MMSC Command Language

Rob Bradshaw
For MMSC Version 1.0.8

The Multi-Module System Controller (MMSC) is used to control a single Origin2000 rack system. When multiple Origin2000 racks are attached to form a single system, each rack has its own MMSC; these MMSC's are then attached to each other over a private ethernet. Each MMSC can accept commands from a number of different sources, including consoles connected via direct serial connection or modem, individual modules (or "bays") in a rack via their module system controllers (MSC), and other MMSC's via the private ethernet. In addition, one rack in a Origin2000 rack system will have a display with several buttons that can also be used to generate commands.

The commands that are issued to an MMSC may directly affect the operation of the MMSC itself or may be passed along to individual bays, either in the same rack as the MMSC or perhaps in a different rack. This document describes the syntax and contents of these MMSC commands. The MMSC may also be used to pass *MSC commands* through to one or more bays. For more information on the MSC and MSC commands, see the IP27 PROM Technical Reference Manual .

Some notes on terminology

When referring to a Origin2000 system, the term **module** is sometimes understood to mean any particular module in a system, and sometimes just means one of the two modules in a specific rack in a system. To avoid ambiguity, this document will use the word **module** when the "logical" definition (i.e. any module in a system) is intended, and will use the term **bay** when the "physical" definition (i.e. a module in a specific rack) is intended.

Originally, the MMSC was referred to as the "FFSC" (Full Feature System Controller), and the MSC was referred to as the "ELSC" (Entry Level System Controller). The old and new terms may still be used interchangably in some places.

Specifying Destinations

Many commands are intended for delivery to specific racks (MMSCs) and/or bays (MSCs), so it is important to have a way to specify particular destinations.

At the lowest level, racks are addressed with small integers starting with 1, and MSCs within a rack (bays) are addressed with letters representing their position (**U** and **L** for Origin2000).

Modules are also addressed with small integers. They are generally written in hexadecimal and may be in the range of 1-ff. The mapping of a module number to its physical (rack and bay) location is arbitrary, and is handled by the IO6 PROM (for more information on module numbers, see The IP27prom Technical Reference Manual). The MMSC obtains this mapping when either the MMSC or the MSC is reset. It can also rebuild the current mappings using the **scan** command.

A partition is a collection of modules that is treated as a single logical system. They are not yet supported by the MMSC, but will be eventually.

One or more addresses can be combined into a *list*. A *list* of rack or module addresses is made up of one or more addresses or *ranges* separated by commas with no intervening white space. A *range*, in turn, is a series of contiguous values specified as two addresses separated by a hyphen, again with no intervening white space. Thus, the following are valid lists of module addresses:

```
1
1,3
1-f
1-4,7,39-3b
```

A list of bay addresses is also formed by one or more individual addresses separated by commas with no intervening white space. For bay addresses, comma separators are optional. However, ranges are not meaningful for lists of bay addresses, and so are not allowed.

Physical destinations

A *physical destination* is used to refer to one or more specific bays without regard to any logical module or partition designations. These are typically used for maintenance commands, such as powering a single bay off to replace a board. A complete physical destination specfication consists of one or more "rack/bay pairs":

```
rack rlist bay blist
```

The keywords **rack** and **bay** can be abbreviated as **r** and **b**, respectively.

A rack/bay pair selects each of the bays in *blist* on each of the racks in *rlist* (think of it as a "product"). Specifying more than one pair extends the selection accordingly (think of it as a "sum"). Thus, the long destination:

```
rack 1-3 bay u,L    rack 4 bay L
```

is equivalent to:

```
rack 1 bay u
rack 1 bay l
rack 2 bay u
rack 2 bay l
rack 3 bay u
rack 3 bay l
rack 4 bay l
```

Instead of specifying a list of addresses for *rlist* or *blist*, it is also possible one of several keywords. These include:

all (abbreviation "*")
: Selects all known *online* addresses. Addresses that are believed to be offline are skipped. If the MMSC's beliefs as to what is online or offline seem to be inaccurate, the **scan** command (see below) can be used to update them. A command can still be directed to a supposedly offline address by explicitly specifying it rather than using the **all** keyword.

local (abbreviation: ".")

Only valid for rlists

Selects the "local" rack, i.e. the rack containing the MMSC that initially accepts the command.

If the rack portion of a physical destination is omitted but the bay portion is not, then `rack local` is implied. If the bay portion is omitted but the rack portion is not, then `bay all` is implied.

Therefore, the following are all equivalent physical addresses in a four-rack system with two bays per rack.

```
rack all bay ul
rack 1,2-4
r * b *
```

Logical Destinations

A *logical destination* is used to refer to individual modules by their module numbers, rather than their physical position in a system. These are used more commonly that physical destinations, since these are the types of addresses used by IRIX and the various PROMs. A logical destination is formed by the keyword **module** followed by a module list:

```
module mlist
```

The keyword **module** can be abbreviated as **m** if desired.

As with physical destinations, *mlist* can be replaced with the keyword **all** to specify all known modules. There is a special case when no modules happen to be defined (this might occur, for example, if none of the modules have been powered on). In this case, `module all` is equivalent to `rack all bay all`.

Many MMSC commands pertain only to a particular rack and not a specific bay within it. If a logical address is used with such a command, the bay address that is implied by the module number is ignored.

Both physical and logical destinations may be specified for the same command, as long as one is not embedded in the middle of another. Thus, the following are valid:

```
rack * bay u module 3,5-7
r 1 b ul    m 8   rack 3 bay 1
```

But these are not, or at least may not give the expected results:

```
rack module 2 3 bay u     (invalid)
r 3 m 5 b u   (valid, but equivalent to:  r 3 b *   m 5   r . b u)
```

When the MMSC executes a command, logical destinations are converted internally to physical destinations using the module number mappings known to the MMSC at the time the command is executed. If the MMSC's mappings are no longer valid (for example, if a module has been powered off or is otherwise unavailable), then commands may time out or be directed to the wrong module. The **scan** command can be used to update the MMSC's mappings manually if necessary.

Default Destinations

Although it is possible to indicate a specific destination for any given command, this can be cumbersome in some cases, such as when there are several commands that need to go to the same complicated list of destinations. Therefore, the MMSC allows specifying a *default destination* by entering a destination with no additional command, or by using the **dest** command. Any subsequent commands that expect a destination will use this default value if no other destination is specified.

The initial default destination is `module all`.

Command Syntax

The general syntax for MMSC commands is fairly simple, other than the destination.

In general, an MMSC command looks like this:

```
[escape] [dest] command [args] <CR>
```

where:

escape
: is an *MMSC-escape* character, typically Control-T by default. This is used in console and MSC pass-through modes to indicate the beginning of an MMSC command. See below for further details.

dest
: is a destination specification, described above.

command
: is the name of an MMSC command, which are described below. *command* may be in upper or lower case (or both); it will be converted to upper case by the MMSC before it is processed. If *command* is not a valid MMSC command, it is assumed to be an MSC command and is passed to the bay(s) addressed by *dest* along with *args* without further translation.

args
: are zero or more arguments to *command*.

<CR>
: commands are always terminated with a carriage-return character. Note that this implies that neither the command nor its arguments may contain an embedded carriage-return character.

If an *MMSC-escape* character is entered in the middle of an MMSC command, all characters between it and the last *MMSC-escape* character are discarded. This can be useful if the current state of a console or other MMSC connection is currently unknown. In addition, the *kill* character (Control-U by default) will do the same thing.

