M-Fit User Manual

Revision 2.6a

For use with M-Fit v2.6.x

April 5, 2017
Jonathan Grant

Revision History

Doc Rev	S/W Rev	Date	Notes	Updater
2.6	2.6.1	17-Mar-2017	Created open source version of M-Fit User Manual for software version 2.6.x.	J. Grant
2.6a	2.6.2	17-Mar-2017	Updated the powerdown test to note its unusual criteria for success and failure.	J. Grant

M-Fit User Manual

M-Fit v2.6.x
April 5, 2017

Introduction

The M-Fit suite is a set of tests originally designed for evaluating Siemens ATC 5.2b and ATC 6.x compliant engine boards. It may also be of use on other manufacturers' engine boards.

The suite is designed to be used in three different ways. First, it is a tool to permit bench testing and long term environmental testing of controllers during the manufacturing process. To support this, the M-Fit suite is easily scriptable with a consistent set of control options for iteration count, verbosity, and logging. In this way, groups of tests can be run in sequence or simultaneously, and the series of tests repeated as needed or indefinitely.

Second, it is a tool for configuration and diagnostic testing of controllers that have failed manufacturing tests or have been returned from the field for repair. In this situation the tests would typically be run by hand in verbose mode.

Finally, it is a tool to assist in validating compliance of the engine board against the ATC specifications.

Prerequisites

Traffic and other user applications must not be running. M-Fit needs exclusive access to the filesystem, SRAM, and mountable devices. The user must be logged in with root privileges to run most tests. This manual presumes that the user has some general familiarity with Linux command line operation.

Typographic Conventions

To describe the usage of the various tests, the following typographic conventions are be used.

- System output as well as command syntax will be in a `fixed-width typeface`.
- User input will be in **`fixed-width bold`**.
- Command names, file names, and device names will be in *italics* when not used as part of a command line example.

When describing command syntax:

- Terms in italics and enclosed in angle-brackets are <*descriptive*> of the value to be supplied.
- Terms in square brackets are `[optional]`.
- Ellipses `(...)` indicate that the preceding term may be repeated an arbitrary number of times.

Installation

The M-Fit suite may be installed in any convenient location. The recommended location is */opt/mfit*. These instructions presume the recommended installation.

Copy the installation file, *mfit.tgz*, to the controller, in this example to the */home/admin* directory. Log in to the controller with root permissions, change directory to */opt*, and un-tar the file with:

```
    tar xzpvf /home/admin/mfit.tgz
```

The files will be un-tar'd to the */opt/mfit* directory.

The M-Fit Suite

The heart of the M-Fit suite is a framework program called *fit*. This program is a multi-call binary which supports most of the M-Fit tests as applets. The applets are: *datakey*, *display*, *eeprom*, *ethernet*, *fio_monitor*, *filesystem*, *sd*, *memory*, *powerdown*, *rtc*, *serial*, *serial_echo*, *serial_port* and *usb*. The user would type **`fit memory`** to test the DRAM. For convenience, symbolic links have been created so that each of the tests within *fit* can be run with just the applet name alone.

The *fit* applets support a common set of command options provided at the framework level: failure-only logging (`-F`), general logging (`-L`), iteration (`-I` *<count>*) and verbose (`-V`). Some applets have additional test-specific options available. The applets require root privileges to run correctly unless otherwise noted.

End a Test Early

Use *<ctrl>*+**C** to interrupt a long test and have it end at the completion of the current iteration. The test ends gracefully and produces the expected log files. It is the only correct way to end a test that has been started with the `-I 0` (run forever) option. Depending on the test, it may still take a while for the test to complete.

Abort a Test

Use *<ctrl>*+\ to abort a long test immediately. This kills the test immediately and ungracefully. Log files may or may not have been generated and may be incomplete. Do not use *<ctrl>*+**Z** to attempt to abort a test, as this simply puts the test to sleep in the background. Use of *<ctrl>*+**Z** may cause subsequent tests to fail due to lack of resources.

Limitations

Certain tests can be run in ways that result in erroneous failures, that is failures which do not truly represent a hardware problem. Typically these are the result of limitations in the underlying device drivers.

