*Flex*ROM II User's Manual

Version 1.0

Covers Models

FR2-1M,FR2-2M,FR2-4M,FR2-8M
FR2-220 , FR2-240 and FR2-280

Table of Contents

Introduction

Thank you for purchasing a TechTools product. We have made every attempt to provide quality tools at reasonable prices. If you have any questions, comments or suggestions, please feel free to contact us by FAX, Voice, email or mail and express your opinions.

How To Reach TechTools

You can reach us at any of the following :

Voice:	(972) 272-9392	Monday-Friday, 9:00-5:00 Central
FAX:	(972) 494-5814	
email:	support@tech-tools.com	
web:	http://www.tech-tools.com	
Mail:	TechTools	
	PO Box 462101	
	Garland, TX 75046-2101	

Up and Running in 15 Minutes!

As Engineers, we understand that you do not have time to troubleshoot your test equipment. You need to be able to trust your test equipment. With that in mind, we thoroughly test each and every *Flex*ROM II to insure that you receive a fully functional unit. In addition, we provide a self-test function to enable you to verify proper emulator operation at any time.

The following sections are designed to get you up and running in less than 15 minutes. If your system is not running by the time you complete these instructions, please call our technical support staff. We have many thousand emulators in use through-out the world and feel confident that we can resolve the problem over the telephone or through email.

Configure the emulator

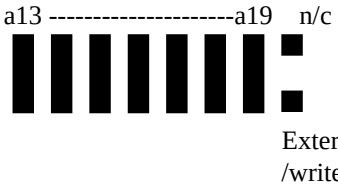
Set the SIZE CONFIGURATION jumpers to match the device size you are emulating. For example, if you will be emulating a 27C512 (64Kx8), set the jumpers for a 64Kx8 device. We printed the configuration information on the emulator's label for easy reference.

The configuration jumpers connect or isolate upper address lines to insure immunity to target wiring variances. If all jumpers are removed, we will ignore address lines A13 and up; emulating an 8Kx8 device. Adding the first jumper will configure us to emulate a 16Kx8 device. Adding the next jumper configures us for a 32Kx8 device and so forth, up to the maximum capacity of this emulator model.

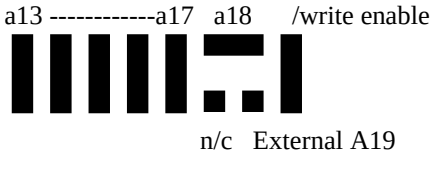
The following diagram identifies the jumper locations for 8 bit and 16 bit emulators:

8 bit Emulators

8 bit EPROM

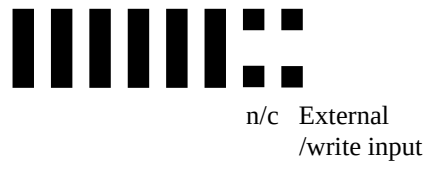


8 bit FLASH

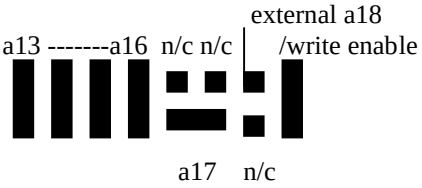


16 bit Emulators

16 bit EPROM



16 bit FLASH



The following table shows the proper settings:

Device Size	Example Device #s	A13 enable	A14 enable	A15 enable	A16 enable	A17 enable	A18 enable	A19 enable
8Kx8	2764	Off	Off	Off	Off	Off	Off	Off
16Kx8	27128	On	Off	Off	Off	Off	Off	Off
32Kx8	27256	On	On	Off	Off	Off	Off	Off
64Kx8 or 64Kx16	27512, 27210, 271024	On	On	On	Off	Off	Off	Off
128Kx8 or 128Kx16	27010, 27220, 272048	On	On	On	On	Off	Off	Off
256Kx8 or 256Kx16	27020, 27240, 274096	On	On	On	On	On	Off	Off
512Kx8	27040	On	On	On	On	On	On	Off
1024Kx8	27080	On	On	On	On	On	On	On

NOTE that this table shows the configuration information for all *Flex*ROM II models. Some of these configurations may not apply to your particular model, depending on its capacity and data width. For example, an FR2-1M-xx is a 128Kx8 emulator and therefore, can NOT be configured to emulate 16 bit or 256Kx8 and larger devices.

If the *Flex*ROM II is plugged into a FLASH socket and you wish to allow the target to write to the emulator, install the /write enable jumper. If *Flex*ROM II is plugged into an EPROM socket and you wish to allow the target to write to the emulator, connect your target's write signal to the external /write input pin.

NOTE: Some targets are not CAPABLE of writing to their EPROM or FLASH socket. This could be due to unidirectional buffers in the data path or chip select decodes that do not permit writes to this space.

Disable Any In-Circuit Programming Voltages

EPROMs and some FLASH devices are usually programmed with "super voltages" (> 5V) and are designed to handle such voltages on some pins; *Flex*ROM II is not! If your target is capable of in-circuit programming, disable or isolate the high voltage, not just the high voltage control signals. Even short surges during power cycling could be damaging.