The Command Set

The following is a description of the valid MMSC commands. If a command other than one of these is specified, it is assumed to be an MSC command and is passed along to the addressed MSC. Many commands accept an optional destination specification, while others do not. Those commands marked with "[*]" will honor destination specifications. Unless otherwise specified, each of these commands can be executed by a user at the *basic* authority level.

authority [*level* [*pw*]]

This command is only meaningful from terminals.

If *level* is not specified, then the authority level associated with the console from which the command was entered is displayed.

Otherwise, the authority level of the console from which the command was entered is changed to *level*. The valid authority levels are **basic**, **supervisor** and **service**. For more information on these, see Security on the Origin2000 Multi-Module System Controller .

If the **supervisor** or **service** level is selected and has a password associated with it, then *pw*, the appropriate password, must be specified after *level*. If the password is not correct, an error message will be printed and the current authority level will be unchanged.

bs *char*
bs ?

This command is only meaningful from terminals.

Set the *backspace* character of the console on which the command is entered to *char*, which may be either a single literal character, an integer value representing a single ASCII character, or a control sequence consisting of a carat followed by a single character (e.g. "^H"). Note that this only affects the *backspace* character used when typing MMSC or MSC commands. It does not affect the *backspace* character used in normal passthrough operation. The default *backspace* character is `Control-H`.

If **bs ?** is specified, then the current *backspace* will be printed. This implies that in order to use the "?" character as the actual *backspace* character, it is necessary to specify it using its numeric ASCII code.

cecho [on | off]

This command is only meaningful from terminals.

cecho controls the echoing of characters when an MMSC command is being typed in the CONSOLE input mode. It only affects the console on which it was entered. When command echo is **off**, any characters that are received after an *MMSC-escape* character are buffered internally by the MMSC but not echoed back to the input source. When command echo is **on**, then once an *MMSC-escape* character has been received, an MMSC prompt and the character following the escape character will be echoed to the input source. All additional input characters up to the

terminating *carriage-return* will be echoed as well. Once the *carriage-return* has been received, input processing returns to its original state.

If **cecho** is entered without arguments, the current **cecho** mode is toggled.

The default setting for **cecho** is **on**.

com *port*
com *port* **cmd on|off**
com *port* **function** *func*
com *port* **oob on|off**
com *port* **rxbuf|txbuf** *value*
com *port* **speed** *baudrate*

[*] **com** is used to set or display communications settings of the various serial ports on the MMSC.

port is a number from 1 to 6 corresponding to a particular serial port on the addressed MMSC:

port	label	default function
1	CONSOLE	TERMINAL
2	UPPER BAY	UPPER
3	LOWER BAY	LOWER
4	BASEIO TTY1	SYSTEM
5	ALTERNATE CONSOLE	ALTCONS
6	TEST	DEBUG

If no other arguments are specified, then the current communication settings for the selected port are displayed.

The **cmd** subcommand is used to indicate whether or not MMSC and MSC commands are accepted from a port. If **off** is specified, then MMSC and MSC commands are not accepted from the port. For ports associated with a terminal (namely ports with functions TERMINAL or ALTCONS), this has the effect of causing the *MMSC-escape* character to be ignored when the port is in the CONSOLE input mode. This can be useful if a particular console is in an insecure location, for example. For ports associated with the system (e.g. functions SYSTEM or DAEMON), this disables the Out Of Band function FFSC_COMMAND. If **on** is specified, then MMSC and MSC commands are accepted from the port. The only way to place a port that has specified `cmd off` into some other input mode is to "steal" the console from a different port (see steal). Thus, *be very careful that at least one console has specified* `cmd on`. The default setting for ports 1 and 5 is `cmd on`, and for ports 2, 3, 4 and 6 is `cmd off`.

The **function** subcommand is used to set the function associated with the selected port to *func*. Valid functions include:

func	**Function of associated port**
TERMINAL	Communication with the user terminal device.
UPPER	Communication with the MSC in the upper bay of the rack.
LOWER	Communication with the MSC in the lower bay of the rack.
SYSTEM	Communication with the operating system. This is the port through which both the user and the MMSC communicate with IRIX. It is typically connected to the TTY1 port on the master BASEIO card.
ALTCONS	Remote service port. This would typically be connected to a modem that is used for communication with an SGI service center.
DAEMON	Communication with a system controller daemon on IRIX, such as *ffscd*. Such a daemon would ordinarily be used for generating bar graph data on the MMSC display. This port would typically be connected to a second serial port on BASEIO card.
DEBUG	MMSC debugging log. This is mostly useful if some sort of MMSC error has occurred. In that event, the debugging log may contain additional information.

Any given function can be assigned to at most one port. If *func* specifies a function that is already assigned to another port, then the other port will have its port changed to an "unassigned" state. Care should be taken to ensure that, at minimum, the TERMINAL function is assigned to a port, or else it may become impossible to communicate with the MMSC.

The **oob** subcommand is used to indicate whether or not Out Of Band (OOB) data received from a port should be intercepted and processed. OOB messages are used by programs such as *ffscd* to display bar graphs and other information on the MMSC display. The actual OOB message protocol is not described in this document. If **on** is specified, then OOB message processing will be done for data received by the port. For ports associated with the system (namely ports with functions SYSTEM or DAEMON), any data preceded by the *Out-Of-Band Prefix* character will be interpreted as an OOB message. The MMSC will perform the requested action and respond as specified in the OOB message protocol. For ports associated with a terminal (namely ports with functions TERMINAL or ALTCONS), OOB message processing means simply doubling the *Out-Of-Band Prefix* character before sending it to the system. This prevents it from being interpreted by IRIX as the beginning of an Out Of Band message. If **off** is specified, then Out Of Band messages that are received from the port are ignored and passed through without alteration. The default setting for all ports is `oob off`.

The **rxbuf** and **txbuf** subcommands are used to change the size of the serial port's receive and transmit buffers, respectively. The default for both is 4096. If the **system** and **terminal** ports have different speeds, it may be necessary to increase the size of the transmit buffer on the slower port or else data may be lost. In extreme cases, serial buffer overflows have been known to knock a serial port out of service, so underestimating the buffer size should be carefully avoided.

The **speed** subcommand is used to set the baud rate of the selected port to *baudrate*. Valid baud rates include 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600 and 115200.

Function changes do not take effect until the MMSC has been reset (see **reset_mmsc**). Speed

changes take effect immediately. The user must have *service* authority to specify the **cmd**, **function**, **oob** or **speed** subcommands.

console [*args*]
cons [*args*]

If *args* are present, they are sent along to the system console port but otherwise ignored. *args* is assumed to start at the first non-blank character following the command.

If *args* is not specified, console pass-through mode is entered. Any subsequent input is passed through to the system console port and any output from that system console port is echoed here. There are two exceptions:

1. Any input preceded by the MMSC escape character, up to and including the first subsequent carriage-return character, is processed by the MMSC rather than being passed through. Any output that is generated by this MMSC command may be discarded or echoed, depending on the current **rmsg** setting.

2. Two MMSC escape characters in a row will cause a single MMSC escape character to be sent to the system console port, then otherwise ignored by the MMSC.

dest

Displays and/or modifies the current default destination. If no address was specified prior to the **dest** keyword, then the console's current default destination will be displayed. Otherwise, the specified address will become the new default destination for the console on which the command was entered. Any subsequent command that specifies no address will automatically use the default destination instead.

direct

Enter direct input mode on both the current console and the "other" console on the MMSC. If the device on which the command is entered happens to be the normal terminal device, then the "other" console is the alternate console device. Likewise, if the command is entered from the alternate console device, the "other" console is the normal terminal device.

