User Manual

Version 3.0

PromICE™

Grammar Engine, Inc.
921 Eastwind Drive Suite 122 Westerville OH 43081
(614) 899-7878

PromICE User Manual

Version 3.0

Contents

1. WARRANTY AND SOFTWARE LICENSE

1.1. WARRANTY

GEI offers a 30-day money back guarantee on its products. If not fully satisfied within 30 days of the receipt of the product, it can be returned for a refund. Shipping and handling fee are non-refundable. All returned merchandise must be complete and returned in the original packing, with RMA number, in working order.

GEI products are covered by a one year warranty. GEI warrants that the equipment is free of manufacturing defects (i.e. defects in material and workmanship under normal and proper use in their unmodified condition). The warranty is for the period of one year from the date of delivery to the customer. GEI shall repair or replace any defective equipment during this period at its option. This warranty does not cover any damage resulting from accident, abuse, or any consequential damages as a result of the use of this product.

WARNING: Do NOT open the PromICE. All warranties are void if the unit is opened!

1.1.1. REPAIR/REPLACEMENT

GEI shall perform all repairs under warranty and re-ship the product via ground shipping at no charge. Special handling charges apply. For all repairs falling outside the parameters of the warranty, a minimum bench charge of $50 will be assessed. To return a product for repair or replacement, GEI shall provide a return material authorization number (RMA#) which is to be clearly marked on the package. A copy of the invoice or packing slip must accompany the return.

NOTE: Any static sensitive device returned to GEI must be shipped in a static shielding bag. *The warranty is VOID if the product is not returned in a static safe container.*

1.2. SOFTWARE LICENSE AGREEMENT

Please read this license carefully before using the software. By using the software, you are agreeing to be bound by the terms of this license. If you do not agree to the terms of this license, promptly return the unused software to the place where you obtained it and your money will be refunded.

1.2.1. License.

The application, demonstration, system, and other software accompanying this License, whether on disk, in read only memory, or on any other media (the "Software") and related documentation are licensed to you by Grammar Engine, Inc. ("GEI"). You own the disk on which the Software is recorded, but GEI and/or GEI's Licenser(s) retain title to the Software and related documentation. This License allows you to:

(a) With respect to the Software, copy the Software either over a network or onto diskettes for use on computers provided all such computers are within your Site. "Site" is generally defined as the following: (i) Higher Education: All students, faculty, and administration at the university/college campus; (ii) Business: All of a division's locations within a 25-mile radius; (iii) Government: All of an agency's/department's locations within a 25-mile radius.

(b) Make one copy of the Software in machine-readable form for backup purposes.

(c) Perform the above provided you reproduce on such copies all copyright notices and any other proprietary legends that were on the original copy of the Software.

(d) Transfer all your license rights in the Software and the related documentation to another party provided the other party reads and agrees to accept the terms and conditions of this License.

1.2.2. Restrictions

The software contains copyrighted material, trade secrets, and other proprietary information, and, in order to protect them, you may not modify, network, rent, lease, loan, distribute, or create derivative works based upon the Software in whole or in part for purposes other than for use with GEI products.

1.2.3. Termination

This license is effective until terminated. You may terminate this License at any time by destroying the Software and related documentation and all copies thereof. This License will terminate immediately without notice from GEI if you fail to comply with any provision of this License. Upon termination, you must destroy the Software and related documentation and all copies thereof.

1.2.4. Export Law Assurances

You agree and certify that neither the Software nor any other technical data received from GEI, nor direct product thereof, will be exported outside the United States except as authorized and as permitted by the laws and regulations of the United States.

1.2.5. Limited Warranty on Media

GEI warrants the disks on which the Software is recorded to be free from defects in materials and workmanship under normal use for a period of ninety (90) days from the date of purchase as evidenced by a copy of the receipt. GEI's entire liability and your exclusive remedy will be replacement of the disk not meeting GEI's limited warranty and which is returned to GEI or a GEI authorized representative with a copy of the receipt. GEI will have no responsibility to replace a disk damaged by accident, abuse, or misapplication. Any implied warranties on the disks, including the implied warranties of merchantability and fitness for a particular purpose, are limited in duration to ninety (90) days from the date of delivery. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

1.2.6. Disclaimer of Warranty on Software

You expressly acknowledge and agree that use of the Software is at your sole risk. The Software and related documentation are provided "as is" and without warranty of any kind, and GEI expressly disclaims all warranties, express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. GEI does not warrant that the functions contained in the GEI Software will meet your requirements, or that the operation of the GEI Software will be uninterrupted or error-free, or that defects in the GEI Software will be corrected. Furthermore, GEI does not warrant or make any representations regarding the use of the GEI Software or related documentation in terms of their correctness, accuracy, reliability, or otherwise. No oral or written information or advice given by GEI or a GEI authorized representative shall create a warranty or in any way increase the scope of this warranty. Should the GEI Software prove defective, you (and not GEI or a GEI authorized representative) assume the entire cost of all necessary servicing, repair, or correction. Some states do not allow the exclusion of implied warranties, so the above exclusion may not apply to you.