*Flex*ROM II is a 5volt ONLY device. Any voltage greater than 5 Volts or less than 0 Volts will damage it.

Plug the emulator into your target

Turn the target power off. Plug the supplied target interface cable into the target's EPROM or FLASH socket. Be careful to align the PIN 1 end of the cable (designated with a stripe) with the PIN 1 end of the target's socket (usually designated with a notch, a dot or a '1').

PLUGGING THE CABLE IN BACKWARDS WILL DAMAGE *Flex*ROM II!

8 bit *Flex*ROM IIs ship with both 28pin and 32pin DIP cables. Use the 28pin cable if you are emulating a 28pin DIP device; the 32pin cable if emulating a 32pin DIP device. 16 bit *Flex*ROM IIs include a 40 pin cable for 16 bit DIP devices. If you are emulating PLCC type devices, you will need to purchase a DIP to PLCC adapter. These are available from TechTools as well as all popular adapter manufacturers.

Connect RESET

The RESET line must be connected to enable proper target operation. The reset signal is automatically asserted during each emulator access. This prevents the target from "running stupid" while we change its code space. The reset signal is released at the completion of the transfer, allowing the target to restart with the new code image.

*Flex*ROM II provides two reset outputs. RESET is an open emitter, ACTIVE HIGH output. /RESET is an open collector ACTIVE LOW output. Connect the appropriate output to your target's RESET circuitry. RESET is pin 16 on the feature connector and /RESET is on pin 15. Refer to appendix A or the emulator's label for the complete feature connector pin-out.

*Flex*ROM II's reset signal should be connected to the target's reset circuitry at the most "upstream" point available. This is usually an RC combination that senses the target's power up or a manual reset switch. Sometimes this circuitry is connected directly to the CPU reset input. In other targets, it will be connected to the input to a reset chip, power management chip, a watchdog timer chip or a Schmitt trigger gate.

DO NOT CONNECT *Flex*ROM II's RESET OUTPUT ACROSS A TOTEM-POLE OR BIPOLAR GATE, OR DIRECTLY TO GROUND OR 5VOLTS. THIS MAY RESULT IN DAMAGE TO *Flex*ROM II OR THE TARGET.

Call our technical support if you are unsure about which output to use or where to connect to your target.

Turn on the Target's Power

The emulator is powered from the target. The target power must always be on to access the emulator from either the target or the HOST side. When target power is applied, the emulator's POWER LED should glow steadily. This LED is powered DIRECTLY by the target's power.

> If the POWER LED does not glow, immediately turn off target power and correct the problem before reapplying power!

The only possible causes for the LED not to light are:

- target cable plugged in backwards or incompletely,
- insufficient target power,
- incorrect target cable adapters (wrong PLCC adapter, etc.),
- incorrect target socket wiring,
- bad target interface cable or
- a damaged emulator.

Connect the Download Cable

Select an unused **PRINTER** port on your PC. Remove any "dongles", security keys, printer switch boxes or extension cables from the port. Plug the supplied download cable directly into the selected printer port. Plug the other end of the cable into *Flex*ROM II's "IN" port.

> PLUGGING *Flex*ROM II INTO A SERIAL PORT WILL DAMAGE THE EMULATOR!

Connect Daisy-chain Cables

Multiple units are daisy-chained by connecting the "OUT" connector from one unit to the "IN" connector of the next unit. Additional units are added in the same manner. Use the supplied 12" daisy-chain cable to interconnect the units.

When daisy-chained together, the emulators automatically determine their position in the chain, making it unnecessary to configure their addresses. However, it is important that you understand how their addresses are determined.

The LAST EMULATOR in the chain (the one without a cable plugged into its "OUT" connector) is always emulator #1. The next emulator in the chain is #2 and so forth. The emulator closest to the PC will always be the highest numbered emulator.

Run self-test

Run FR2TEST.EXE. This file is located on the FRII Loaders Diskette. This self test will verify proper operation of the parallel port and all *Flex*ROM II functions except the target interface buffers.

If you are using multiple *Flex*ROM IIs, we recommend that you test them collectively to insure reliable operation in the final configuration.

Type "FR2TEST" to run a full speed test over printer port 0x378. You can change the printer port by using a /P parameter. For example, if your emulator is connected to a printer port at 0x3bc, you would enter "FR2TEST /P3BC". The most common printer port addresses (in order of popularity) are 378, 278 and 3bc.

If any of these tests fail, refer to the troubleshooting section for assistance. If the troubleshooting section does not resolve the problem, contact our technical support people for help.

Load your program

Start with a file that is known to work in this target. The binary image of a functioning EPROM is the best file to start with. Most EPROM programmers can read an EPROM and save its contents into a raw binary file. If binary is not an option, have the programmer store the file in Intel HEX or Motorola 'S' format.

FR2LD loads binary files into one or more *Flex*ROM IIs. If your image file is in a HEX format, use HEX2BIN to convert it to binary before running the ER2LD.exe program. Both HEX2BIN and ER2LD are batchable, allowing fully automated loading of binary and HEX files.