All input from the current console will be sent directly to the other console and vice versa. The only exception is that typing the *exit* character on the current console will cause both consoles to leave direct mode and return to their usual input modes (console mode for the terminal and RAT mode for the alternate console).

Direct mode is mostly useful for programming a modem that happens to be attached to the "other" console.

A user must have *service* authority to execute this command.

downloader disable
downloader enable

[*] This command is used to set or clear the "serial downloader" flag in the addressed MMSC's initialization PROM. The command **downloader enable** will set the flag, and the command **downloader disable** will clear it.

When the "serial downloader" flag is set, the initialization PROM on the MMSC's single-board computer will load the "serial downloader" rather than the normal MMSC firmware. The serial downloader can be used to load a new firmware image into the MMSC's flash RAM. For more details on loading the MMSC firmware with the serial downloader, see the document Flashing MMSC Firmware.

Note that once the MMSC has been initialized with the "serial downloader" flag set, there is no way to return to normal MMSC operation until a new MMSC firmware image is loaded. If you inadvertantly set the "serial downloader" flag on the wrong rack, use the **downloader disable** command to reset the flag as soon as possible. If an MMSC is reset, either with the **reset_mmsc** command or because it has lost power, it will be necessary to follow the procedure in Using the Serial Downloader to restore normal operation.

The **downloader enable** command is typically used to place an MMSC that does not have an attached display into serial downloader mode. A rack which does have a display can be placed into serial downloader mode by holding down the bottom two buttons of the display (ENTER and DOWN) while the MMSC is being powered on.

A user must have *service* authority to execute this command.

end *char*
end ?

This command is only meaningful from terminals.

Set the *end-of-command* character for the console on which the command was entered to *char*. *char* may be either a single literal character, an integer value representing a single ASCII character, or a control sequence consisting of a carat followed by a single character (e.g. "^M"). The *end-of-command* character is used to signal the end of an MMSC command while in the console or MMSC input modes. The default *end-of-command* character is `Control-M` (a.k.a *carriage-return*).

If **end ?** is specified, then the current *end-of-command* character will be printed. This implies that in order to use the "?" character as the actual *end-of-command* character, it is necessary to specify it using its numeric ASCII code.

esc *char*
esc ?

This command is only meaningful from terminals.

Set the *MMSC-escape* character for the console on which the command was entered to *char*. *char* may be either a single literal character, an integer value representing a single ASCII character, or a control sequence consisting of a carat followed by a single character (e.g. "^T").

The *MMSC-escape* character is used to signal the beginning of an MMSC command while in console mode. The default *MMSC-escape* character is `Control-T`. Note that changing the *MMSC-escape* character does not affect the MSC escape character, which is always `Control-T`.

If **esc ?** is specified, then the current *MMSC-escape* character will be printed. This implies that in order to use the "?" character as the actual *MMSC-escape* character, it is necessary to specify it using its numeric ASCII code.

exit
exit *char*
exit ?

If neither *char* nor "**?**" is specified, leave the current input mode and return to the console mode. If you are already in console mode, this command has no effect.

In MSC and MMSC mode, the same effect can be accomplished by simply typing the *exit* character. Notice that the *only* way to leave MSC mode is by typing the *exit* character, since MMSC commands are not accepted in MSC mode.

If *char* is specified, set the *exit* character for the console on which the command was entered to *char*. *char* may be either a single literal character, an integer value representing a single ASCII character, or a control sequence consisting of a carat followed by a single character (e.g. "^E").

The default *exit* character is `Control-E` (for reasons I do not fathom). An alternative might be `Control-D`, the standard Unix EOF character that is often used to exit shells.

If **exit ?** is specified, then the current *exit* character will be printed. This implies that in order to use the "?" character as the actual *exit* character, it is necessary to specify it using its numeric ASCII code.

flash [from system]
flash from console

[*] Flashes a new firmware image into non-volatile storage on the MMSC for the addressed rack. An error will occur if more than one rack is addressed by this command. The bay portion of the destination is ignored.

In the first form, `flash from system` (the words "`from system`" are optional), the image is provided to the MMSC from the console of a running IRIX system using the `flashmmsc`(1m) command. The `flashmmsc` IRIX command and the **flash** MMSC command must both be issued from the same terminal device. Typically, the `flashmmsc` command is issued first with the `-m` option, then an *MMSC-escape* character is typed and the **flash** command entered from the same terminal.

In the second form, `flash from console`, the image is read from the terminal device itself. This is useful when the user terminal is in fact a terminal emulator (perhaps on a PC) capable of XMODEM file transfer. For this form of the **flash** command, one would issue the **flash** command to the MMSC first, then arrange for an XMODEM or XMODEM-1K file transfer of the firmware image from the user terminal. The use of XMODEM-1K may improve the transfer time by as much as 50%.

Be forewarned that this command takes a *long* time to run. In addition to the time it takes to download the firmware image over a serial line (MMSC firmware images are approximately 1MB in size) an additional 100 seconds is required for clearing out the flash RAM prior to installing the

new image. **Every effort should be made not to interrupt the MMSC while this command is taking place.** Early versions of the MMSC do not have sufficient flash storage to hold two separate images; therefore attempting to flash a new image will necessarily wipe out the old firmware image, even if the attempt turns out to be unsuccessful. Do not reboot an MMSC that has tried and failed to flash a new firmware image. If this does occur, there is an emergency procedure to flash new firmware onto it if necessary. For more information on this procedure, see Flashing MMSC Firmware.

A user must have *service* authority to execute this command.

help [*cmd* | ALL]

This command is only meaningful from terminals.

Displays help about the MMSC commands. If no argument is specified, then a list of available commands is printed. If a command is specified, more specific information about that particular command is printed. If **ALL** is specified, specific information about all available commands is printed.

In all cases, the data will be displayed as paged output (see **pager** for details).

kill *char*
kill ?

This command is only meaningful from terminals.

Set the *kill* character for the console on which the command was entered to *char*. *char* may be either a single literal character, an integer value representing a single ASCII character, or a control sequence consisting of a carat followed by a single character (e.g. "^U"). The *kill* character is used while typing an MMSC/MSC command in console mode to cancel any characters that have been typed since the last *MMSC-escape* character. In MMSC mode, the *kill* character cancels any characters that have been typed since the last prompt was printed.

The default *kill* character is `Control-U`.

Note that this does not affect the normal Unix kill character associated with the console when it is in passthrough mode.

If **kill ?** is specified, then the current *kill* character will be printed. This implies that in order to use the "?" character as the actual *kill* character, it is necessary to specify it using its numeric ASCII code.

log [*log*] **clear**
log [*log*] [**dump**] [*num*]
log [*log*] **disable|enable**
log [*log*] **info|?**
log [*log*] **lines** *num*
log [*log*] **length** *num*

[*] The **log** command is used to manipulate the various message logs maintained by the MMSC.