1.2.7. Limitation of Liability.

Under no circumstances, including negligence, shall GEI be liable for any incidental, special, or consequential damages that result from the use or inability to use the GEI Software or related documentation, even if GEI or a GEI authorized representative has been advised of the possibility of such damages. Some states do not allow the limitation or exclusion of liability for incidental or consequential damages so the above limitation or exclusion may not apply to you. In no event shall GEI's total liability to you for all damages, losses, and causes of action (whether in contract, tort [including negligence] or otherwise) exceed the amount paid by you for the Software.

1.2.8. Controlling Law and Severability

This License shall be governed by and construed in accordance with the laws of the United States and the State of Ohio, as applied to agreements entered into and to be performed entirely within Ohio between Ohio residents. If for any reason a court of competent jurisdiction finds any provision of this License, or portion thereof, to be unenforceable, that provision of the License shall be enforced to the maximum extent permissible so as to effect the intent of the parties, and the remainder of this License shall continue in full force and effect.

1.2.9. Complete Agreement

This License constitutes the entire agreement between the parties with respect to the use of the Software and related documentation, and supersedes all prior or contemporaneous understandings or agreements, written or oral, regarding such subject matter. No amendment to or modification of this License will be binding unless in writing and signed by a duly authorized representative of GEI.

2. INSTALLATION

2.1. Unpacking

Your PromICE most likely traveled by air to arrive at your place. While rushing through the air at several hundred miles an hour it would have picked up as much as 50,000 volts of static charge. First time you take your unit out of the static shielding bag and you are in an unsafe static environment you could kill it. Replacing blown input buffers on the ROM interface is the most common repair we do. You can develop a long and productive relationship with your PromICE by following a few safety guidelines and some handling precautions:

1. Take the entire box, the PromICE came in, to the work site. If the box must be opened elsewhere for inspection under no circumstance should you or anyone else remove the PromICE unit from its static shielding bag.

2. Make sure that static safe environment is maintained at the work site at all times. Everytime you approach the work space ground yourself by touching a grounding strip.

3. Wear a grounded wrist strap if you expect to handle the PromICE or the target system much, such as when setting up. At other times if you are sensitive to static issues, you can practice safe static handling without the need for the wrist strap.

4. If you must move the PromICE, wear a grounded wrist strap, disconnect all cables from the PromICE. Return the PromICE to its static shielding bag before transporting. Never ever carry the PromICE with the ROM cables still attached. You are guaranteed to blow the buffers very easily and also you can easily crush the ROM cable DIP header pins.

5. When handling the PromICE casually in use, make conscious effort not to touch any exposed pins. Most of the pins are direct connections to inputs on buffers and are very sensitive to static damage. A damaged buffer can cause intermittent and permanent failures.

Remove this manual from the shipping container and leave the rest of the stuff in the box. If you are verifying the receipt of complete product, check each item against the packing list enclosed. The following packing list is for reference only:

2.1.1. Packing List

Your PromICE is shipped in a custom cardboard container with cutouts for various parts. Some small items may be inside the box and may require some shaking etc. to remove. **In all cased do not remove the PromICE unit from its static shielding bag until you get to, connecting the hardware section:**

PromICE - in its static shielding bag
External power supply - in a box
7' modular cable
DB9 and DB25 adapter for RS232 (female)
5' Parallel cable
Reset mini clip
User manual - with Host software - LoadICE
Rubber feet
One or two 12" ROM cables (depending on PromICE model number)

PromICE's with model numbers beginning with "P2*xxx*" will receive two ROM cables. PromICE's with model numbers beginning with a "P1*xxx*" will receive one ROM cable.

(Optional)	12" 24,28 or 32 pin DIP Shielded ROM cable
(Optional)	12" 32 pin PLCC cable and adapter
(Optional)	12" 40 pin DIP cable and adapter
(Optional)	12" 44 pin PLCC cable and adapter
(Optional)	Male DB9 or DB25 adapters
[Optional]	Serial daisy chain adapter
(Optional)	Parallel daisy chain cable
(Optional)	Any other custom items

When transporting the PromICE unit, first place the unit in the static shielding bag that it came in. Failure to do so may result in damage to the unit and is not covered under warranty.

