transfers information starting at the current location until a delimiter character is encountered or the maximum number of characters specified on the `READ` command is reached.

OPEN, USE, and CLOSE Parameters

When a Host File device is opened, or input or output is directed to the device with the **USE** command, additional information can be specified with the command to alter the characteristics of the device. These options remain in effect until they are either changed by a subsequent **OPEN** or **USE** command or until the device is **CLOSE** d and are used to specify which file is to be accessed or to position the Host File device at a given location within the file. The format of the **OPEN** , **USE** , and **CLOSE** commands is:

OPEN *{:Condition} Device{:{(Option1:...:Option8)}{:Timeout}{::Translate}}*

USE *{:Condition} Device{:{(Option1:...:Option8)}{::Translate}}*

CLOSE *{:Condition} Device*

Condition is a post-conditional statement used to control execution of the command. If the post-condition is true then the command is executed, otherwise the command is not executed.

Device is the device to be operated upon.

Option1-Option8 are the Host File device options described in the following paragraphs.

Timeout on the **OPEN** command indicates the maximum amount of time in seconds to wait for the specified device to become available. If **Timeout** is specified, then after the command completes, the **$TEST** Special Variable will contain true (1) if the **OPEN** was successful; otherwise, it will contain false (0). **$TEST** remains unchanged if a timeout value is not specified.

Translate is the name of a translation table to use for this device in order to translate characters on input and output to cope with variations in national character sets and terminal devices.

File Name - (Option1)

This option specifies the name of the host operating system file to be accessed optionally including a path and file extension. If an `OPEN` command specifies a Host File device that is already opened for this job, then the currently open host file will be `CLOSE` d before the new file is `OPEN` ed.

This option is not valid for the `USE` command and is ignored.

I/O Mode - (Option2)

This option is used to specify the mode in which the host file is to be accessed. The following modes are valid:

A Append to file

B Block mode using VIEW buffer (device 7100)

C Use block READ and WRITE commands

M Mixed (Read and Write)

R Read only (assumed if this option is omitted)

W Write only

The value of this option can be uppercase or lowercase and must be enclosed in quotation marks (for example, `"M"`).

When using block mode, the Host File device shares the VIEW buffer with the VIEW device (7100) and is set to use block `READ` and `WRITE` commands. A <`NOPEN`> error occurs if block mode is specified but the VIEW device is not already opened.

This option is not valid for the `USE` command and is ignored.

Offset - (Option3)

This parameter is used to specify an offset into the host file. The value is relative to the location specified in **Option4** . If this option is omitted then, for `OPEN` , an offset of 0 is assumed and for `USE` the offset remains unchanged.

Block Number or Location - (Option4)

This parameter is used to specify either a block number or a location dependent on the mode in which the file is opened.

When a file is opened in block mode using the shared VIEW device, then it is treated as a set of contiguous 1K blocks starting at 0. This option is then used to specify the block number to be used for subsequent input or output operations. If this omitted, the file will be positioned at block 0.

For modes other than block mode, this option can be used to specify the starting point for any offsets specified in **Option3** as follows:

0 Beginning of file

1 Current position in the file

2 End of the file

If this option is not specified, the default is 0.

Format Flags - (Option5)

This option can be used to determine the mode of recording the data in the file. The values of this option can be uppercase or lowercase and must be enclosed in quotation marks (for example,`"A"`).

Table 5-2. Valid formats for Format Flags option

Format Flag	Description
D	Data with user settable delimiter string using **Option 6** (the default format).
F	Fixed length record format. The record size must be specified in **Option 8**.
J	Special format for reading the M21 after image journal circular file.
S	Stream format consisting of length followed by data with no delimiters. Records cannot span blocks.
V	Variable length format. Four bytes with length in ASCII followed by the data, no other delimiters are allowed.
X	Extra stream format with 2 byte length then data, no other delimiters are allowed.

Y	Stream format consisting of a length followed by data with no delimiters. Records can span blocks.
Z	Ignore null bytes when reading.
-	Disable buffering of multiple records.
+	Enable buffering of multiple records (the default).