The *log* argument specifies which log is to be manipulated. It may be one of the following values:

log	**function**
MSC	A log of all messages and other output from the addressed MSC.
SYSTEM	A log of all output generated by the Base I/O board on the addressed rack, typically consisting of output from the operating system.
TERMINAL	A log of all output that has been sent to the main terminal console attached to the addressed rack. This is different from the SYSTEM log in that it may contain output from the MSC and MMSC input modes, as well as any messages received from an MSC (typically generated by various PROM's during system initialization).
ALTCONS	A log of all output that has been sent to the alternate console attached to the addressed rack.
DEBUG	A log of all debugging messages produced by the addressed rack's MMSC.
DISPLAY	A log of each command issued by the MMSC display and the corresponding response.

If *log* is not specified, the default is **msc**.

The **clear** subcommand causes the contents of the specified log to be discarded. Logging will continue as usual unless it has been disabled with the **disable** subcommand.

The **dump** subcommand causes the contents of the specified log to be dumped on the current terminal. The data will be displayed as paged output (see **pager** for details). If *num* is specified, then only the last *num* lines of the log will be dumped. This is the default subcommand if none is specified.

The **disable** subcommand will cause logging of data to the specified log to be stopped. Any data that is currently in the log will remain unchanged. The **enable** subcommand will resume logging.

The **info** subcommand shows information about the specified log, such as its size and its enable/disable state. The **?** subcommand does the same thing.

The **lines** subcommand sets the maximum size of the specified log to *num* lines, where a "line" is defined to be a sequence of characters ending with either CR/LF or LF/CR. The actual number of lines that can be held in a log is also subject to the average line length (see below). The change will not take effect until the MMSC has been reset (see **reset_mmsc**). If **default** is specified for *num* then the actual value will be taken from the **LOG_DFLT_NUMLINES** environment variable.

The **length** subcommand sets the average line length of lines in the buffer to *num*. The total amount of storage set aside for log data is the product of the log's **lines** and **length** values. If the **length** value is too small, then the log might wrap before it contains the maximum number of lines. If the **length** value is too large, it will waste storage. A change to the length value will not take effect until the MMSC has been reset (see **reset_mmsc**). If **default** is specified for *num* then

the actual value will be taken from the **LOG_DFLT_LINELEN** environment variable.

The **lines**, **length** and **enable/disable** settings for each log are separate from the other logs. They are saved in NVRAM and will be restored after the MMSC has been reset.

A user must have *service* authority to execute the **lines**, **length**, **enable** or **disable** subcommands.

mmsc [*args*]

[*] If *args* are present, they are sent along as a command to the MMSC on the addressed rack(s). *args* may be a valid MMSC command *only*; MSC commands are not recognized in this case and will result in an ERROR CMD response. This usage is mostly useful for avoiding ambiguity with MSC commands. It can also be used to send a normally "local" command (e.g. **esc**) to a remote MMSC. *args* is assumed to start at the first non-blank character following the command.

If *args* is not specified, enter MMSC mode. In MMSC mode, the MMSC prompt:

```
MMSC>
```

is displayed, and output from the IO6 is held and/or discarded. Any input from the terminal keyboard is handled directly by the local MMSC, using the address specified with the **mmsc** command as the default destination. To leave MMSC mode, use the command **exit** or type the current *exit* character.

Ordinarily, the *MMSC-escape* character has no special meaning in MMSC mode and will processed like any other character. However, if the *first* character of a line is the *MMSC-escape* character, it will instead be discarded. This allows automated tools (and engineers with bad memories) to type the same sequence of characters in order to issue an MMSC or MSC command from either MMSC or console mode.

mmsg [on | terse | off]
mmsg rack [*rackid*]
mmsg altrack [*rackid*]

[*] The first form of this command is only meaningful from terminals and ignores any destination. The second and third forms of this command do honor a destination.

mmsg controls the display of unsolicited messages from the MSC (e.g. those sent to the MMSC in response to an MSC "acp" command). It only affects the console on which it was entered. When **mmsg** is **off**, unsolicited MSC messages are discarded silently. When **mmsg** is **on**, all unsolicited MSC messages are echoed to the input source on their own lines (i.e. preceded and followed by *carriage-return/linefeed* as needed), prefixed with a string identifying the originating MSC. The prefix is of the form:

```
R<rack-addr><bay-addr>-
```

For example:

```
R1U-This is a message from rack 1 bay U
```

When **mmsg** is **terse**, unsolicited MSC messages are still echoed on their own line, but the identifying prefix is omitted.

If **mmsg** is entered without arguments, the current **mmsg** mode is cycled in some undefined, but consistent, order.

If **mmsg** is entered in the second form, then any unsolicited MSC messages generated by the addressed racks will be sent to the TERMINAL device attached to the rack specified by *rackid*. Specifying a *rackid* of **none** will cause MSC messages to be discarded, and so is functionally equivalent to **mmsg off**.

Because an MMSC is not able to detect the presence of a BaseIO card on a particular rack, it is necessary to run this command at least once when the system is first set up in order to receive unsolicited MSC messages. The command would typically be of the form "`rack all mmsg rack` *rackid*"

mmsg rack without a *rackid* will return the ID of the rack currently designated to receive unsolicited MSC messages.

The third form of **mmsg** is similar to the second form except that it specifies a rack whose ALTERNATE console port should receive unsolicited messages. The *rackids* specified for **rack** and **altrack** do not have to be the same. Either or both may be **none**.

The default **mmsg** mode is **off**. The default rack to receive unsolicited MSC messages on both the TERMINAL and ALTERNATE ports is 1.

msc [*args*]

[*] If *args* are present, they are sent along to the MSC on the addressed bay(s), but otherwise ignored. They will be prefixed with the MSC escape character, CTRL-T. This could be useful for forcing a command to be passed on to the MSC, such as when the command has the same name as an MMSC command. *args* is assumed to start at the first non-blank character following the command. For information on individual MSC commands, see the IP27 PROM Technical Reference Manual .

If *args* is not specified, MSC mode is entered. MSC mode is (supposed to be) functionally equivalent to having a direct connection to the MSC port: all keyboard input is echoed directly to the addressed MSC, and all output from the MSC is echoed to the user terminal without modification. There are two exceptions. The first is that the current *exit-character* will cause an exit from MSC mode and return you to the previous input mode. (There is no way to send an *exit-character* to the MSC in MSC mode.) The second exception is that any *output* from the MSC that begins with a `Control-T` character is assumed to be a message to the MMSC; all characters up to and including the next carriage return will be handled by the MMSC on the same rack as the MSC.

When the **msc** command is used without *args* specified, it may address only one MSC or else an error will occur.

Note that when an MSC has been attached to a console with the **msc** command, that MSC is no

longer available to perform commands issued from other consoles. For example, if the alternate console is in MSC mode, then any MSC commands issued from the normal terminal or the display will be rejected. The error message in this case is typically `ERROR INUSE`.

Output from the system will be held and/or discarded while in MSC mode. Messages from other MSC's will be displayed only after a CR/LF sequence has been echoed to the user terminal, or if the terminal has been inactive for some period of time (approximately 2 seconds).

nap_time [*value* | **default**]
This command is only meaningful from a terminal.

If *value* is specified, then the *nap interval* for the console on which the command is entered is set to *value* microseconds. The *nap interval* is the frequency with which the MMSC will attempt to perform various clean-up tasks when the console is otherwise idle. The most visible effect of these clean-up tasks is that any incomplete messages which have been received from an MSC will be printed. (An "incomplete message" is one which is not terminated with a CR/LF combination.)

If **default** is specified instead of a *value*, the nap interval will be reset to the system default, which is normally 2 seconds.