Run FR2LD without parameters to see a help screen on the command line parameters. Notice that all parameters are optional and have logical defaults. You only need to specify the parameters you would like to change from their defaults.

EXAMPLE LOAD PARAMETERS:

FR2LD file.bin	Loads FILE.BIN into emulator #1 at port 0x378
FR2LD file.bin /p3bc	Loads FILE.BIN into emulator #1 at port 0x3bc
FR2LD file.bin /e2	Loads FILE.BIN into 2 emulators, starting with #1.
FR2LD file.bin /e4 /f2	Loads FILE.BIN into four emulators, starting at #2. (2,3,4,5)
FR2LD file.bin /d0 /c	Loads FILE.BIN into emulator #1 at full speed with checksum verification enabled.
FR2LD file.bin /e2 /f2 /S	Loads FILE.BIN into emulators #2 and #3. The bytes within each 16 bit emulator are swapped so that the first byte in the file ends up connected to bits D8-D15 in a 16 bit emulator.

During the load and/or verify, the "SELECTED" LEDs of all involved emulators will light. At the completion of the transfer, all "SELECTED" LEDs will extinguish.

The *Flex*ROM II loader is very flexible, allowing any combination of 8 and 16 bit emulators to be loaded in almost any manner desired. The loader can load to individual units or automatically perform file splitting between 2 or more units. When the loader splits a file between multiple emulators, it will always place the **FIRST BYTE** of the file into the **LOWEST NUMBERED EMULATOR** involved in the transfer. Successive bytes are loaded into successive emulators. The following examples demonstrate this:

EXAMPLE 1

If you have two emulators daisy-chained together, and wish to split a file between them, you would specify: "FR2LD filename /e2 /f1". The /f1 is optional since the default is start with emulator #1. The /e2 parameter tells the loader that two emulators are involved in this transfer.

If these are 8 bit emulators, the loader will place the EVEN bytes (0,2,4..) of the file into emulator #1 (the LAST emulator in the chain). The ODD bytes (1,3,5..) will be placed into emulator #2. Note that you control which emulator holds the ODD bytes and which one holds the EVEN bytes by the order in which you daisy-chain the emulators.

If these are 16 bit emulators, the loader will place the first TWO bytes of the file into emulator #1, the next TWO bytes into emulator #2, the next TWO bytes into emulator #1 and so forth. The first byte of the file will be placed in the byte connected to D0-D7 in emulator#1. The second byte will connect to D8-D15 in emulator #1. If you would like to reverse this order (so that the first byte connects to D8-D15), specify a "/S" parameter.

EXAMPLE 2

If you have four 8 bit emulators daisy-chained together, a "/E4 " parameter would cause the loader to perform a 32 bit split between all four emulators. Bytes 0,4,8.. would be placed in the lowest numbered emulator (#1 by default). Bytes 1,5,9.. would land in the next higher numbered emulator. Bytes 2,6,10.. would be end up in the next emulator. Finally bytes 3,7,11… would be found in the highest numbered emulator involved in the transfer Again the byte ordering is determined by the emulator daisy-chain order.

EXAMPLE 3

If you are loading into a 16 bit emulator, The EVEN bytes will be placed in the byte connected to D0-D7 on a standard 16 bit EPROM. The ODD bytes will show up on pins D8-D15. If you would like to swap this order, you can specify a "/S" parameter on the command line. This is a global setting that affects ALL 16 bit emulators involved in the transfer.

If your system is not running at this point, check the troubleshooting section for suggestions. If that does not resolve the problem, PLEASE CONTACT OUR TECHNICAL SUPPORT at (972) 272-9392 or support@tech-tools.com. We are confident that we can solve the problem in a timely manner!

Enhancements

Increase Transfer Speed

Once basic functionality is established, you may choose to speed-up your downloads. The download speed defaults to its slowest setting to insure compatibility with all printer ports. The self-test program does a reliability test at a specified speed setting. Re-run the FR2TEST program with a faster speed setting (lower /Dx number). If the reliability test shows that the system is reliable at this speed, you will be able to use this same parameter on the ER2LD and ER2CAP programs.

Faster Downloads - Turn Off Verify

After you have used the emulator for a while and gained confidence that it is loading reliably, you can add a /V0 parameter to tell the loader to skip the verify operation. This will reduce your load times by over 75%! *Flex*ROM II is optimized for the fastest possible downloads. Therefore, the data stream FROM the HOST TO *Flex*ROM II is much faster than the reverse direction.

If you ever wish to verify the CURRENT contents of *Flex*ROM II's memory against a file, simple do a load with a "/V2" option (verify ONLY).

Error Detection and correction

If you add a "/C" parameter to the load command, The loader and *Flex*ROM II will generate and verify checksums on each packet. In addition, if an error is ever detected, the packet will be automatically re-sent. This adds about a 5% overhead to the transfers.

This is much more efficient than doing a complete verify of the SRAM contents against the load file and still provides a high degree of confidence that the download is valid.