2.2. Software Installation

The main software you will need to accomplish all your work with the PromICE is the LoadICE application. It is distributed with a command line user interface for all systems. And for the Macintosh there is also a graphical user interface (GUI) in addition to the MPW tool. You should have received a 3.5" floppy disk, in most cases, with the application and sources to the application on it.

If the distribution disk has an executable LoadICE for your system, then you do not need to install the sources on your machine. Otherwise install the sources and use the make command to make the LoadICE executable. You would most likely do this if you are using a UNIX system.

2.2.1. DOS

Insert the floppy disk in to drive A: or B: as appropriate

If you have a directory on your system for local commands or development tools then its is best to copy just the LoadICE executable to this directory. For example, let us assume that you keep a set of your non-DOS commands in a directory called CMD. This directory should also be in your PATH. Then just do the following to install LoadICE on your system.

```
copy b:loadice.exe c:\cmd
```

Now you can execute LoadICE from anywhere in your system.

If you wish to install the whole LoadICE package then simply make a LoadICE directory and copy every thing into it:

```
md LoadICE
cd LoadICE
xcopy a:*.* /s
```

This will copy the LoadICE executable as well as the source directory. Sources may be used to rebuild LoadICE. You may choose to copy the executable to some directory that is in your PATH, or include the LoadICE directory in your path. Refer to DOS manual for modifying the PATH.

If you choose to remake LoadICE, then make the changes to the source files in the SOURCE directory, and use the make command to remake the application. You can choose the Turbo C or Microsoft C:

```
cd source
make -fmakefile.tcc
nmake -f makefile.msc
```

2.2.2. Mac

Drag the LoadICE folder to your hard disk. This folder contains the LoadICE application as well as the LoadICE MPW tool. The sources to remake the MPW tool are supplied in the MPW tool folder.

If you choose to use the application, then for frequent and direct access to it you may put the application in your *apple menu items* folder in the *system folder.* The application must first connect with the PromICE over the modem port at 9600 baud. You can then change the parameters to your desired settings and save the preferences. The application is somewhat self documenting and once you are in dialog mode (shift-LOADROM button pressed) you will have access to the commands described in this manual.

The MPW tool is used directly from MPW and is almost identical to the standard LoadICE command line application described in this manual.

2.2.3. UNIX

If you have a 3.5" floppy drive on your workstation, then the software can be installed from a DOS diskette as follows (This example is taken from a SUN workstation):

Insert the floppy disk in the drive and do the following commands:

```
mkdir LoadICE
cd LoadICE
mount /pcfs
cp /pcfs/source/*.* .
dos2unix makefile.unx makefile
make
eject
```

This will make a LoadICE executable. Ignore warning during compile that are caused by multiple includes in the UNIX header files. You need only convert the makefile from DOS to UNIX format. If you decide to edit some files then they must also be converted. You also have other makefiles that are specific to other flavors of UNIX, i.e. there is a `makefile.hp` and also `makefile.unx` as a generic make file.

You can also obtain new software over Internet from an anonymous ftp site. Some of the instructions for doing that are as follows:

```
ftp dialup.oar.net
login:anonymous
password: <your e-mail address>
cd /pub/gei/unix-sources
ftp > bin
```

send email to `support @gei.com`

2.3. Hardware Installation

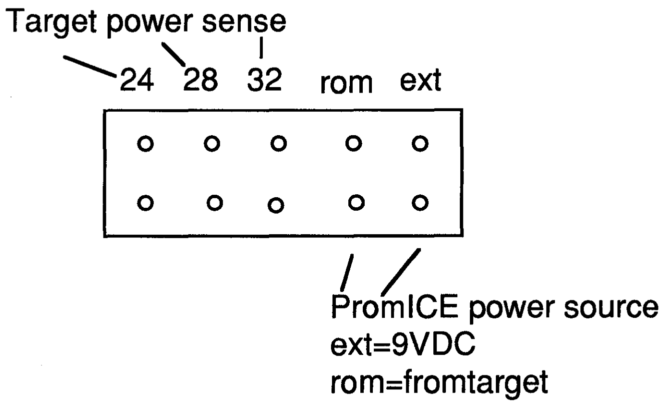
When you are ready to put the PromICE to use, bring the whole shipping box to your work area and wear the grounded wrist strap. Remove all the cables and accessories from the box, as well as the PromICE unit. Do not remove the unit from its protective bag yet.