If neither *value* nor **default** is specified, then the current nap interval will be printed.

A user must have *service* authority to execute this command.

options [*value*]
This command is only meaningful from a terminal.

Sets the option flags to *value*. The currently defined option flags, which can be logically OR'ed together in almost any combination, include:

0x00000001	Do not display the MMSC prompt in console mode until the character *after* the MMSC escape character has been typed. This may be useful for people who expect to send the MMSC escape to the system (i.e. by typing it twice) from time to time.
0x00000002	If the end-of-command character (usually *carriage-return*) is typed immediately after the MMSC escape character, enter MMSC mode on the local rack. This would be equivalent to typing the command `rack local mmsc`. This may be useful for people who use mmsc mode frequently.
0x00000004	The MMSC always sends an `ech 0` command to the MSC before it takes control of it, so when control is given to the user after entering MSC mode, echoing will not normally be enabled. When this option is set, an `ech 1` command will be sent to the MSC prior to entering MSC mode. This should cause the MSC to echo keyboard input as it is typed.
0x00000008	Ordinarily, blank messages from the MSC are ignored, even with **mmsg on**. When this option is set, blank messages from the MSC will be printed like any other message.
0x80000000	Rob Mode

pager { **back** | **fwd** | **quit** } *char*
pager [**info** | **?**]
pager lines *val*
pager { **on** | **off** }

The **pager** command controls the built-in pager used by the MMSC to display large blocks of output, such as logs or help messages. It is conceptually similar to the standard unix command *more*.

The **back** subcommand is used to specify the character that is typed to scroll backwards through the paged output. *char* may be either a single literal character, an integer value representing a single ASCII character, or a control sequence consisting of a carat followed by a single character (e.g. "^U"). Notice that the only way to specify a <space> character is with its integer value, 32. The default **back** character is "b".

Similarly, the **fwd** subcommand is used to specify the character that is typed to scroll forwards through the output, and the **quit** subcommand specifies the character that is typed to discontinue the output. The default **fwd** character is <space>, and the default **quit** character is "q".

The **info** subcommand is used to display information about the current pager settings. The **?** subcommand does the same thing. This is the default if no subcommand is specified.

The **lines** subcommand is used to set the number of lines in a single page of output. After *val* lines have been displayed, a trailer line will be printed and output will be halted until the **fwd**, **back** or **quit** character has been typed. Specifying **default** for *val* will cause the default value to be restored. The default setting is controlled by the PAGE_DFLT_LINES environment variable, which is typically 23 lines (leaving room for a single trailer line on a standard 24-line terminal).

The **off** subcommand turns off paged output. Any output that would ordinarily be paged is instead sent to the terminal in one single, large block. The **on** subcommand reverses this.

The **back**, **fwd**, **quit** and **on/off** settings for each console are separate from the other consoles. They are saved in NVRAM and will be restored after the MMSC has been reset.

password { **set** | **setmmsc** } *passwd string*
password unset *passwd*

[*] The **password** command changes the *passwd* password. Unless otherwise specified, you must be at the *supervisor* or *service* authority level to use this command.

passwd may be one of the following values:

passwd	Corresponding password
msc	The password on the addressed MSC. This is the same password that is specified with the MSC **pas** command. This will be passed along to the MSC before any restricted commands when the user is in the *supervisor* or *service* authority level.
supervisor	The password used to enter the *supervisor* authority level.
service	The password used to enter the *service* authority level. You must be at the *service* authority level to change this password.

The **set** subcommand changes the password *passwd* to *string*. If *passwd* is **msc**, then the password will be changed on the MSC first by passing the command "`pas s` *string*" to the MSC. If the command does not succeed, then the password will not be changed on the MMSC either.

The **setmmsc** subcommand is identical to the **set** subcommand, except when *passwd* is **msc**. In that case, the password will *not* be set on the MSC first. This is useful if the password had been changed on the MSC without the MMSC's knowledge (for example, while in the MSC input mode).

The **unset** subcommand is used to remove the password associated with *passwd*. Note that the MSC does not have a notion of "removing the password", so specifying a *passwd* of **msc** will affect the MMSC only. In this case, no attempt is made to set or revoke the MSC supervisor mode prior to sending commands to it.

printenv [all] [default]

[*] Displays the names and values of any environment variables that have non-default settings. If **all** is specified, all of the known environment variables will be listed. Those with non-default values will have their values printed as well. If **default** is specified, the default value for each variable will also be displayed. The data will be displayed as paged output (see **pager** for details).

rackid [*value*]

[*] If *value* is specified, the rack ID of the addressed rack is changed from its current setting to *value*. This setting is saved in NVRAM and should persist across power cycles and resets. The change becomes effective immediately. This may cause trouble if you happen to be in MMSC or MSC mode when the **rackid** command is issued, since the rack ID that was previously addressed by MMSC/MSC mode is no longer valid.

If *value* is not specified, then the destination is ignored and the current rack ID of the **local** rack is returned.

A user must have *service* authority to execute this command.

rat

Places the console into RAT (Remote Access Tool) mode.

RAT mode is sort of a combination of Console, MMSC and MSC input modes that is supposed to make the MMSC behave somewhat like an MSC. It is intended mainly for use by certain automated service tools. It has the following characteristics:

- User input is not echoed.
- Any input that is not preceded by the *MMSC-escape* character is discarded.
- User input is treated as if it had been entered in MMSC mode: if it is a valid MMSC command, it is processed as such, otherwise it is passed on to the addressed MSC(s).
- Response and ELSC messages are printed in the same way as for Console mode.

reset_mmsc

[*] Restarts the addressed MMSC(s). This may be useful after flashing a new firmware image onto the MMSC so that the changes can be picked up. This can also be used if the MMSC and/or system console has hung even though the system itself is still running. Of course, if it is used for this second purpose, then it would be appropriate to file a bug report/service call.

This does *not* affect IRIX or any other part of the system. Only the addressed full-featured system controllers are rebooted.

For historical reasons, the command **reboot_mmsc** does the same thing as **reset_mmsc**.

A user must have *service* authority to execute this command.

reset_nvram

[*] Resets the contents of non-volatile storage on the addressed MMSC(s) to default values. Obviously, this command should be used with caution.

Values that are stored in non-volatile storage include the rack ID, MMSC serial port speeds, and the **mmsg** rack settings.

A user must have *service* authority to execute this command.

rmsg [on | error | off]

This command is only meaningful from terminal devices.

rmsg controls the echoing of responses to MSC or MMSC commands. It only affects the console on which the command was entered. When **rmsg** is **off**, the response string that is generated by an MSC or MMSC command is discarded silently. When **rmsg** is **on**, all command responses are echoed to the input source on their own lines (i.e. preceded and followed with *carriage-return/linefeed* as needed). When **rmsg** is **error**, only error responses (i.e. those not prefixed by "OK" or "OFFLINE") are echoed.

If **rmsg** is entered without arguments, the current **rmsg** mode is cycled in some undefined, but consistent, order.

In either **error** or **on** mode, the terminal does not return to its normal passthrough mode until a response has actually been received or some timeout has been exceeded.

The default setting for **rmsg** is **on**.

scan

[*] Check for the presence or absence of the addressed MSC's and update the internal table of module number to physical address mappings. This may be necessary if a module has become unavailable or its module number has changed without the MMSC's knowledge.