It is known that on some engine boards, if a serial port has been tested in synchronous mode, the same port may not be correctly tested in asynchronous mode without an intervening reboot. Likewise going from asynchronous to synchronous requires a reboot. In other cases, the failure of a test may leave the port in an inconsistent and untestable state. Again, a reboot resets the port. These issues affect the *serial*, *serial_port* and *serial_echo* tests. The limitations do not represent an issue in manufacturing, ATC compliance or actual usage, as port modes are never switched on the fly.

Command Details and Usage

Fit Framework

The Fit framework is responsible for managing the common options on behalf of the applet being called. It provides command line processing and logging services to the underlying applets.

`fit`

`Usage: fit [<applet>] [<common-options>]... [<options>]...`

Run the specified applet.

The Common Options available to all applets are:

`-F`	enable Failure-only logging to file `fitFail.log`
`-L`	enable Logging to file `<applet-name>.log`, example: `memory.log`
`-I <count>`	run *<count>* Iterations (default = 1; continuous = 0)
`-V`	enable Verbose printout to console

The applets available are *datakey*, *display*, *eeprom*, *ethernet*, *filesystem*, *fio_monitor*, *memory*, *rtc*, *sd*, *serial*, *serial_echo*, *serial_port*, *powerdown* and *usb*.

Fit Applets

The applets are executed in the context of the Fit framework. The options for *fit* (upper case) apply to all of its applets as well and are thus called the "common options". The installation typically creates appropriate symbolic links so that only the applet name needs to be used.

`datakey`

`Usage: display [<common-options>]...`

Non-destructively test a Datakey. It saves a copy of the key's contents, erases the key, writes a known pattern and verifies the pattern. It then erases again, restores the original contents, and verifies them. This test exercises the Datakey and the SPI bus.

`display`

`Usage: display [<common-options>]... [<option>]`

Perform various display and keyboard tests. This test exercises the LCD display, keypad and SP6 (C60S).

By default, *display* sends a stream of text to the front panel in various formats for visual inspection. The eight user-defined characters are set to known patterns and added to the 96 printable ASCII characters to form a 104 character test set. Auto scrolling is enabled. First the eight user-defined characters are displayed with a pause for user examination. Then characters 1-40 of the test set are printed on the first line, followed characters 2-41 on the second, etc. In combination with the auto-scroll, this assures that every character is displayed in every location on the screen during the test cycle. The test continues by printing four lines of each with various combinations of reverse video and blink, followed by a pause for user examination. (Note that underline is not tested as it is not supported by most front panels.)

Optional tests are available as follows:

`-a`	Report the position of the Auxiliary switch.
`-d`	Run the display test described above (default if no options). This test takes about 1 minute to run on a 16 line display.
`-k`	Allow the user to test the keypad and Aux switch. Each key pressed or switch change is displayed on the screen for verification. Once 39 entries have been detected, the test ends.
`-s`	Show the display type (2070-3D, etc.) and number of lines.
`-t`	Test the tab stops on all lines in all positions. This test takes about six minutes to run on a 16 line display.

`-v`	Report the version of the display applet.
`-z`	Enable auto-scroll and auto-wrap (setup for further manual tests).
`-h`	Command help.

`eeprom`

`Usage: eeprom [<common-options>]... [<options>]...`

Run a high level, non-destructive test on the Host Board EEPROM. Also allows erasing, saving, programming and verifying the contents. This exercises the Host Board EEPROM and the SPI bus.

By default, *eeprom* saves a copy of the EEPROM's contents, erases the chip, writes a known pattern and verifies the pattern. It then erases again, restores the original contents, and verifies it.

Optional modes are available as follows:

`-d`	Program the EEPROM with a valid ATC Version 1 dataset. Once programmed, it should pass the `-v` validation test.
`-e`	Interactively edit the contents of the EEPROM field by field.
`-i`	Initialize (i.e., erase) the EEPROM.
`-l`	Load the EEPROM with the contents of a binary file; user is prompted for filename. This may be used to program the controller's finished configuration.
`-s`	Save the contents of the EEPROM to a binary file; user is prompted for filename if the image is valid (see `-v`).
`-p`	Print the ATC header information from the EEPROM.
`-v`	Verify the contents of the ATC header against the two CRCs included in the header.
`-h`	Command help.