Consider the steps you will need to take to properly hookup the PromICE to your target system. The size, number and foot print of ROMs. The options of powering the PromICE. Connecting the auxiliary signals to the PromICE. Selecting the host connections. Turning on the whole thing and getting to do real work. We will first connect the PromICE to the target system, and then to the host system. After that the power can be turned on and host can be used to operate the configuration.

Please remember that any time you are handling the connectors, whether from the target to the PromICE or from host to the PromICE, always remove power first. Never connect or disconnect anything without first turning off the power to all associated devices (except the host).

2.3.1. Connecting the PromICE to the target system

2.3.1.1. Power Source Selection for PromICE



Set one jumper on either the "ext" pins for external power (preferred) or the "rom" pins for parasitic power (When using parasitic power, the PromICE takes its power directly from the target's ROM socket. No external power supply is connected to the PromICE).

2.3.1.2. Target Power Sense Selection

Place the other jumper across the pins that correspond to the size cable being used. If you are using a 24 pin DIP package set the jumper to "24". For a 28 pin DIP set the jumper to "28". All other adapters use the "32" jumper (i.e. 32 pin DIP; 40 pin DIP; PLCC's, etc.).

PromICE uses this jumper position for two things. If you have selected "rom" as the power selection jumper then the PromICE is drawing power through the pin selected by this second jumper. i.e. from pin 24, 28 or 32 of the ROM cable.

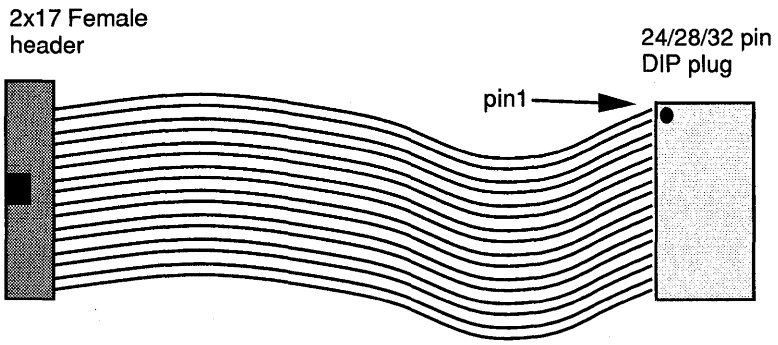
The target-power-sense circuit in the PromICE keep the PromICE from emulating when the target power is not applied. If the voltage at the pin selected by this jumper falls much below 5VDC then the PromICE will shut off emulation n and force the LOAD light to go on, indicating that the unit has shut off emulation and is now in LOAD mode. This protects data inputs on the target from going into latchup when power is applied to the target.

2.3.1.3. Connecting ROM cables

Before you plug the cable in consider the byte order if emulating multiple ROMs. You will have to describe to the software which module is plugged into which ROM. Plug the cable(s) into the target system as instructed below:

2.3.2.3.1. Connecting 28 and 32 pin DIP's

Carefully locate pin 1 of the ROM socket. It is the top left pin when the notch on the socket is on the top. If you do not understand how to locate pin 1 of a given socket, then get help with that before proceeding further, plugging this cable in backwards will smoke the PromICE or the target or both. Connect the cable to the target ROM(s) with pin 1 on the cable DIP aligned with pin 1 on the ROM socket.



WARNING: CONNECTING THE CABLE INTO THE ROM SOCKET BACKWARDS WILL DAMAGE THE PromICE.

Connect the other end of the cables to the PromICE. There is a 34 pin female IDC connector with a key. Insert it carefully on the back of the PromICE. If you are using a 16 bit or larger system, Make sure the byte order is set correctly .

2.3.2.3.2. Connecting the 40 pin DIP

Old Style Cable: There are two styles of the 40-pin DIP adapter. The old style has an adapter board that has two short 34-pin cables to connect to the PromICE and one long 40-pin for connection to the target. Connect the JP5 header to the target via the supplied cable. The cable is keyed so that it may not be connected backwards on the adapter. Connect the other end of the cable to the target. Make sure that the cable's pin 1 matches the target's pin 1. Connect header JP1 to the PromICE's master unit (bottom) or the first PromICE on the chain (daisy chain). Connect JP2 to the PromICE's slave unit (top unit).