Unlike most commands, if "`bay all`" is specified in the destination for this command, then *all* bays in the specified racks will be probed, even if they were originally thought to be offline. This is useful for determining if a formerly offline MSC has been brought online somehow. "`module all`" behaves normally and so is not useful for finding new modules.

A user must have *service* authority to execute this command.

setenv *var* [[=] *value*]

[*] Change the setting of environment variable *var* to *value*. *value* must normally be an integer. The "=" separating *var* and *value* is optional. If it is specified, there should be no whitespace separating it from either *var* or *value*. If *value* is not specified, then the current setting of the environment variable *var* is displayed.

Only environment variables known to the MMSC may be stored or displayed. Valid environment variables include:

Variable	Function	Default value
DEBOUNCE_DELAY	The debounce delay (in microseconds) used by the MMSC display switches. It may be helpful to increase this value if single key presses are being registered multiple times. On the other hand, if some key presses seem to ignored when pressing keys rapidly, it may help to decrease this value.	200000
LOG_DFLT_NUMLINES	The default number of lines in a log. Once a log has been filled with this many lines of data, older lines will be discarded to make room for new lines. The MMSC must be reset (see **reset_mmsc**) for this setting to take effect.	200
LOG_DFLT_LINELEN	The default average line length for a log. The total amount of storage allocated for logged messages is NUMLINES * LINELEN. If many lines in a log are larger than LINELEN characters in length, then it may be necessary to discard older lines even before the log contains NUMLINES lines. The MMSC must be reset (see **reset_mmsc**) for this setting to take effect.	80
PAGE_DFLT_LINES	The default number of lines per page when paging output.	23
PWR_DELAY	The MSC "pwr u" and "pwr c" commands may be intercepted in order to sequence the power to two or more racks one rack at a time to avoid an excessive power surge. (See Intercepted MSC Commands, below.) This variable sets the number of microseconds to wait between "pwr" commands.	5000000

A user must have *service* authority to execute this command.

steal

This command is only meaningful from the terminal or alternate console device, and only when it is in MMSC mode. If another device is currently in console mode, it is placed into MMSC mode. Then the device on which the command was entered is placed into console mode. In effect, the system console is "stolen" by the device on which the command was entered.

A user must have *supervisor* authority to execute this command.

unsetenv *var*

[*] Restore the default value for environment variable *var*.

unsteal

This command is only meaningful from the terminal or alternate console device, and only when it is in console mode. It places the "other" device into console mode, effectively undoing a steal command. If the device on which the command is entered happens to be the normal terminal device, then the "other" console is the alternate console device. Likewise, if the command is entered from the alternate console device, the "other" console is the normal terminal device. The device on which the command was entered is then placed into either MMSC mode (when the device is the terminal) or RAT mode (when the device is the alternate console).

A user must have *supervisor* authority to execute this command.

ver

[*] Returns a string indicating the MMSC firmware revision. This command returns a different string than the MSC "ver" command, and so can be used from a tty device to determine if it is attached to an MSC or an MMSC. The format of the response is "MMSC *major.minor*", for example:

```
MMSC 1.2
```

A user must have *service* authority to execute this command.

carriage-return

If neither a destination nor a command follows an MMSC escape character (i.e. the first character of the command is carriage-return, essentially an "empty command") then the command is ignored.

If a destination but no command is specified, then the destination becomes the new "default destination" (see above).

Intercepted MSC Commands

Certain MSC commands are intercepted by the MMSC before being passed along to the MSC itself.

This interception is *not* performed when the commands are sent to an MSC using the **msc** ***args*** command. In general, you should avoid sending commands to an MSC with the **MSC** ***args*** command whenever possible, since the additional actions performed by the MMSC are usually beneficial. The intercepted commands and the additional actions associated with them include:

MSC command	intercepted actions
pas s *pw*	The command is converted into the MMSC command "**password set msc** *pw*", which should have the same effect on the MSC password, plus the added benefit of setting the password on the MMSC as well.
pwr u **pwr c**	The command will be forwarded to the addressed MSC's, but one rack at a time with a time delay between each. This will prevent all of the modules from being powered on nearly simultaneously, which could cause a fairly severe power surge.
ver	The MMSC version string, rather than the MSC version, is printed, as documented above.

up a level

Send questions and comments to: Rob Bradshaw

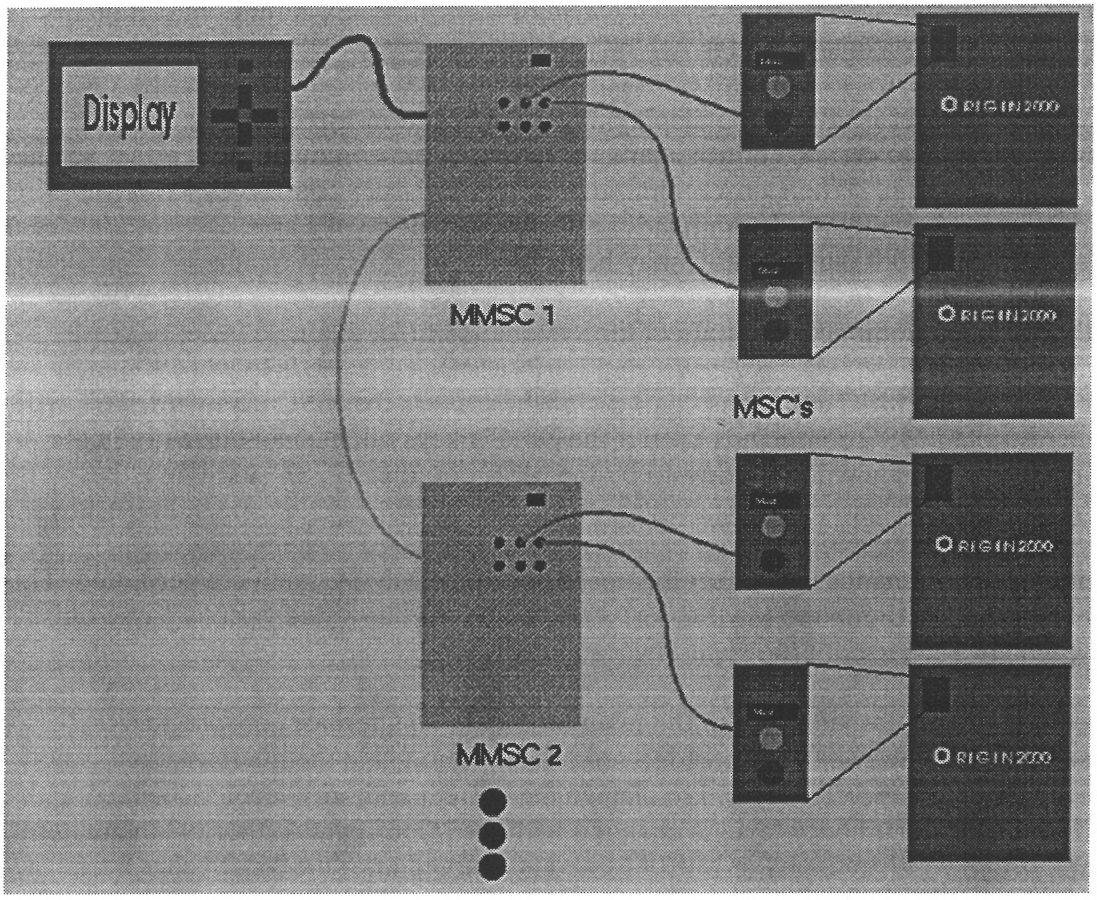
Last modified: Wed Dec 11 19:24:12 PST

Security on the Origin2000 Multi-Module System Controller

The Origin2000 Multi-Module System Controller (MMSC) provides the main system control and console interface on an Origin2000 system composed of two or more modules. Consequently, it is not unreasonable to expect that the MMSC should provide at least a small measure of protection against unauthorized access. This page documents the security features that are present in the MMSC, as well as potential security holes.