Delimiters - (Option6)

This option can be used to specify a sequence of 1 to 3 ASCII characters, which will be used to separate records in the default format (" **d** "). Whenever a **WRITE** ! is carried out with a Host File device as the current device, then this sequence is written to the file. Whenever performing a **READ** operation from a Host File device, if this sequence is encountered, then an end-of-record condition occurs. If no delimiter is required, then specify this parameter as the null string (""). If omitted, the normal operating system-specific delimiter will be used.

Buffer Size - (Option7)

This option can be used to specify the buffer size for Host File device operations. The size can be between 512 bytes and 32K bytes and, if not specified, will default to 2048 bytes.

Record Size - (Option8)

This option is only relevant for accessing the Host File device in fixed length record mode (" `f` "). It can be used to specify the record size for input or output operations. The value of this option may be between 1 and the buffer size specified in **Option 7** . The buffer size in **Option 7** must be an exact multiple of the record size specified by this option.

$ZA, $ZB, and $ZC Feedback Information

After each Host File input or output command, status information about the device and the last operation performed is stored in the **$ZA** , **$ZB** , and **$ZC** Special Variables. For Host File devices, these variables have the following meanings:

$ZA contains the number of bytes transferred to or from the file. If an error occurs on transferring the data **$ZA** will contain the value -1. After opening a Host File device **$ZA** will either contain 0, if the file was opened successfully or -1 if the open failed.

$ZB contains the current block number if the file was opened with mode " `B` " or " `C` ". In any other mode, `$ZB` contains the current byte offset into the file .

$ZC Contains 0 if the operation completed normally, -1 if end-of-file was reached, or 1 if a device error occurred.

The values of **$ZA** , **$ZB** , and **$ZC** are always returned for the current device. A **USE** command must be issued to make the terminal device the current device.

Host File Device Examples

The following examples illustrate how to access terminal type devices using the **OPEN** , **USE** , and **CLOSE** commands:

Table 5-3. Host File Device Examples

Command	Description
O 8000:("/etc/password")	Opens the file **/etc/password** for reading.
O 8000:("test":"M")	Opens the file **test** for both read and write.
U 8000	Sets the principal device to the host file device.
U 8000:("LOG":"W"	Opens the file **LOG** for write only.

Chapter 6. The NULL Device

M21 provides one NULL device, which is used to discard output. Any READ command, with the NULL device as the current device, returns immediately with a result of null (""). The NULL device is assigned an identifier of 8050. The following commands and functions are valid for the NULL device:

Table 6-1. Valid M commands for use with the NULL device

OPEN	WRITE	READ
CLOSE	(Z)PRINT	ZLOAD
USE	(Z)WRITE	

To use the forms of the `VIEW` command or `$VIEW` function that access the VIEW buffer, the VIEW device needs to be `OPEN` ed. It need not be the current device.

OPEN, USE, and CLOSE Parameters

Before issuing the `VIEW` command or using the `$VIEW` function, obtain ownership of the VIEW device by issuing an `OPEN` command to the device. The format of the `OPEN` , `USE` , and `CLOSE` commands is as follows:

`OPEN` **{:Condition} 8050{:{(Option1:Option2:Option...)}{:Timeout}{:Domain}{:Translate}}**

`USE` **{:Condition} 8050{:{(Option1:Option2:Option...)}{:Domain}{:Translate}}**

`CLOSE` **{:Condition} 8050**

Condition is a post-conditional statement used to control execution of the command. If the post-condition is true then the command is executed, otherwise the command is not executed.

Option1-Option... are options that can be specified but are not evaluated.

Timeout on the `OPEN` command indicates the maximum amount of time in seconds to wait for the specified device to become available. If **Timeout** is specified, then, for the NULL device, it is ignored.

Domain when specified on the `OPEN` command is a comma-separated list of Mnemonic Namespace **domain** names that will be available to this device after the `OPEN` has completed. The first **domain** in the list will be the default domain and will be used for subsequent mnemonic operations. When specified as part of a `USE` command, there should only be one domain specified and this will become the current domain for future mnemonic operations.