Capture

*Flex*ROM II incorporates a new feature called Address Capture (or SNAP-SHOT). *Flex*ROM II hardware captures the value of the address lines during each valid target access. This data can be viewed with FR2EDIT, FR2CAP or custom programs utilizing FR2UTIL.LIB. The capture register can be interrogated without using any arbitration or worrying about interfering with the target.

While it does not maintain a history of every target cycle (like a logic analyzer or TRACE), it can provide considerable insight into your target's activities, once you understand exactly what it is giving to you. It is NOT a simple polled view of the current state of the address lines. It holds the address of the last valid access to the emulator from the target. A quick glance at the current capture value can tell you if the program has "left the road", or has gotten stuck in a tight loop somewhere. If you have a map file handy, you can even tell where it is stuck!

Have you ever forgotten to clear an interrupt flag and found yourself being endlessly interrupted? This would be easy to spot with the capture register. Of course if you see addresses in the 0xC000 range and you know your code stops at 0x8000, you instantly know that all is not well.

It would not take too much effort to use our library to build a profiler. Simply poll the capture register and gather statistics on the hit rates of each address in the EPROM/FLASH. If this were correlated back to your map file, you would have a non-intrusive source code profiler that did not require any special libraries or other modifications to the source.

Some customers are using this feature to help reverse-engineer the data table formats of automotive EPROMs, video game graphics, etc..

Trigger

FlexROM II incorporates an address match trigger circuit. This feature generates a pulse on the TRIGGER output (on the feature connector) each time the target accesses a specified memory location. You can specify the match value with FR2TRIG, FR2EDIT or through custom programs with FR2UTIL.LIB. In addition to pulsing the trigger pin, an internal flag is set to hold the event until *FlexROM* II's status register is polled. The status register can be polled through FR2CAP, FR2EDIT or a library function.

The TRIGGER output can be used to trigger a Logic Analyzer, SCOPE or logic probe. It could even be used to trigger some external hardware.

Even if you have a Logic Analyzer, you will find this feature very useful. It is always ready and available without having to connect all 20 address lines. Since it is always there (even when somebody "BORROWS" the logic analyzer), you will tend to make use of it.

When combined with a SCOPE, it provides a fully qualified Logic Analyzer type trigger on address matches to the SCOPE. When combined with a Logic Analyzer, it eliminates the need to connect all those signals to the address bus. This can increase the Logic Analyzer's trigger capabilities, free-up all those extra probes to look at other things and shorten your setup times.

The trigger circuit can even help you diagnose hardware problems during the first bring-up of a new board. Simple set a trigger at the initial jump vector. Reset the target and see what happens. If you never receive a trigger, then the processor is not running at all (check power, decodes, oscillators, etc. …) or the chip select decoding is wrong. If you get continuous triggers, it is being reset over and over. This could be due to bad op-code fetches (bad data lines..), run-away watch-dogs, faulty reset circuit, etc. If you get the one and only one trigger expected, move the trigger to MAIN() and see if you get there, and so on.

Another trick involves using simple test programs or even data patterns (NOPS, JUMP-HEREs, JUMP to 0s) to provide predictable stimulus. For example, if you filled the memory with NOPs, most processors will walk through memory. Since all memory cells contain the same data, we should get the same results, regardless of stuck address lines or other address problems. If the processor does NOT increment through memory, it is probably fetching something other than NOPs . This points us to a bad data path. If it does walk through memory, the data path is reasonably healthy so we look at the address path. We can set a trigger at each power of 2 (address 0,1,2,4,8,16….). until we find the bad address line. Remember that a 16 bit address bus would require at most 16 trigger settings!

Memory Retention

*Flex*ROM II incorporates a memory retention feature. This allows you to turn the target's power on and off without loosing the program loaded into the emulator's memory. This allows you to test power-up reset circuits and program behavior. *Flex*ROM II draws backup power for its SRAMs from the PC's printer port. The emulator senses target power and isolates its SRAM from the target power when the target is turned off. This permits safe target power cycling without the maintenance issues of batteries. The memory retention feature will work as long as the PC is powered up.

Arbitration

*Flex*ROM II provides hardware based arbitration support. This hardware works with your target's hardware to affect arbitration, allowing you to read and write the emulator's memory while the target is accessing that same memory. This may be useful for updating data tables, strings or constants on-the-fly. The full-screen editor will even allow you to look into memory to see if the target left any messages there for you.

*Flex*ROM II supports several different arbitration methods.

Request/Grant

In Request/Grant arbitration, *Flex*ROM II will assert a /REQUEST before each access to the memory space. The *Flex*ROM II hardware arbiter will then wait until the target asserts a /GRANT signal before generating the memory cycle. At the completion of the access, *Flex*ROM II will release the /REQUEST signal and return control of the memory space back to the target.

*Flex*ROM II does not actually access the target's bus, but requesting access (and waiting for a grant) insures that the target will not attempt to interrupt our access to our memory.

These arbitration signals (/REQUEST and /GRANT) could be connected to your target's BUS request/grant, DMA request/grant or similar control lines. Of course many micro-controllers do not support any type of bus mastering or DMA operations.

Note that this is hardware based arbitration and very efficient. *Flex*ROM II will release /REQUEST within 850ns after /GRANT is asserted.