Architectural overview

First, a brief overview of the way system controllers are set up on multi-module Origin2000 systems.



A single module always has a Module System Controller (MSC) which is used for the normal system controller functions like reset, power, NMI, etc. Each MSC has a physical keyswitch which enables "dangerous" commands such as power down or reset. If the keyswitch is in the "disable dangerous commands" position, one can still issue dangerous commands over a console connection by providing a password. Every MSC has a password; by default, it is "none".

If a system has two or more modules, then each rack of two modules will have a Multi-Module System

Controller (MMSC) attached to the two MSCs. In a system with two or more racks, each rack will have its own MMSC and the MMSC's will all be linked together over a *private* network. Exactly one MMSC in the entire complex will have an LCD display with six buttons. A user can issue commands to the MMSC(s) either from the display or from a terminal attached to one of the MMSCs.

The purpose of the MMSC is to provide a single point of control for a multi-module system. Thus, rather than having to issue (for example) a "power on" command to each MSC, one can issue it once to the MMSC and the MMSC will send the command to each of the MSCs. To a user, the MMSC is equivalent to the "sysctlr" port on Challenge systems. However, an MMSC's only point of attachment to the rest of the system is through the MSC's - it cannot do anything to a particular module that the module's MSC can't do.

MMSC Authority Levels

Like the MSC, the MMSC restricts access to certain commands that could potentially disrupt the system. The available commands are determined by the *authority level* of the user. Each authority level other than *basic* may have a password associated with it.

In the *basic* authority level, the user is allowed to display information and manipulate the console characteristics (for example, change certain control character settings), but commands that affect actual system operation will not be allowed. When a user with *basic* authority issues commands to an MSC, each command will be preceded by the MSC command "`pas`", which will revoke any supervisor/diagnostic mode password that the MSC may have.

In the *supervisor* authority level, commands that affect the system itself (for example, reset or power commands) are permitted. When a user with *supervisor* authority issues commands to an MSC, each command will be preceded by the MSC command "`pas` *pw*", where *pw* is the MSC password. This will allow the user to issue restricted MSC commands even if the MSC's keyswitch is not currently in the diagnostic position. These commands will be followed by the MSC command "`pas`" to revoke the password.

In the *service* authority level, commands that affect the operation of the MMSC itself are permitted. This would include commands to reconfigure the MMSC serial ports, and to change the various passwords. The *service* authority level is a superset of the *supervisor* authority level.

There are separate authority levels associated with the main terminal, the alternate console, and the MMSC display. Thus, the display may be authorized to reset the system while the main terminal is not, for example. The **authority** command can be used to change the current authority level associated with a user on a console. The display's authority level is changed with the "ConfigurelAuthority" menu item. Each user's current authority level is retained in NVRAM and is restored after the MMSC has been reset. The default authority level for each user is *service*.

There is one password associated with each authority level (other than *basic*) on each MMSC. To change to a higher authority level on a console, it is necessary to specify the appropriate password along with the **authority** command. No password is required to switch to a lower authority level. The **password** command can be used to change the current password associated with an authority level.

Notice that no password is required to change authority levels on the display at this time. Because

someone who has access to the display also has access to the cables and circuit breakers, this was considered to be a relatively low-grade risk. However, the ability to require a password for changing the display's authority level is still planned for a future release of the MMSC firmware.

The current authority level is valid on all MMSC's in a system, even if the other MMSC's have different passwords for that authority level. Therefore, unless the terminal, alternate console and display are physically separate and secure, and attached to different MMSC's, there is little advantage in using different passwords for the same authority level on different MMSC's.

Clearing the MMSC Passwords

An MMSC's passwords, as well as its internal copies of the MSC passwords, are stored in NVRAM on the MMSC itself. If the system adminstrator puts the system into a low authority level and then subsequently forgets the appropriate passwords, the system could effectively be rendered unusable. Therefore, the MMSC has a procedure for clearing out its NVRAM in order to recover from such emergencies.

Unless the system administrator is very agile, the procedure requires two people. One person must go to the back of the rack containing the MMSC with the display and cut the power to the MMSC (most conveniently by unplugging it from its power supply -- it is *not* necessary to cut power to the entire rack). The second person stands at the MMSC display and holds down the LEFT and RIGHT buttons. The first person then restores power to the MMSC. Once the MMSC has booted far enough to be able to read the display buttons, it will clear the contents of its NVRAM and print an appropriate message on its display. The MMSC then resumes booting in the normal fashion.

Once the MMSC has finished booting, the passwords will have been cleared, and the system administrator can set them to new values. If necessary, the passwords on the other MMSC can be set as well. It will also be necessary to restore any other settings that may have been stored in NVRAM, such as serial port speeds, rack IDs, and the various MSC passwords.

Note that clearing an MMSC's NVRAM also clears its copies of the MSC passwords, thus making it incapable of issuing destructive commands to the MSC (assuming the MSC has a non-default password in the first place). On a carefully secured system (i.e. one on which non-default passwords have been set everywhere), this will render the MMSC effectively useless: a user would be able to reconfigure the MMSC, but would not be able to do issue destructive commands to the actual modules. Thus, the ability to clear an MMSC's passwords should not constitute an especially severe security risk.

Rob Bradshaw
Last modified: Wed Dec 4 20:10:46 PST

Flashing MMSC Firmware

Note: the MMSC is also known by its older name, "FFSC". Similarly, the MSC was also once known as the "ELSC".

The firmware for the Origin2000 MMSC (Multi-Module System Controller) resides in non-volatile storage on the MMSC itself. There are times when it is necessary to replace the existing firmware with a new (or at least different) firmware image. There are a couple of ways to accomplish this, depending on the circumstances of the installation and the available equipment. This document describes these different procedures and explains when each is necessary.

Basics

The MMSC has six serial ports, each capable of running at speeds of up to 115200 Baud. The ports are *not* interchangable; each has a specific function, as follows:

Port	Function
COM1	Terminal device (the "tty1" port on a lego rack system is actually connected to this port).
COM2	Connection to the MSC (Module System Controller) in the upper bay of the rack.
COM3	Connection to the MSC in the lower bay of the rack.
COM4	Connection to the master IO6 board.
COM5	Remote service modem (*not* a general purpose modem port). Also used for direct firmware downloads in emergencies.
COM6	MMSC debugging port (not used in customer systems).

Each serial port has a standard female 8-pin mini-din jack that should be labelled with its port number. The COM1 and COM5 ports can be attached to an Indy (etc.) using a normal 8-pin mini-din to 8-pin mini-din null modem cable. The COM2, COM3 and COM4 ports are attached to their respective hardware components with cables that should be provided with each system. They are, however, fairly standard 9-pin serial to 8-pin mini-din null-modem cables and in a pinch can be constructed by hooking together the appropriate off-the-shelf adapters and cables.

On a multi-rack system, the COM1 and COM4 ports are typically used on only one rack in the system (the one with the master IO6). The COM2 and COM3 ports are generally used on every MMSC.