Translate is the name of a translation table to use for this device in order to translate characters on input and output to cope with **variations in national character sets and terminal devices** .

$ZA, $ZB, and $ZC Feedback Information

For the NULL device the `$ZA` , `$ZB` and `$ZC` Special Variables always return the value 0.

Chapter 7. Interjob Communication / Pipe Devices

The Interjob Communication (IJC) facility of M21 allows two or more jobs, which are executing to pass messages back and forth using named pipe communication. Devices are private to the M21 process accessing them so that the same device number can be opened by different jobs. An interjob communication device can either be opened for reading or writing

The Interjob Communication devices are assigned device identifiers in the range 8070 to 8079. The following commands are valid for the IJC devices:

Table 7-1. Valid M commands for use with Interjob Communication / Pipe Devices

OPEN	WRITE	READ
CLOSE	(Z)PRINT	ZLOAD
USE	(Z)WRITE	

To use the Interjob Communication facility, an M21 job opens and uses one of the devices in write mode specifying a logical name for the communication channel and then writes one or more messages. To receive messages, an M21 job opens and uses a free device in read mode, specifying the same logical communication channel name and then reads each message that has been sent.

Logical communication channel names are similar to global names in that they are unique to the UCI in which the M job is executing unless the first character of the name is the " `%` " character, when they are system wide. The process of opening an Interjob Communication device associates the given device number with the logically named communication channel as either a reader or a writer.

It is possible for multiple jobs to send messages to a single job by opening an Interjob communication device in read mode with the same logical communication channel name (same name in the same UCI or starting with " `%` "), sending a message, and then closing the device. Similarly, it is possible for multiple jobs to receive messages from a single logical communication channel by opening for read with the same logical name. However, the transmitted message will not be received by all jobs listening on the logical channel, but rather, the first job to be scheduled and run by the operating system. Because there is no way to predict the order in which multiple jobs will execute, be cautious when using Interjob Communication. If a job reads from a device and does not receive a message, it will wait indefinitely unless a timeout value is specified on the `READ` .

All transmissions are fully buffered using support of the underlying operating system. This means that the receiving job does not need to have the device open at the time the sending job writes to the device. The sending job can continue putting information into the device until the buffer becomes full with execution being suspended for the sending job until information is removed from the buffer by the receiving job. Similarly, the receiving job can retrieve information from the buffer after the sending job has `CLOSE` d the sending device. The size of the Interjob Communication internal buffer is 5K bytes.

Unlike other devices, whenever a `WRITE` command with multiple operands is issued to an Interjob Communication device, each operand is a separate record in the buffer. The receiving job must issue a `READ` command for each record in the buffer. Fixed-length `READ` s are not supported for these devices.

Errors can be generated if the Interjob Communication devices are used incorrectly. If a **READ** *command is issued to a device that is opened in write mode, M21 generates a* <**MODER**> *error. Similarly, if a* **WRITE** *command is issued to a device open in read mode, M21 generates a* <**MODER**> *error.*

OPEN, USE, and CLOSE Parameters

The format of the `OPEN` , `USE` , and `CLOSE` commands for the Interjob communication devices is as follows:

`OPEN` **{: Condition } Device {:{(Option1 : Option2)}{: Timeout }{::Translate}}**

`USE` **{: Condition } Device {:::Translate}**

`CLOSE` **{: Condition } Device**

Condition is a post-conditional statement used to control execution of the command. If the post-condition is true then the command is executed otherwise the command is not executed.

Timeout on the `OPEN` command indicates the maximum amount of time in seconds to wait for the specified device to become available. If **Timeout** is specified, then after the command completes, the `$TEST` Special Variable will contain true (1) if the `OPEN` was successful; otherwise, it will contain false (0). `$TEST` remains unchanged if a timeout value is not specified.

Translate is the name of a translation table to use for this device in order to translate characters on input and output to cope with **variations in national character sets and terminal devices** .

Name - (Option1)

This is the logical name to give to the communications channel. The name is unique to the UCI in which the M job is executing unless the first character of the name is the " `%` " character, in which case it is system wide. The name can be up to 32 characters in length.