Ready

In Ready Arbitration, *Flex*ROM II inserts wait states into target cycles to affect arbitration. NOTE that this is radically different than simply slowing down your target with wait-states. *Flex*ROM II will insert wait-states ONLY DURING COLLISIONS. Your target runs at full speed except for during the brief moment in which the target attempts to access *Flex*ROM II's memory while it is being accessed by the *Flex*ROM II itself.

This is a hardware based arbitration method. The HOST posts a request to read or write to the emulator's memory space. *Flex*ROM II hardware watches the target's accesses and waits for the end of a target access before starting its own access. This minimizes the chances of a collision (minimizing the impact of arbitration). Once *Flex*ROM II sees a target access complete, it isolates its SRAM from the target and starts its own access. If the target starts an access before the *Flex*ROM II has completed its access, *Flex*ROM II asserts the /READY signal. This signal is held active until *Flex*ROM II has completed its access, re-enabled the target side buffers and the memory has had time to complete the current target access. At this point, /READY is released and the HOST is notified that the access completed.

The /READY line should be connected to your target's wait-state circuitry.

Cycle Interleaving

This technique depends on the emulator's internal CYCLE time plus its target ACCESS time being less than the target's CYCLE time. Since *Flex*ROM II avoids interrupting target accesses, and uses hardware based arbitration, it is possible to use CYCLE Interleaving if your target's memory CYCLE time is greater than 450ns. Many micro-controllers and some microprocessors meet this criteria.

To use this method, configure *Flex*ROM II for READY mode arbitration, but leave the /READY line disconnected.

"SLAM IT"

This is not true arbitration, but is worth mentioning. In some cases, arbitration is not really required, as long as the emulator does not tie up the bus too long. A good example of this is look-up data tables.

We have several customers using this technique to do on-the-fly modifications to Engine Controller ROMs, wave-form synthesizer ROMs, graphics ROMs, character generator ROMs

If your application uses the EPROM or FLASH device to hold data rather than code, and it can tolerate an occasional bad fetch (wrong data), it may be acceptable to simply allow *Flex*ROM II to collide with the target during the access. Note that *Flex*ROM II will not hold the target's bus or corrupt anything on the target's bus. It simply isolates itself from the target's address and data bus during its access, causing the target to fetch garbage during the collision. Each *Flex*ROM II access will isolate it from the target for approximately 300ns.

You have two options when using this method. You can configure *Flex*ROM II for CYCLE Interleaving (ready mode, but do not connect the ready line). This has the advantage of minimizing the probability of a collision since *Flex*ROM II will wait until it sees a valid target cycle end before starting its cycle. Also we will only collide if the target cycle time is less than 450ns. The down-side is that if the target stops accessing the emulator's memory space, arbitration will time out. The other option is to configure *Flex*ROM II for REQUEST/GRANT arbitration and then short the REQUEST line to the GRANT line; insuring that *Flex*ROM II always receives an immediate grant. This method will always allow *Flex*ROM II access to the memory, but will blindly disrupt any current target access.

RESET

Of course this is not arbitration, but it is always worth asking "Do I really need to arbitrate at all?". If the changes that you will make to the EPROM/FLASH will require re-starting the target to take affect(like code changes), arbitration is not required (or desired). Simply let *Flex*ROM II assert reset during the modifications. *Flex*ROM II will then be guaranteed access to the emulation memory and the target will be automatically re-started after the changes.

Control Lines

*Flex*ROM II provides 4 user control lines. These are TTL outputs on the feature connector that can be individually set high or low. You can control these with library functions or from within FR2EDIT.

Status Lines

*Flex*ROM II provides 4 user input pins. These pins are sampled at the beginning of each valid target access to the emulator. In addition, whenever a target access generates a trigger, the value of the status pins at that instant are held until the status register is read. The status register can be read through a library function, FR2CAP or FR2EDIT.

Libraries

*Flex*ROM II ships with libraries to provide complete control over all of *Flex*ROM II's features to your custom programs. Complete documentation for the library functions is provided on the distribution disk.

Software

*Flex*ROM II ships with several programs. The command line driven programs document their valid parameters if you invoke them with a '?' for a parameter. For example, to se the valid options for the capture program, type "FR2CAP ?". All programs except FR2EDIT are batchable command-line style programs. The following are brief descriptions of the functions of each program.

FR2LD

FR2LD loads binary files into one or more emulators. It can also perform a verify ONLY if you suspect that the emulator contents have been corrupted. FR2LD fully supports all arbitration methods.

FR2TEST

FR2TEST verifies the reliability of the printer port and system wide communications with the emulators. It also performs a very complete self-test of all attached emulators. This program tests all emulator functions except the actual target interface buffers and target interface cable.

FR2EDIT

FR2EDIT is a full-screen editor/loader. It loads BINARY and HEX files into one or more emulators. FR2EDIT fully supports all arbitration modes during reads/writes and editing. It also provides a setup dialog to set the trigger value and a live status window. The live status window allows you to view the current state of the status pins, control pins, capture flag, trigger flag and the value of the capture register, without impacting the target.