New Style Cable: On the new style adapter, the adapter plugs directly on to the back of the PromICE unit and the 40-pin cable is attached to the 40-pin header on the center of the adapter. Make sure that the lettering on the adapter is oriented correctly, i.e. TOP is on top. Also this adapter may not have keys on the female headers that plugs into the PromICE. Make sure that these headers are aligned with the center and that all pins are mating with the opposite site.

2.3.2.3.3. Connecting 32 pin PLCC

Old Style Cable: PLCC adapter has three headers with one 34-pin cable going to the PromICE and one of the other two headers connects with the PLCC cable. Connect the PLCC cable from JP2 to the target if you are trying to emulate a 27010, 27020, 27040 or 27080 ROM. If you are using a 2764, 27128, 27256 or a 27512 connect JP3 to the target ROM. The PLCC cable should be connected to the header that matches the largest ROM that the target can support. Connect the cable to the target. Next, connect JP1 to the PromICE.

New Style Cable: The new style adapter connects directly to the PromICE and the PLCC cable is attached to the 34-pin male header. There is a separate adapter board for the 27512 and smaller ROMs and one for the 27010 and larger and all flash ROMs. Also this adapter may not have key on the female header that plugs into the PromICE. Make sure that this header is aligned with the center and that all pins are mating with the opposite site.

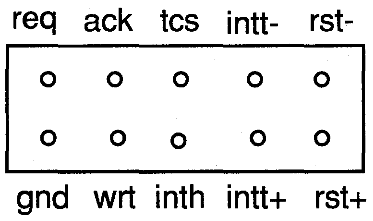
2.3.2.3.4. Connecting 44 pin PLCC

Old Style Cable: On the old style adapter: Connect the PLCC cable to the target. Connect JP2 to the master PromICE (lower) unit. Connect JP1 to the slave PromICE (upper) unit.

If the PLCC cable is not connected to the adapter board use the LL and RR markings on the connectors for proper match. If there are no markings then hold the PLCC plug with the notched corner up (you will be looking at the bottom of the connector). The cables to the right and left of the notch go to a single female connector. That connector plugs into the JP3 socket. The other side connects to JP4.

New Style Cable: The new style adapter plugs directly on to the back of the PromICE unit with the two female 34-pin headers. The PLCC cable plugs to the 44 pin male header. Make sure that the adapter is plugged in correctly since the female adapters do not have keys to line them up. The adapter board will be correctly centered when plugged in correctly.

2.3.1.4. Optional Connections



PromICE back panel has a few auxiliary signals that allow you to control the target system from the host, or allow additional features of the PromICE to interact with the target. These signals are as such:

rst- and rst+ : (outputs) These are reset signals that are driven by the PromICE whenever the unit is in LOAD mode or is instructed by the reset command from LoadICE. Both polarities of the signal are provided and they are driven by a 74HCT125 tri-state buffer. The signals are driven when asserted and go tri-state when not asserted. This allows these signals to be shared by other sources.

intt- and intt+ : (outputs) These are interrupt signals that are driven by the PromICE whenever the HDA is set in the AI status register when AI is used in transparent mode, or driven by a host request. Both polarities of the signal are provided and they are driven by a 74HCT125 tri-state buffer. The signals are driven when asserted and go tri-state when not asserted. This allows these signals to be shared by other sources.

wrt: (input) This is a low asserted input that is usually the system write line from the target. This allows the target to do write cycles to the PromICE..

inth: (input) This signal directly drives the CTS pin on the RSR232 interface on the PromICE front panel. It allows the target to directly interrupt or alert the host. Its use is limited only by your imagination.

tcs: (output)This signal is a combined chip_select and out_put enable signal. It can be monitored by a scope to see that target is indeed accessing the ROM interface. LoadICE can also determine that from PromICE status return.

req: (output) This signal is driven directly by the PromICE micro-controller to request the target systems bus for PiCOM protocol use. Its polarity is programmable via LoadICE.

ack: (input) This is the input from the target responding to req above. Its sense polarity is programmable by LoadICE.

gnd: (ground) Extra ground for general use

2.3.2.4.1. Reset Line

You may choose to connect the target reset line to the back of the PromICE. PromICE will control the target reset, either automatically, i.e. asserted while down-loading data and released when emulating, or it can be asserted by issuing direct command. The reset signal as driven by the PromICE is driven only during the asserted state and is tri-stated (i.e. not driven) during the non-asserted state. This allows the connection to be shared. However, you must make sure that your target allows a shared reset. Connecting to unsharable reset on the target will damage the drivers in the PromICE as well as on the target. Refer to hardware information about your target system

If the target reset line is low asserted connect the reset line from the target to the **rst-** pin on the PromICE. If the target reset line is high asserted connect the target reset line to the **rst+** pin on the PromICE

If you choose not to connect the reset line between the PromICE and the target system, then you must boot your target either by pressing some reset button or by power-cycling the target system.