Mode - (Option2)

Use this option to specify the mode in which the Interjob Communication device is to be used. The following modes are valid:

`R` Open Interjob Communication device for read.

`W` Open Interjob Communication device for write.

The value of this option can be uppercase or lowercase and must be enclosed in quotation marks (for example, **"R"** *). If the mode is not specified this option defaults to* **R** *(read).*

$ZA, $ZB, and $ZC Feedback Information

After each Interjob Communication input or output command, status information about the results of the command is stored in the `$ZA` , `$ZB` , and `$ZC` special variables. For the Interjob Communication devices, these variables have the following meanings:

`$ZA` Contains the number of characters either written or read by the last input or output command to the Interjob Communication device, depending on whether the device was opened for read or write. The `$ZA` Special Variable is set to 0 at the beginning of each `READ` or `WRITE` operation.

`$ZB` Contains the total number of characters either written to the Interjob Communication device, depending on whether the device was opened for read or write. The `$ZB` Special Variable is only set to 0 when the device is initially opened.

`$ZC` Contains a value of 0 if the operation completed normally, a value of -1 if the end of file was reached or a value of 1 if a device error occurred.

The values of `$ZA` , `$ZB` , and `$ZC` are always returned for the current device. A `USE` command must be issued to make the IJC device the current device.

Interjob Communication Device Examples

The following examples illustrate how to access the Interjob Communication devices using the **OPEN** , and **USE** commands.

Table 7-2. Interjob Communication Device Examples

Command	Description
O 8070:("test":"w"):60	**Opens logical communication channel " test " for write, and waits up to 60 seconds for it to become available.**
U 8071	Makes an Interjob Communication device the current device.
O 8072:("%a":"r")	**Opens logical communication channel " %a "(system wide) for read.**

Chapter 8. Host System Spooling Devices

The host system spooling devices are assigned device identifiers in the range 8100 to 8499. The following commands are valid for these devices:

Table 8-1. Valid M commands for use with Host System Spooling Devices

OPEN	WRITE
CLOSE	(Z)PRINT
USE	(Z)WRITE

OPEN, USE, and CLOSE Parameters

The format of the `OPEN, USE` and `CLOSE` commands is as follows:

OPEN **{:Condition} Device{:{:Timeout}{:Domain}{:Translate}}**

`USE` **{:Condition} Device{:{:Domain}{:Translate}}**

`CLOSE` **{:Condition} Device**

Condition is a post-conditional statement used to control execution of the command. If the post-condition is true then the command is executed, otherwise the command is not executed.

Device is the device to be operated upon.

Timeout *on the* **OPEN** *command indicates the maximum amount of time in seconds to wait for the specified device to become available. If* **Timeout** *is specified, then after the command completes, the* **$TEST** *Special Variable will contain true (1) if the* **OPEN** *was successful; otherwise, it will contain false (0).* **$TEST** *remains unchanged if a timeout value is not specified.*

Domain when specified on the `OPEN` command is a comma-separated list of Mnemonic Namespace **domain** names that will be available to this device after the `OPEN` has completed. The first **domain** in the list will be the default domain and will be used for subsequent mnemonic operations. When specified as part of a `USE` command, there should only be one domain specified and this will become the current domain for future mnemonic operations.

Translate is the name of a translation table to use for this device in order to translate characters on input and output to cope with **variations in national character sets and terminal devices** .

To use a host system spooling device, an M21 job simply opens one of the host system spooling devices. The mapping between the print spooler and the device number within the M environment is specified in an external configuration file (**xxxxx.ports**), where **xxxxx** is the database name. The **xxxxx.ports** file needs to be placed where the database files reside. The **xxxxx.ports** file maps the M device number to the printer via the UNIX print spooler. The appropriate code to write to the host system spool device must be present as `COMMAND={code}` . To include comments in this configuration file prefix them with " `##` ".