FR2TRIG

FR2TRIG allows you to set the trigger value in any specific emulator.

FR2CAP

FR2CAP continuously displays the value of the status pins, capture flag, trigger flag, and capture register from any specified emulator. It can display these on a single line or can be instructed to scroll the samples up the screen so that you can see the last couple of dozen samples. Pressing the space bar will toggle a freeze of the display. Pressing any other key will exit the program.

FR2RESET

FR2RESET asserts, releases or pulses the target reset pins on all attached emulators simultaneously.

HEX2BIN

HEX2BIN converts all Intel HEX, Motorola 'S' and Tektronix HEX files to BINARY. It supports 8 formats in all.

Troubleshooting

The distribution diskette contains a test program called FR2TEST.EXE. FR2TEST verifies the port address, its reliable at the selected transfer speed and the emulator's functionality. The first step in troubleshooting is to run this program and make special note of any problems it reports.

Regardless of the symptom, verify the following:

1. There are no "dongles", security keys, switch boxes or other devices between the *Flex*ROM II download cable and the printer port.
2. Try each available printer port configuration Some ports work best in STANDARD, AT, NIBBLE or COMPATIBLE mode. Others may provide more reliable high-speed operation when set to ECP, EPP, PS2, Bi-directional or HIGH-SPEED mode.
3. The target is powered up.
4. All download, daisy-chain and target interface cables are tight and fully inserted.
5. The selected printer port is not being serviced by a print spooler or some type of TSR.
6. If the PC is a laptop or a "GREEN" PC, verify that the printer port is not disabled, powered down or configured for any type of power savings mode.

In all cases, try "booting" clean from a DOS diskette to see if the symptoms disappear. If the loader works under this condition, chances are good that some other program is interfering with our access to the printer port.

Printer Port Not Found Message

The loader could not find a printer port at the specified address. Try another address with the /Pxxx parameter. The most common addresses are 378, 278 and 3BC. Also verify that the attached emulators are powered up.

No Emulator(s) Found

Specified an existing but incorrect printer port.

This error message indicates that the specified port was found, but no emulators were found at that port. The emulators might be plugged into a different port. Change the command line parameter /Pxxx to specify the correct port, or move the download cable to the specified port. The most common printer port addresses are 378, 278 and 3BC.

Transfer speed set too high for this port.

The transfer reliability is verified by the FR2TEST program. Test the desired transfer speed reliability with "FR2TEST /dx" (where x is 0,1,2,3 or 4). /d0 is the fastest setting.

If you are daisy-chaining several emulators, their combined load can slow down the printer port rise times, resulting in a need to slow down the transfer speed to avoid over-running the port signals. Run the FR2TEST program with the emulators daisy-chained together to properly reproduce the eventual working environment.

Target is not powered up.

*Flex*ROM II draws its operating power from the target's EPROM socket. The target must be powered up during all load or verify operations. If multiple emulators are daisy-chained, they must ALL be powered up to load to ANY of them.

Target power is noisy or Voltage too low.

*Flex*ROM II draws more current than the EPROM it is replacing. If the target is a very low power system, or is already pushing its power supply limits, it may not have enough current to spare to power the emulator, causing power fluctuations or noise. This is a rare condition, but worth mentioning

Checksum Errors

Checksum errors indicate a communications problem. This may be due to bad download cable connections, a slow rise-time printer port, noise on the download cable, noise on the target power, etc. You may try a slower download speed (increase the /d parameter). Also verify that the download cable is not laying across a power transformer, your monitor, an arc-welder, etc.

Arbitration Time-outs

Arbitration time-outs are reported when *Flex*ROM II was unable to complete a memory access within 65 us. This can only happen if arbitration has been enabled. If you did NOT intend to do arbitrated accesses, remove the "/A" parameter from the loader command line or turn on the "RESET during DOWNLOADS" option in FR2EDIT. If you truly meant to do arbitrated accesses to the memory space while the target is using that same space, verify the following:

If you are using READY arbitration, the emulator MUST see target activity to properly arbitrate with it. If the target is not accessing the memory space emulated by this emulator, it will not see activity and therefore will not be able to arbitrate. The emulator must see ACTIVITY (both start of accesses and end of accesses). Also note the /READY line uses an open collector driver and does NOT provide a pull-up resistor. If the target does not provide a pull-up at the connection point, you will need to add a 1K - 10K resistor between the connection point and +5V.

If you are using REQUEST/GRANT arbitration, verify that the target arbiter is working, both /RESET and /GRANT are properly connected and that the target arbiter expects an active LOW /REQUEST and generates an active LOW /GRANT. Also note that /REQUEST is an open collector driver to allow multiple emulators arbitrate in parallel. If your target does not provide a pull-up resistor at the attachment point, you may need to add one (around 10K should do).

If you are using CYCLE INTERLEAVING arbitration, verify that you set the arbitration option to READY mode and left the ready line disconnected. Also verify that the target cycle time is at least 450ns. This is measured from the END of one cycle (CS, OE or WRITE going INACTIVE) to data stable requirement in the next cycle.