`ethernet`

`Usage: ethernet [<common-options>]... [<options>]...`

Test both */dev/eth0* and */dev/eth1* using ping messages. This exercises both Ethernet channels.

The test requires external computers to be used as ping targets. In the course of the test, *eth0:5* and *eth1:5* are created. In the absence of a configuration file (`-c` option) these devices are configured based on the address of *eth0*. The second octet of the host IP address is incremented by 1 and 2, respectively, to give addresses that are on different networks from the main address and from each other. The controller then attempts to ping from both ports to another device located on the same network but with the last octet set to 254.

Example: If *eth0* has address 10.227.8.29, then *eth0:5* will be 10.228.8.29 and *eth1:5* will be 10.229.8.29. By default, *eth0:5* will attempt to ping a target at 10.228.8.254, and *eth1:5* at 10.229.8.254.

If a configuration file is specified, the addresses and ping targets are taken from the file, overriding the default behavior and the `-o` option. See Appendix A for information on Ethernet configuration files.

The following options are available:

`-c` *<configfile>*	Specify the optional configuration file. Overrides `-o` option.
`-o` *<last-octet>*	Set the last octet of the address of the ping targets (default = 254).
`-t` *<timeout>*	Set the ping reply timeout in seconds (default = 5).
`-h`	Command help.

`fio_monitor`

`Usage: fio_monitor [<common-options>]...`

Report the Field I/O type and read the raw input state. This test exercises SP5s. This is an experimental test which may be used to verify the presence of a Field I/O. FIO types 1-6 (ref. ATC 6.24 §7.2.1.7.13) are recognized.

filesystem

`Usage: filesystem [<common-options>]...`

Non-destructively test the onboard filesystems, excluding SD and USB. This exercises the NAND flash, the SRAM, the DRAM and the 60x Bus. The test generates a file, *fitFile*, with a known pattern of bytes and stores it on various filesystems. The file is read back, verified and deleted. The tested filesystems are: */home*, */opt*, */var*, and */tmp*. Also checked are */r0* or */media/sram* depending on board type.

memory

`Usage: memory [<common-options>]...`

The DRAM and SRAM is tested with various diagnostic patterns of bits. This exercises the RAM and the 60x Bus. The DRAM testing is done in stack space, initialized data (program global) space, and heap space. The heap space portion of the test takes about 30 seconds since all of the available space is tested. SRAM testing requires that */r0* or */media/sram*, depending on board, be mounted. The filesystem will be unmounted, the SRAM tested, and the filesystem remounted.

powerdown

`Usage: powerdown [<common-options>]...`

Test reception of the PowerDown signal. This test waits indefinitely for a power interruption. Interruptions of less than 500 ms will trigger this test to complete without resetting the controller. Note: this test logs success at power down and failure when power is restored.

rtc

`Usage: rtc [<common-options>]...`

Test the hardware real-time clock. This tests the RTC chip and the I^2C bus. The RTC's time registers are saved, tested with various patterns and values, then the original values restored.

sd

`Usage: sd [<common-options>]...`

Non-destructively test a mounted SDHC card. This exercises the SD card, SD and/or USB controllers and the 60x Bus. The test generates a file, *fitFile*, with a known pattern of bytes and stores it on the card. The file is read back, verified and deleted. This test is currently very hardware dependant and may not work on all models of engine board.

serial

`Usage: serial [<common-options>]... -c <configfile> [<options>]...`

Test the serial port(s) described in *<configfile>* with a comprehensive routine at all baud rates. This exercises the serial port hardware and any attached adapters and cables.

The defined ports are tested multiple times at all possible message sizes at each legal baud rate (a *test sweep*). The test sweep may be executed multiple times to create a single final test result.

The required flag for this command is:

`-c` *<configfile>*	Specify the configuration file. See Appendix B for file format.