The following example **xxxxx.ports** configuration file illustrates how a host spool devices are mapped to M device numbers. For more information see the **.ports** file documentation:

```
## Printer Device
PORT=8101
CLASS=HS
COMMAND=lp -dprinter1
TYPE=PRINTER
END!
```

Chapter 9. TCP/IP and UDP Socket Devices

Sockets are accessed and manipulated using a socket device. The socket device can contain a collection of sockets. At any one time, a single socket from the collection is the current socket. Furthermore, socket can be attached (added) and detached (deleted) from a device's collection of sockets. **Socket** devices are private to the M21 process accessing them so that the same device number can be opened by different jobs. There is no limitation on the number of sockets a partition can open, other than that imposed by the host operating system.

The Socket devices are assigned device designators in the range 9050 to 9099.

The following M commands are valid for use with Socket devices:

Table 9-1. Valid M commands for use with Host System Spooling Devices

OPEN	WRITE	READ
CLOSE	(Z)PRINT	ZLOAD
USE	(Z)WRITE	

The `READ` command is used to obtain data from a socket. A `READ` command will terminate if any of the following are detected, in the order specified:

- An error condition. `$DEVICE` reflects the error. `$KEY` is assigned the empty string. No data is returned.
- Read timeout. `$KEY` is assigned the empty string. The `READ` command returns data received up to the timeout.
- Read delimiter. `$KEY` is assigned the delimiter character that terminated the read. The `READ` command returns data received up to, but not including, the delimiter.
- Fixed-length `READ` requirements are satisfied. This occurs only after the specified number of characters are received. `$KEY` is assigned the empty string. The `READ` command returns the characters received.
- For a stream oriented socket, when the buffer is empty the read will block. When at least one character is available, the `READ` command returns the available characters up to the maximum string length (32 Kbytes). Note that the number of characters returned is not predictable. `$KEY` is assigned the empty string.
- For a datagram socket, when a complete message is received `READ` returns the message. `$KEY` is assigned the empty string.

The `WRITE` command is used to send data to a socket. No `WRITE` commands affect internally buffered input data. The `WRITE` handler will loop until all data has been written to the current socket or an error occurs. If the host operating system's internal socket buffer is full, the `WRITE` command may block until the buffer has sufficient space to resume execution.

`WRITE !` causes the <`CR`><`LF`> sequence to be written to the device's current socket. `$X` is set to 0. and `$Y` is incremented by 1.

`WRITE #` causes the new page sequence, <`CR`><`FF`> , to be written to the current socket. `$X` and `$Y` are set to 0.

In the event of a `WRITE` error, `$DEVICE` will contain an error code detailing the error, otherwise it will contain zero (0).

OPEN, USE, and CLOSE Parameters

When a Host File device is opened, or input or output is directed to the device with the **USE** command, additional information can be specified with the command to alter the characteristics of the device. These options remain in effect until they are either changed by a subsequent **OPEN** or **USE** command or until the device is **CLOSE** d and are used to specify which file is to be accessed or to position the Host File device at a given location within the file. The format of the **OPEN** , **USE** , and **CLOSE** commands is:

OPEN ***{:Condition} Device{:{(Option1:...:Option7)}{:Timeout}{::Translate}}***

USE ***{:Condition} Device{:{(Option1:...:Option7)}{::Translate}}***

CLOSE ***{:Condition} Device:Socket***

Condition is a post-conditional statement used to control execution of the command. If the post-condition is true then the command is executed, otherwise the command is not executed.

Device is the device to be operated upon.

Option1-Option7 are the socket device options described in the following paragraphs.

Timeout on the **OPEN** command indicates the maximum amount of time in seconds to wait for the specified device to become available. If **Timeout** is specified, then after the command completes, the **$TEST** Special Variable will contain true (1) if the **OPEN** was successful; otherwise, it will contain false (0). **$TEST** remains unchanged if a timeout value is not specified.

Translate is the name of a translation table to use for this device in order to translate characters on input and output to cope with variations in national character sets and terminal devices.

Socket If the number of a specific socket (obtained using `$ZSOCKET`) is supplied to the `CLOSE` command, then this socket alone is closed, leaving all other sockets attached to the device unaffected. If no close **Socket** parameter is given, then all sockets attached to the device are closed.