2.3.2.4.2. Interrupt Line

If the target interrupt line is low asserted connect the **int-** line from the PromICE to the desired interrupt on the target. If the target interrupt line is high asserted connect the **int+** from the PromICE to the desired target interrupt line. Usually, you will want to connect this line to the NMI line on the target system.

The PromICE has the ability to interrupt the target system. This feature is used to return control to the debugger if your program runs away. Some debuggers will want to know which target interrupt line the PromICE is connected to.

2.3.2.4.3. Write Line

If you plan to write to the ROM space from your target system (for debugger use or otherwise) you should connect the target system *write* line to the header pin marked **wrt.** A write cycle is defined as the target access to the PromICE with the system write driven low and *chip_select* to the ROM driven low. A write cycle changes the data within the PromICE's emulation RAM. *Output_enable* signal is ignored during such a write cycle.

2.3.2. Connecting the PromICE to the host PC

You will have one of the possible four scenarios of connecting your PromICE units to the host system's ports. 1) serial only; 2) parallel only; 3) multiple units on the serial port 4) multiple units on the serial port with parallel for down-loading only.

2.3.2.1. Serial

Connect the serial cable from the host PC to the PromICE using the adapter and cables supplied with the unit. Use the DB9 or the DB25 female adapter on the PC COM ports. On Macintosh systems use the DIN8 to DB25 male adapter (standard Mac modem cable) and a DB25 female connector. On UNIX systems generally the DB25 male adapter is used. Use the 6-conductor modular cable supplied with your unit and plug it between the DB and the serial RJ11 on the PromICE front panel.

You must use the cables and adapters provided with your unit. Standard phone cables only have 4 wires and will not work.

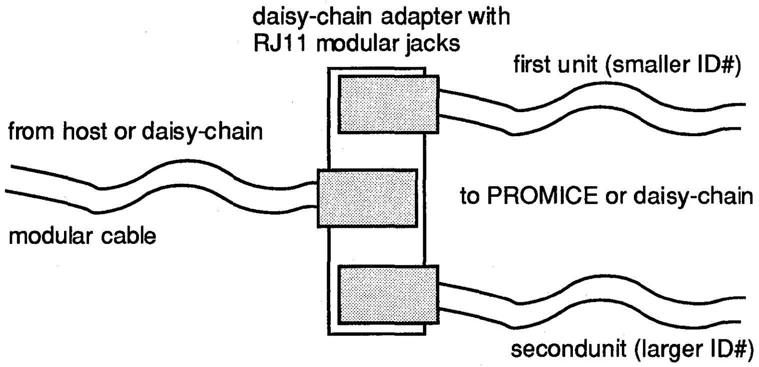
2.3.2.2. Parallel

Connect the parallel cable from the host PC to the PromICE using the cables supplied with the unit. The standard parallel cable is a 5' ribbon cable with a 26 pin (2x13) IDC female connector at one end and a DB25 male at the other. The male end of the cable goes to the PC LPT port and the IDC end plugs into the PromICE parallel port connector on the front panel.

2.3.2.3. Multiple PromICEs on the serial link

To connect multiple PromICEs to a single serial port, you must use the daisy-chain modules. A daisy chain module lets you create a loop with the transmit signal of one unit going in as the receive signal of the next. You can connector multiple daisy chain modules together to emulate large configuration of ROMs. Up to 256 modules can be daisy chained from a single serial port.

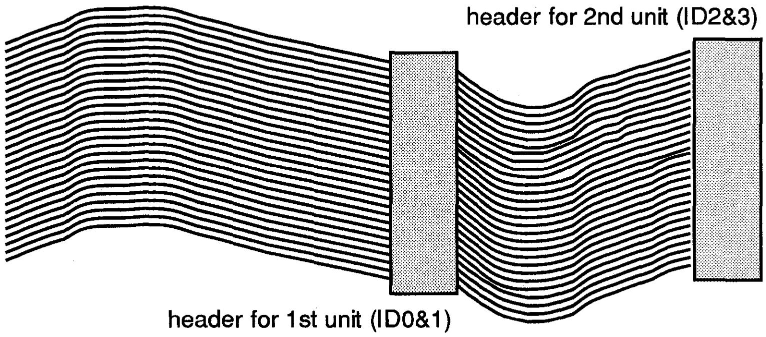
On the following page is a diagram of the daisy chain module. It is a T shape module with three RJ11 connector. The middle connector is connected to the host or the higher up daisy chain module. With the middle connector pointing toward you, the left connector goes to the first units on the chain and the right one to the next. So the units connected on the left will take the lower ID numbers and the ones of the right higher ID numbers.