Options for this command are:

`-a`	Use the official ATC 6.24 test string as the initial 49 bytes of the message.
`-b` *<baud-rate>*	Limit the test to only this baud rate.
`-f`	Test the flow control signals also. RTS, CTS and DCD are tested along with TXD and RXD. Used only with asynchronous ports. See Appendix C for wiring.
`-i` *<count>*	Run *<count>* iterations of the test sweep for a single test result (default 1).
`-m` *<behavior>*	Specify monitor loop behavior: * 0 – Wait and scroll: report intermediate progress at 10 sec intervals;

	1 – Wait and unblock;
	2 – Wait and block: wait until all tests are complete before reporting (default).
`-q`	Fail the test quickly, i.e., abort the test after the first failure.
`-r` *<rxflags>*	Override receiver open flags. This becomes the flags argument of open(). *
`-t` *<txflags>*	Override transmitter open flags. This becomes the flags argument of open(). *
`-x` *<rxtype>*	Choose receiver software design type. *
	0 – Receive packet in a single read;
	1 – Receive packet one byte at a time;
	2 – Receive packet from multi-byte reads (default);
	3 – Transmit only; reception is on different hardware.
`-y` *<multiplier>*	Increase the dynamic message timeout by this multiplier (default 1000).*
`-h`	Command help.

Note: * options not intended for general testing but to aid in hardware and device driver development.

serial_echo

```
Usage: serial_echo -c <configfile> [<common-options>]... [<options>]...
```

Test the serial port(s) described in *<configfile>* in a command and response simulation.

The defined ports are tested multiple times with a single message pattern at all baud rates. The test message is sent from the originator to the responder as defined in the configuration file. The responder then returns the message back to the originator. This test may be executed multiple times to create a single final test result.

Options for this command are:

`-c` *<configfile>*	Specify the configuration file (mandatory). See Appendix B for file format.
`-b` *<baud-rate>*	Limit the test to only this baud rate.
`-f`	Test with modem flow control signals also. See Appendix C for wiring.
`-i` *<count>*	Run *<count>* iterations of the test sweep for a single test result (default 1).
`-h`	Command help

The nature of NTCIP traffic and SDLC traffic to field devices is typically command and response. Thus the half-duplex nature of this test may provide a more real-world test.

serial_port

```
Usage: serial_port -c <configfile> [<common-options>]... [<options>]...
```

Test the serial port(s) described in *<configfile>* with a simple routine at a fixed baud rate.

The defined ports are tested multiple times with a single message pattern. Asynchronous ports are tested at 115,200 baud; synchronous ports are tested at 153,600 baud. This test may be executed multiple times to create a single final test result.

Options for this command are:

`-c` *<configfile>*	Specify the configuration file (mandatory). See Appendix B for file format.
`-i` *<count>*	Run *<count>* one-shot tests (iterations) and combine as a single test result.
`-h`	Command help

usb

```
Usage: usb [<common-options>]...
```

Non-destructively test a mounted USB memory stick. This exercises the USB stick, the USB controller, and the 60x Bus. The test generates a file, *fitFile*, with a known pattern of bytes and stores it on the stick. The file is read back, verified and deleted. This test is currently very hardware dependant and may not work on all models of engine board.

Appendix A – Ethernet Test Configuration Files

The Ethernet test configuration file must contain exactly six lines. Each line contains a single parameter. Comment lines are not permitted. The order and content for the test configuration lines are:

Line 1: *eth0:5* IP address
Line 2: *eth0:5* netmask
Line 3: *eth0:5* ping target address
Line 4: *eth1:5* IP address
Line 5: *eth1:5* netmask
Line 6: *eth1:5* ping target address

All addresses and netmasks are in IPv4 dotted decimal format.