Type - (Option1)

A type of "UDP" specifies a datagram socket and "TCP" specifies a stream socket. If omitted, a default of "TCP" is assumed.

Address - (Option2)

The address option specifies the IP address of the end point to connect to, and is thus only applicable for client sockets. Therefore, if this field is blank a server socket will be created.

The IP address can be specified either in hostname or dotted decimal, i.e. a.b.c.d, format.

Port - (Option3)

The port option specifies the IP port number and is compulsory for the creation of all socket types.

Delimiters - (Option4)

This option can be used to specify a string of single I/O delimiter characters for the specific socket device. Socket devices do not support multi-character delimiter strings.

Multiple appearances of a character in the delimiter string will only be counted once. There is no limit on the amount of allowed delimiter characters (other than the fact that an 8-bit variable only has 256 discrete values).

If no delimiters are specified for the socket then none are used.

I/O Error Mode - (Option5)

This option specifies the I/O error-trapping mode. A value of " `NOTRAP` " specifies that I/O errors on a device do not raise error conditions, and a value of " `TRAP` " means that I/O errors will raise error conditions.

The default error-trapping mode for a socket is " `NOTRAP` ".

Attach Socket - (Option5)

This option can be used to specify the handle of a socket (obtained using `$ZSOCKET`) to attach to the device. If the socket is already attached to another device, the operation will fail. The newly attached socket will become the current socket for the device even if the socket is already held by the device (this is useful to force a particular socket to become the active one).

This parameter must be used on its own.

Detach Socket - (Option5)

Specifies the handle of a socket (obtained using `$ZSOCKET`) to detach from the device, without affecting the socket's existing connection. The socket can then be attached to another device.

This parameter must be used on its own.

Socket Control Mnemonics

/listen[(expr)]

The use of this mnemonic establishes the queue depth for incoming connections on a server socket. Note that this only applies to stream sockets and has no meaning for datagram sockets.

If **expr** is not supplied, a default queue depth of 3 will be set.

/wait[(expr)]

This mnemonic is used to wait for an event on a server socket, subject to a timeout. If the **expr** timeout value is omitted or negative, it is assumed to be 0. When the operation completes, `$KEY` contains a value identifying the event that occurred. In the event of an error or a timeout, `$KEY` returns the empty string.

For a stream server socket, the process will wait for an incoming client connection. If the timeout specified is 0, the wait will be indefinite; otherwise it will end when the timeout expires. If a connection request is received, `$KEY` will contain the value " `CONNECT` ", a new socket will be allocated to handle the connection with the client, and this new socket becomes the current socket for the device.

For datagram sockets, the process will wait for a received message on all datagram server sockets associated with the device. If a message arrives, `$KEY` will contain the value " `READ` ", and the socket which received the message will become the current socket for the device.

The $ZSOCKET Function

This function is used to return implementation specific information on the sockets in use. The following syntax is supported.

$ZSOCKET(device) = intexpr

Each device has a collection of sockets associated with it, each of which is assigned a unique handle number. This case is used to return the handle for the current socket on a device. This handle can then be supplied to the `OPEN` or `USE` command to attach or detach a socket to or from a device.

Note that socket handles are unique across all partitions in a database, otherwise they would not be able to be used in attach and detach operations between different processes.

$ZSOCKET(device, handle, "DELIMITER") = expr

This returns a string containing all the active delimiter characters for the device in ascending ASCII code order.

$ZSOCKET(device, handle, "IOERROR") = expr

This returns the current I/O error-trapping mode for the device, i.e. " `TRAP` " or " `NOTRAP` ".

$ZSOCKET(device, handle, "LOCALADDRESS") = expr

This returns the local IP address for the specified socket in dotted decimal notation.

$ZSOCKET(device, handle, "PROTOCOL") = expr

This returns the protocol type for the socket, i.e. " `TCP` " for stream sockets or " `DATAGRAM` " for datagram sockets.

$ZSOCKET(device, handle, "REMOTEADDRESS") = expr

This returns the IP address for remote end of the network connection on the specified socket in dotted decimal notation.