With the center jack's opening facing you, connect the master (unit 0 on a simplex or units 0 and 1 on duplex units) PromICE to the left connector and the slave (unit 1 on a simplex or units 2 and 3 for duplex units) PromICE to the right (the jacks that go to the PromICE will be facing away from you). Any number of PromICE units can be daisy chained in this way.

2.3.2.4. Multiple PromICE on serial link with parallel link for down-loading

When using the PromICE in this configuration, the serial port is used for full communication with the PromICEs and the parallel link is used to achieve fast down-load times. There are two type of parallel cables to connect the PromICE units to the parallel port. The old style parallel cable only supported two units and the IDC end is shown below:.



Two PromICE units can be daisy chained in this configuration. Connect the parallel adapter cable with the center header to the first PromICE (unit 0 on a simplex or 0 and 1 on duplex units). The header farthest away from the host connection should be connected to the second PromICE (unit 1 on a simplex or units 2 and 3 on duplex units).

If you are connecting with the parallel bus mode cable, then connect the parallel cable headers to the PromICE units in the most convenient manner. The parallel bus mode cable has multiple IDCs sharing the same lines, whereas as the old style cable has special wiring for the second header. LoadICE will correctly enable the parallel ports on the units via the serial port.

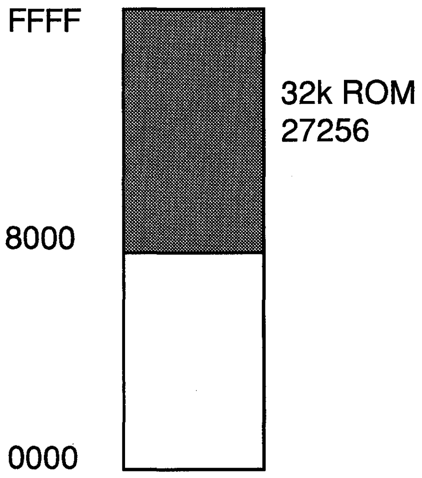
This completes the connection of the PromICE hardware. Your target system should be now connected to the PromICE properly and the PromICE should be connected to the proper host ports. You can now proceed with the setup of the host software for down-loading and emulating some code. It is best to check your setup by simply down-loading some known code and making sure that your target system and the PromICE are functioning properly and that you can repeat the down-loading and restarting process.

The Quick Start will take you through the set up for down loading and emulating some code. You can later do more sophisticated things by reading up on the commands available in LoadICE, doing things specific to some debugger you may also have purchased. You may also be planning to further exploit the capabilities of the PromICE/LoadICE by adapting your own software tools to the PromICE.

3. QUICK START

3.1. Introduction

Imagine the ROMs of your target board in the target address space. Determine exactly where they are located and exactly how they are addressed. PromICE will appear not to work, if you do not understand your hardware configuration as far as addressing the ROMs is concerned. The following is a simple example showing a 32K byte ROM device (27256) plugged in the upper half of a 64K byte address space. The ROM is addresses starting at address 0x8000. This information will be needed to correctly set all the parameters for LoadICE's use.



Your hex files for this ROM may start with address 0x8000 and must be mapped to beginning of the ROM space (i.e. 0x0000). If however your hex file contains relative addresses, or you have a binary file then no particular mapping is needed. In all cases understanding the relationship between the addresses in your data files, the address the target accesses ROM at, and the real ROM size and configuration. It is even more important if you are emulating multiple ROMs. It will save you lots of headache when trying to get your configuration running.

3.2. Software Configuration

(Creating the LoadICE.ini file)

1. The LoadICE software uses the LoadICE.ini file to initialize the PromICE to match the ROM that is being emulated. LoadICE is case sensitive. All entries should be lower case, unless otherwise specified. For more information on any of the following LoadICE.ini instructions, see the *Command Reference* section of this manual.

2. If the PromICE is using the serial port for communications, the first line in the LoadICE.ini file should read:

```
output=comX:abc
```

`X`	the serial port number being used.
`:abc`	(optional) Specify only if the port address is not standard.