Fails Verify

File too large

The file may be too large for the emulator, causing the data to wrap-around or to be truncated. For example, an FR2-1M can hold up to 1Mbit of data (128Kbytes). Any file larger than 131,072 bytes would overflow an FR2-1M. Look at the BINARY file size and verify that it is less than or equal to the emulator's maximum size. If you are working with HEX files, note that the size of the HEX file is irrelevant. Look at the size of the BINARY file created by HEX2BIN.

Target Power noisy or Voltage too low.

Noise problems are very dynamic. It is possible that we could identify the emulators but be unable load and verify properly. *Flex*ROM II draws more current than the EPROM it is replacing. If the target is a very low power system, or is already pushing its power supply limits, it may not have enough current to spare to power the emulator, causing power fluctuations or noise.

Transfer speed set too high for this port.

The transfer reliability is verified by the FR2TEST program. Test the desired transfer speed reliability with "FR2TEST /dx" (where x is 0,1,2,3 or 4). /d0 is the fastest setting.

If you are daisy-chaining several emulators, their combined load can slow down the printer port rise times, resulting in a need to slow down the transfer speed to avoid over-running the port signals. Run the FR2TEST program with emulators daisy-chain together to properly reproduce the eventual working environment.

A print spooler or TSR is interfering with our port accesses

Try "booting" clean from a DOS diskette to see if things stabilize. If this corrects the problem, suspect that some other program(like a print spooler) is interfering with our access to the port.

Printer port set to an incompatible mode

If your printer port is configurable through BIOS settings. Try each configuration. Some ports will perform better in STANDARD, AT, NIBBLE or COMPATIBLE mode. Others will allow higher transfer speeds when configured for ECP, EPP, PS2, HIGH-SPEED or similar settings.

Verifies, but target does not run

Target not being reset

If the wrong reset output is being used or it is connected to the wrong spot on the target, it may not be resetting the target during the download. We would get a good download, but the target was fetching garbage during the transfer. Some processors will HALT, others will simply execute the random garbage they received. In either event, the processor will not "know" to start over and execute the new code.

Try removing the reset line, turning off the target power for 10 seconds and then turning the power back on. If the target now comes up and runs, the reset signal connection was wrong. While you could do this after each load, we highly recommend correcting the reset problem to make resets automatic and to prevent the target from running garbage during the download.

Emulator device size set wrong.

The device SIZE configuration affects only the TARGET'S view of the emulator. The file will download and verify correctly as long as the emulator is large enough to hold the file. If the size is set too small, the target will not be able to access all of the file image. For example, if the file is 60,000 bytes long, it would load into an FR2-1M and verify properly, regardless of the jumper settings. However if the emulator is configured to emulate a 27256, the target would see only the first 32Kbytes mirrored into both halves of the 64KByte space. This is exactly what would happen if you burned the first 32Kbytes into a 27256 and plugged it into the same socket.

File Too Large

If the file size exceeds the configured device size, the target will not be able to access the entire file. Note that the file may still verify correctly, as long as it does not exceed the emulator's maximum capacity. If the emulator is configured correctly and the file is larger than the device setting, your code has just outgrown its space. If you are emulating the maximum size device the target is wired to accept, you will have to reduce the code size. If the target has some unused space contiguous to the current EPROM space, you may be able to modify the target board to accept a larger EPROM. This would typically involve adding another address line to the socket and possibly modifying the EPROM chip select decodes. Of course you would need to re-configure the emulator for the larger device as well (if it has room to spare).

Wrong PLCC or other adapters used.

Any adapter used between the target cable and the target socket itself has the potential of introducing errors. In particular, PLCC footprints vary by EPROM size, model and manufacturer. In general, small EPROMs (<1Mbit) generally have one pin out and larger EPROMs have a different pin out. Most FLASH chips are pinned out like large EPROMs. However, SOME small FLASH are wired like SMALL EPROMs. In general, large devices (>=1Mbit) use a 32 pin DIP to 32 pin PLCC adapter. Smaller devices usually need a 28 pin DIP to 32 pin PLCC adapter but MIGHT need the 32 pin version. Give us a call if you are not sure.

File is not a raw binary image of the EPROM

The command line loader loads only unformatted, raw binary images. If the file is a formatted HEX file, use HEX2BIN.EXE to convert it first. If you are not sure about the file's format, "TYPE" it to the screen. If you see random printable, non-printable and graphics characters, it is BINARY. If you see well formatted HEX characters, it is a HEX file. Intel HEX files start each line with a ":". Motorola 'S' files

start each line with an “S”. Tektronix HEX files start with a “%” or a “/”. HEX2BIN automatically recognizes and converts 8 variations of these HEX formats.

Remember, your Resume’ will load and verify, but most likely will not execute properly!

Failed to use a /E parameter to split the file

If you want the loader to spilt the file between multiple emulators, you need to specify a /Ex parameter to tell the loader how many emulators to split the file between.

Failed to use a /8 parameter.

If you are using a 16 bit emulator to emulate a single 8 bit device (with the ADP2x8 adapter), you need to tell the loader with a /8 parameter. If you fail to specify the /8 parameter, the loader will load the file into both the upper and lower bytes of the emulator (16 bits wide). Of course this will verify OK, but the target will only get ½ of the data .