$ZSOCKET(device, handle, "SOCKETHANDLE") = expr

This returns the UNIX file descriptor for the specified socket.

$ZA, $ZB, and $ZC Feedback Information

After each socket input or output command, status information about the device and the last operation performed is stored in the **$ZA** , **$ZB** , and **$ZC** Special Variables. For socket devices, these variables have the following meanings:

$ZA contains the number of bytes transferred to or from the socket device. After opening a socket device **$ZA** will contain 0.

$ZB is undefined for socket device operations .

$ZC Contains 0 if the operation completed normally, -1 if end-of-file was reached, or 1 if a device error occurred.

The values of **$ZA** , **$ZB** , and **$ZC** are always returned for the current device. A **USE** command must be issued to make the terminal device the current device.

Chapter 10. Example M Code

The following section contains the code examples for client and server socket creation and usage.

Client Process

```
; Connect to a local stream socket server
s timeout=10
s dev=9050
s port=6969
o dev:(:"localhost":port):timeout
i $dev w !,"Unable to connect to server, $dev=",$dev c dev q
s handle=$zsocket(dev)
s address=$zsocket(dev,handle,"REMOTEADDRESS")
u $p w !,"Connected to server at: ", address
u dev:(:::$c(13,10))
; ==> READ/WRITE dialogue with server
c dev
q
```

Serial Server Process

```
; Create a stream server socket
s timeout=10
s dev=9050
s port=6969
o dev:(::port):timeout
i $dev w !,"Unable to create server socket - $device=",$dev c dev q
; Establish a listen queue depth of 3
u dev w /listen(3)
; Wait for a client connection
f  u $p w !,"waiting for client..." u dev w /wait(timeout) q:$dev  i $key'="" d
. u dev:(:::$c(13,10))
. ; ==> READ/WRITE dialogue with client
. s handle=$ZSOCKET(dev)
. ; Close the current socket handle
. c dev:handle
c dev
q
```

Concurrent Server Process

```
; Create a stream server socket
s timeout=10
s dev=9050
s port=6969
o dev:(::port):timeout
I $dev w !,"Unable to create server socket - $device=",$dev c dev q
; Establish a listen queue depth of 3
u dev w /listen(3)
; Wait for a client connection
```

```
f  u dev w /wait(timeout) q:$dev  I $key’="" d
. ; ==> READ/WRITE dialogue with client
. s handle=$zsocket(dev)
. ; Detach the current socket
. u dev:(::::::handle)
. ; Start a new job with the detached socket handle
. j server^srv(handle)
. c dev:handle
c dev q
```

Routine:^srv

```
Server(handle) ;

; Open a socket device with an attach socket
s dev=9050
o dev:(:::::handle)
u dev
; ==> READ/WRITE dialogue with client
c dev
q
```

Chapter 11. Routine Interlock Devices

M21 provides a series of logical devices which may be used for programmer purposes such as synchronisation between different processes whilst they are executing. These devices are assigned device numbers 9200 to 9299. The only valid commands that can be issued to these devices are `OPEN` and `CLOSE` .

The format of the `OPEN` and `CLOSE` commands is as follows:

`OPEN{:` **Condition** `}` **Device** `{::` **Timeout** `}`

`CLOSE{:` **Condition** `}` **Device**

Condition is a post-conditional statement used to control execution of the command. If the post-condition is true then the command is executed, otherwise the command is not executed.

Device is the device to be operated upon.

Timeout on the `OPEN` command indicates the maximum amount of time in seconds to wait for the specified device to become available. If **Timeout** is specified, then after the command completes, the `$TEST` Special Variable will contain true (1) if the `OPEN` was successful; otherwise, it will contain false (0). `$TEST` remains unchanged if a timeout value is not specified.

Application program usage of Routine Interlock devices determine their function and use of these devices has no defined meaning in M21.

The following examples illustrate how to access the Routine Interlock devices using the `OPEN` and `CLOSE` commands:

Table 11-1. Routine Interlock device examples

Command	Description
O 9200::60	Opens a Routine Interlock device and waits up to 60 seconds for it to become available.
C 9200	Closes a Routine Interlock device.