Example 1 – this is equivalent to default (without `-c`) behavior for a controller with an *eth0* address of 10.227.7.1 and a Class B netmask:

```
10.228.7.1
255.255.0.0
10.228.7.254
10.229.7.1
255.255.0.0
10.229.7.254
```

Example 2 – *eth0* is on a Class A private network with a ping target of 10.10.10.100. *eth1* is on a Class C private network with a ping target of 192.168.1.2:

```
10.10.10.1
255.0.0.0
10.10.10.100
192.168.1.101
255.255.255.0
192.168.1.2
```

Appendix B – Serial Port Test Configuration Files

A serial port configuration file may contain one or more lines. Each line describes a test that will be run simultaneously with all the other lines in the file. Comment lines beginning with '#' are also allowed. The format for the test description lines is:

<receiver> <transmitter> <min-size> <max-size> <sweeps>

Where:

<receiver>	the receiving or responding serial port device name without the leading */dev/*, i.e., `sp1` means use asynchronous port */dev/sp1*; `sp5s` means use synchronous port */dev/sp5s*.
<transmitter>	the sending or originating serial port device name without the leading */dev/*.
<min-size>	the smallest test message to be used. This must be greater than zero and less than or equal to *<max-size>*. This value is ignored for *serial_port* and *serial_echo* tests.
<max-size>	the largest test message to be used. The typical range would be 1 to 200. Use 49 to exactly match the ATC test string. This size is always used for *serial_port* and *serial_echo* tests.
<sweeps>	is the number of passes through the base test to be executed and considered as a single test. This is independent of the iterations specified with `-i` or `-I` on the *serial* or *serial_port* command line.

Examples:

```
# This file causes an asynchronous loopback test for SP2 (/dev/sp2).
# It will use test messages of 1 to 5 bytes in length and repeat the
# underlying test twice before exiting with a PASS or FAIL
sp2 sp2 1 5 2
```

```
# This file causes a synchronous test between SP3s and SP8s
# (/dev/sp3s and /dev/sp8s). It tests traffic in both
# directions at the same time.
sp3s sp8s 1 5 10
sp8s sp3s 1 5 10
```

```
# This file runs several unrelated tests at the same time.
sp3s sp8s 1 5 4
sp8s sp3s 1 5 4
sp1 sp2 5 9 4
sp2 sp1 5 9 4
sp5s sp5s 1 5 4
```

```
# This file, along with the -a option, tests SP2 (/dev/sp2) strictly
# against the 49 byte ATC test string.
sp2 sp2 49 49 1000
```

Appendix C – Serial Port Test Cables and Loopback Plugs

Default operation of the *serial* or *serial_port* tests require only the data lines to be cross connected or looped back for asynchronous ports. Synchronous ports require the clock lines in addition. Including the flow control (`-f` option) testing for asynchronous ports requires that RTC be connected to CTS and DCD. Below are the wiring patterns. Note that for asynchronous port-to-port cables, shielding may be necessary for proper operation at baud rates above 38,400. Flow control is not available for ports SP4, SP5 and SP6.

Synchronous

PortA		*PortB*
TXD+	⟶	RXD+
TXD-	⟶	RXD-
TXC+	⟶	RXC+
TXC-	⟶	RXC-
RXD+	⟵	TXD+
RXD-	⟵	TXD-
RXC+	⟵	TXC+
RXC-	⟵	TXC-

Synchronous Loopback

PortA		*PortA*
TXD+	⟶	RXD+
TXD-	⟶	RXD-
TXC+	⟶	RXC+
TXC-	⟶	RXC-

Asynchronous

PortA		*PortB*
TXD	⟶	RXD
RXD	⟵	TXD
GND	——	GND
SHLD	——	

Asynchronous Loopback

PortA		*PortA*
TXD	⟶	RXD

Asynchronous Loopback w/ Flow Control

PortA		*PortA*
TXD	⟶	RXD
RTS	─┬⟶	DCD
CTS	⟵┘	

Asynchronous w/ Flow Control

PortA		*PortB*
TXD	⟶	RXD
RTS	─┬⟶	DCD
CTS	⟵┘	
RXD	⟵	TXD
DCD	⟵┬─	RTS
	└⟶	CTS
GND	——	GND
SHLD	——	

Appendix D – Quick Reference

fit

```
Usage: fit [<applet>] [<common-options>]... [<options>]...
```
Run the specified applet
```
Common Options:
   -F          enable failure-only logging to file fitFail.log
   -L          enable Logging to file <applet>.log
   -I <count>  run <count> Iterations (default = 1; continuous = 0)
   -V          enable Verbose printout to console
Applets:
   datakey, display, eeprom, ethernet, filesystem, fio_monitor, memory,
   rtc, sd, serial, serial_echo, serial_port, powerdown, usb
```

datakey

```
Usage: datakey [<common-options>]...
```
Non-destructively test a Datakey.