The system console (tty1) on lego rack systems is attached to the IO6 *through* the MMSC. This allows the MMSC to intercept certain command strings, and also makes it possible for the MMSC to obtain data (such as new firmware images) from either a running unix system or the system console itself.

The tool used to flash MMSC firmware runs under IRIX and is called `flashmmsc`. It ships as part of IRIX 6.4, and can also be obtained from:

```
homegrown.engr:/usr/dist/mmsc/flashmmsc
```

A version that runs under IRIX 6.2 can be obtained from:

```
homegrown.engr:/usr/dist/mmsc/flashmmsc-6.2
```

The MMSC firmware image typically resides in a file named `mmscfw.bin`. The latest version can often be found in the same directory:

```
homegrown.engr:/usr/dist/mmsc/mmscfw.bin
```

Using the Serial Downloader

After certain catastrophic failures (or when an MMSC first arrives from the vendor) the non-volatile storage that usually holds the MMSC firmware is erased. When an MMSC is powered on in this condition, it will often show a blank blue screen, or perhaps some PC-style BIOS configuration gibberish. Under these circumstances, it is necessary to flash the firmware image into non-volatile storage using the "Serial Downloader", a simple tool provided in the MMSC's BIOS.

This procedure requires a direct serial connection from either a PC or an IRIX system to the COM5 port on the afflicted MMSC. The serial connection should be set up for 19200/8/N/1 (only!). If using a PC, it must have a terminal emulator capable of performing xmodem or xmodem-1K file transfers. If using an IRIX system, it should have the `flashmmsc` program if possible, although an xmodem file-transfer program can be used if necessary.

In addition, the MMSC *must* have a display attached to it. If the MMSC in question does not reside in the rack with the display panel, then the MMSC must be unbolted from the rack and moved closer to the display, or vice versa.

Starting the Serial Downloader

To start the serial downloader, it is necessary for the *MMSC*, as opposed to the rest of the system, to be powered up while at the same time holding down the bottom two buttons ("down" and "enter") of the MMSC display. This is unfortunately a fairly awkward procedure and usually requires two people, one to stand at the front of the system and hold down the bottom two buttons of the display, and the other to go to the back of the system and unplug (then reconnect) the MMSC from its power supply. There is actually a short delay before the buttons need to be pressed; if you are desparate or just gymnastically inclined, it may be possible to power cycle the MMSC then run around to the front panel in time to hold down the buttons.

The display buttons should be held down until the serial downloader menu appears. It will look approximately like this:

```
SBC-FFSC Serial Downloader 1.0

(U)pload an application
```

At this point, you can use either a terminal emulator or the `flashmmsc` program to transfer the firmware

image.

Transferring the Firmware Image Using a Terminal Emulator

If the COM5 port of the MMSC is attached to a PC (or any other system capable of doing an XMODEM file transfer) you will flash the firmware by simply uploading it to the MMSC. The menu from the MMSC display should also have been echoed to the PC. If it was not, try hitting ENTER once or twice, check your connections/baud rate/etc. and make sure you are using a null modem cable (as opposed to one wired straight through) to connect the MMSC and PC.

Once you have gotten the serial downloader menu from the MMSC, simply type `u`, then instruct your terminal emulator to send the firmware image file, typically `mmscfw.bin`. The file transfer protocol should be either XMODEM-CRC or XMODEM-1K. The transfer will take approximately 10 minutes. The display will show a series of dots to indicate progress; a typical firmware image may take 10-12 lines of dots to complete.

Transferring the Firmware Image Using `flashmmsc`

If the COM5 port of the MMSC is attached to a serial port on a system running IRIX, you can flash the image using the `flashmmsc` program. The format of the command is:

```
flashmmsc -d -l /dev/ttydXX -f mmscfw.bin
```

where `/dev/ttydXX` is the device name of the direct serial port you are using, and `mmscfw.bin` is the name of the file containing the firmware image you intend to load.

This program should be run once the display shows the serial downloader menu described above. It may take 10-20 seconds for the file transfer to begin, but once it does, a series of dots will be printed on both the MMSC display and the IRIX window. If these dots do not start after 30 seconds or so, check your connections as described earlier.

Upgrading MMSC Firmware From IRIX

If an MMSC is (more or less) operational and just needs to have its firmware upgraded to a new version, this can be accomplished from the IRIX system console using the `flashmmsc` command.

The flashmmsc command is assumed to be issued from a terminal that has access to one of the MMSC's associated with the system (such a terminal can access the MMSC itself by first typing the MMSC-escape character, typically CONTROL-T). The flashmmsc command is first issued using the `-m` option. Next, the MMSC **flash** command is issued to the desired MMSC by first typing the MMSC-escape character then the appropriate flash command.

For example, to flash a new firmware image onto rack 2 of a system that uses CONTROL-T for its MMSC-escape character, the sequence of events may look like this:

```
% flashmmsc -m
```

```
Ready to transfer new image to full-feature system
controller. To begin the transfer, type your MMSC
escape character (normally CTRL-T) followed by the
command:

    rack <rackid> flash

where <rackid> is the identifier for the system
controller you wish to upgrade.

[User types CONTROL-T]

 mmsc> rack 2 flash
Waiting for MMSC to initiate transfer...
```

This will take the firmware image from the default location `/usr/cpu/firmware/mmscfw.bin`. To specify a different image location, use the `-f` option:

```
flashmmsc -m -f my_special_firmware.bin
```

While this procedure is faster than using the serial downloader, it can still take 10 or more minutes to run, depending on the speed of the connection between the IO6 and the MMSC. As with the serial downloader, a series of dots will be printed to show progress.

Increasing the speed of the MMSC/IO6 connection

NOTE: This only applies to the speed of the MMSC/IO6 connection that is used when downloading an image from IRIX. There is no way to increase the speed of the connection used by the serial downloader.

The time it takes to download a new MMSC firmware image from IRIX can be improved by increasing the speed of the serial connection between the MMSC and the IO6. The maximum speed supported by the MMSC at this time is 57600 baud. Changing the serial line speed must be done in two steps:

1. First, tell IRIX to change the speed of the IO6 serial port. This is generally done with the `stty` command. For example, this would change the speed of the serial port for the console on which it was entered to 57600 baud:

   ```
   stty 57600
   ```

2. Next, tell the MMSC to change the speed of its serial port. This is done with the MMSC **com** command. To change the speed of the MMSC port used to talk to the IO6 to 57600 baud, you would first type the *MMSC-escape* character (usually Control-T), then enter the command:

   ```
   rack local com 4 speed 57600
   ```

It is important to do these two commands in order. If you change the MMSC serial port speed first, you won't be able to talk to IRIX in order to have it change its serial port speed. In that case, you would need to restore the MMSC serial port to its original speed (usually 9600 baud) and start over.

Once the speed of both serial ports has been changed, you should be able to issue commands to IRIX in the usual way. At that point, you can proceed with the `flashmmsc` command described above.

Note that under normal circumstances, the serial port speed of the MMSC will remain unchanged after a

reset or power cycle. However, the IRIX serial port will return to its original setting (usually 9600 baud) as soon as you log out. Therefore, it is usually wisest to change the speed of both serial ports back to their original settings after the downloading has been completed. This can be done as follows:

```
from IRIX:
stty 9600

type the MMSC-escape character, then:
rack local com 4 speed 9600
```

Rob Bradshaw
Last modified: Wed Oct 30 19:27:57 PST