File loaded into the wrong emulator(s)

If you have 2 or more emulators daisy-chained, you may have loaded the file into the wrong unit(s). Watch the “SELECTED” LEDs to see which units where actually selected for the transfer. The loaders default to emulator #1. To select a different emulator, specify a /Fx parameter. This tells the load the FIRST emulator involved in the transfer.

Incorrect Byte Order

If you are loading into a 16 bit emulator, remember that the EVEN bytes will load into data bits D0-D7 and the ODD bytes will end up in data bits D8-D15. If your target is wired reverse to this, you will need to specify a “/S” parameter to tell the loader to swap the bytes.

If you are loading into multiple emulators with automatic splitting, remember that the loader will load the **FIRST byte in the file into the lowest numbered emulator** involved in the transfer. If the emulators are plugged into the wrong target sockets, you can move them to the correct sockets or you can change the order in which the emulators are daisy-chained.

Target Power noisy or Voltage too low.

Noise problems are very dynamic. It is possible that we could load and verify OK, but as soon as we release reset and let the target start running, system noise increases and trashes the system. Low system voltage could behave in a similar manner. It is possible that we could load and verify with a target voltage that is well under spec, but the target will not execute properly. *Flex*ROM II draws more current than the EPROM it is replacing. If the target is a very low power system, or is already pushing its power supply limits, it may not have enough current to spare to power the emulator, causing power fluctuations or noise.

Emulator too slow

*Flex*ROM IIs are rated at 45ns or 90ns. The last two digits of the part number specify the emulator's speed. The emulator must be at least as fast as the access times required by the target (smaller numbers are faster). If the target's access requirements are not documented, you can get a general idea of its requirements by looking at the EPROM's rating. The EPROM's speed is not an absolute indicator because many EPROMs run faster than they are marked; particularly at full voltage and room temperature. We have seen many systems that appear to run fine, even though their access times routinely violate the EPROM's rating. Fortunately, *Flex*ROM IIs generally run faster than rated as well!

Bad Code

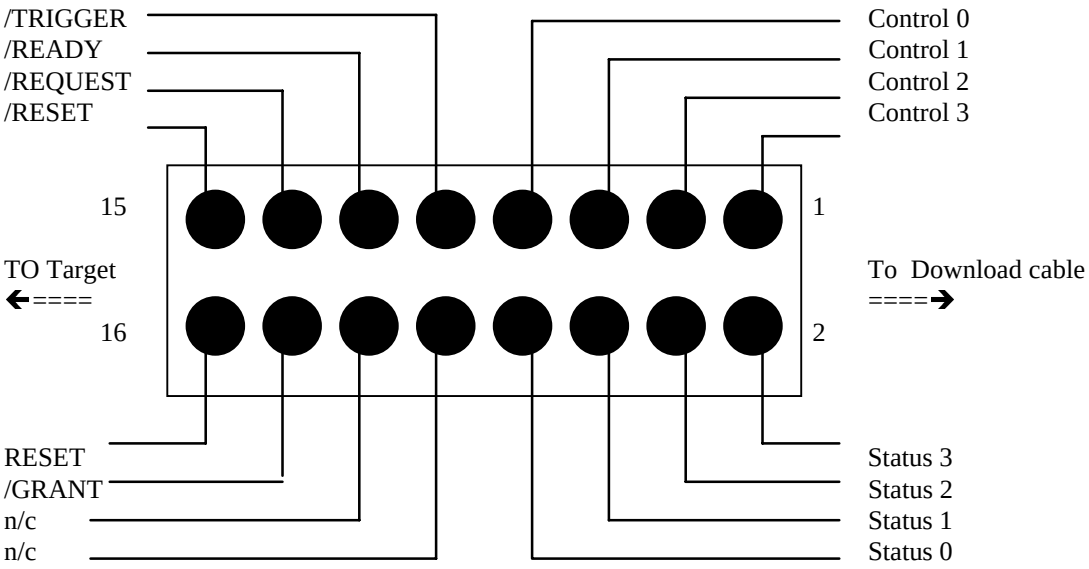
Remember, the one variable that is always changing in this set up is the code itself! If you keep a baseline code image on hand that is known to work, you could always load that image into the emulator as a sort of sanity check.

If these tips did not solve the problem you are experiencing, please contact our technical support people at

Voice: (972) 272-9392 FAX: (972) 494-5815
email: support@tech-tools.com
WEB: http://www.tech-tools.com

Appendix A - Feature Connector

The feature connector gives you access to *Flex*ROM II's reset outputs, arbitration, status and control pins. The following diagram documents the location of these pins:



Signal	Direction	Notes
Reset	OUT	Open Emitter with 10K pull-down
/Reset	OUT	Open Collector with 10K pull-up
/Request	OUT	Open Collector (no pull-up provided)
/Grant	IN	
/Ready	OUT	Open Collector (no pull-up provided)
/Trigger	OUT	
Control 0-3	OUT	
Status 0-3	IN	Sampled on access, Latch on trigger