display

```
Usage: display [<common-options>]... [<options>]
```
Test the LCD display, the keypad, or other functions.
```
Options:
   -a    show Auxiliary switch position
   -d    exercise the LCD Display (default)
   -k    test Keyboard with user assistance
   -s    Show (report) display type
   -t    exercise the Tab stops
   -v    show Version of display applet
   -z    enable auto-scroll and auto-wrap
   -h    command Help
```

eeprom

```
Usage: eeprom [<common-options>]... [<options>]...
```
Non-destructively test or manipulate the Host Board EEPROM.
```
Options:
   -d    generate a Default dataset
   -e    Edit the dataset
   -i    Initialize (erase) the device to all ones
   -l    Load dataset from a file
   -s    Save dataset to a file
   -p    Print the header
   -v    Verify dataset against CRCs
   -h    command Help
```

ethernet

```
Usage: ethernet [<common-options>]... [<options>]...
```
Test both /dev/eth0 and /dev/eth1 using ping messages.
```
Options:
   -c <configfile>   Configuration filename, overrides -o
   -o <last-octet>   last Octet of ping address (default = 254)
   -t <timeout>      reply Timeout in seconds (default = 5)
   -h                command Help
```

filesystem

```
Usage: filesystem [<common-options>]...
```
Test the onboard flash, DRAM and SRAM filesystems by writing and reading a test file on each.

fio_monitor

```
Usage: fio_monitor [<common-options>]...
```
Report the Field I/O type and read the raw input state.

memory
`Usage: memory [<common-options>]...`
Test the DRAM with various patterns. Root access is not required to run this test.

powerdown
`Usage: powerdown [<common-options>]...`
Test reception of the PowerDown signal.

rtc
`Usage: rtc [<common-options>]...`
Test the hardware real-time clock.

sd
`Usage: sd [<common-options>]...`
Test a mounted SDHC card by writing and reading a test file on it.

serial
`Usage: serial -c <configfile> [<common-options>]... [<options>]...`
Test serial port(s) described in *<configfile>* with a comprehensive routine.

```
Options:
   -c <configfile>   configuration filename
   -b <baud-rate>    use this Baud rate only
   -f                include Flow control signals (async ports only)
   -i <count>        run <count> Iterations as a *single* test
   -l                use local Loopback
   -m <behavior>     specify Monitor loop behavior [0-2] (default 2)*
   -q                fail the test Quickly: abort after the first failure
   -r <rxflags>      override Receiver open flags*
   -t <txflags>      override Transmitter open flags*
   -x <rxtype>       choose receiver software design type [0-3] (default 3)*
   -y <multiplier>   dynamic message timeout by this multiplier (default 1000)*
   -h                command Help
```

serial_echo
`Usage: serial_echo -c <configfile> [<common-options>]... [<options>]...`
Test the serial port(s) described in *<configfile>* in a command and response simulation.

```
Options:
   -c <configfile>   configuration filename
   -b <baud-rate>    use this Baud rate only
   -f                include Flow control signals (async ports only)
   -i <count>        run <count> Iterations as a *single* test
   -h                command Help
```

serial_port
`Usage: serial_port -c <configfile> [<common-options>]... [<options>]...`
Test the serial port(s) as detailed in *<configfile>* with a very simple routine.

```
Options:
   -c <configfile>   configuration filename
   -i <count>        run <count> Iterations as a *single* test
   -h                command Help
```

usb
`Usage: usb [<common-options>]...`
Test a mounted USB drive by writing and reading a test file on it.

User Notes