3. If the PromICE is connected via the parallel port only, the specification should be:

```
pponly=lptX:abc
```

`X`	the parallel port number being used.
`:abc`	(optional) Specify only if the port address is not standard.

4 If the PromICE is connected via the parallel port in addition to the serial, the specification should be:

```
parallel=lptX:abc
```

`X`	the parallel port number being used.
`:abc`	(optional) Specify only if the port address is not standard.

You may also have a ppmode command to use bus mode of parallel port

```
ppbus 2
```

5. Specify the Socket size of the ROM to be emulated. The socket size is the largest ROM that the target can use. For example: if you are trying to emulate a 27010 (1Mbit) part but you can put a 27040 (4Mbit) part in your target without rewiring it, specify the socket size as:

```
socket=27040
```

The socket size should equal the largest ROM that can be used by the target. This will ensure that both LoadICE and the target 'see' the same address space within the PromICE units. This is specially critical when you are emulating a ROM smaller than the amount of memory the PromICE unit has. Also this will ensure that if you are using a debugger with the PromICE, it will communicate via the AI option correctly.

6. Specify the ROM size to be emulated. This size should not exceed the size specified in the socket statement above. Place the following line in the LoadICE.ini file after the baud or pponly statement:

`rom=27512`	Specify the part number of the ROM (i.e. 27512), or
`rom=64k`	Specify the ROM size in bits.

If you are using a 40 pin DIP or a 44 pin PLCC set the ROM statement to emulate half the size of the ROM you are emulating. The PromICE master and slave modules will add up to the size you want to emulate.

7. Next, specify the word size (and byteorder if necessary). The word size is the width of the target bus in bytes. Place the line as follows:

`word=8`	for an 8 bit word
`word=16 0 1`	For a 16 bit word with the first byte going to the "0" (master) PromICE unit and the second byte going to the "1" (slave) unit. To reverse the byteorder, swap the "0" and "1".

Unless another order is specified, the default order is 0, 1, 2,... The byte order allows you to hook your PromICE modules in the order most convenient for your target.

If you are using a 40 pin DIP adapter or a 44 pin PLCC adapter set the word size to 16.

8. The file specification is used to specify the file and where the data is to be loaded:

```
file=filename fileaddr=[id:] romaddr
```

`filename`	The file that is to be loaded
`fileaddr`	The fileaddress that the file starts at (i.e. the hex records in the file say to start loading the data at this address.
`id:`	(optional) The PromICE ID number that the file is to be loaded into. This is an optional specification. If the PromICE is a duplex unit or more than one unit is being used. The bottom unit in the duplex PromICE is ID 0.

`romaddr`	The location in the PromICE where the code is to be loaded.

If a binary image is to be used the line will appear as:

```
image=filename skipcount=[id:]romaddr
```

skipcount allows you to skip any data (usually load header) from the beginning of the file. There are no other differences in syntax between the two file specifications.

At this point the LoadICE.ini file should appear like one of the following:

```
output=com1
baud=57600
socket=27040
rom=27010
word=16 0 1
file=mad.hex fe000=0:0     *Sets the  file's address that starts at
                           * fe000 to position 0  in the ROM


pponly=lpt1                *This tells LoadICE the parallel port  is
socket=27080               *being used as a bi-directional link
rom=27040
word=8
file=mad.hex fe000=0       *The "0:" specification is not necessary
                           *in an 8 bit configuration

output=com1                *Use the output, baud and parallel
                           *specifications
baud=19200                 *if you want to use parallel port
                           *downloading
parallel=lpt1              *and the serial port for communications
socket=27010
rom=27256
word=8
file=mad.hex fe000=0
```

9. Apply power to the PromICE and the target system. The RUN light should come ON on the PromICE.

10 Execute LoadICE from the directory where the LoadICE.ini file is located. The RUN light will blink as LoadICE connects. If you are using the serial link, you will also see the activity on RxD and TxD lights.

11. Cycle power to the target or press the reset button (This is not necessary if the reset line is connected to the target). The LOAD light on the PromICE should have already been out. If you boot your target by power up then the LOAD light should go OFF when target power is turned on.

12. If the target won't run or the LOAD light won't go out or you are experiencing some other failure. Then double check your steps. If there is any sign of problem with power, such a lights flickering or dim on the PromICE or any target LEDs, then immediately shut off power to target system and double check all connections.

If nothing can be determined to be wrong in your preliminary search, then check your ROM configurations, file mapping and target booting process.

Consult the Quick Troubleshooting section for more help, before calling for technical support.

4. TUTORIAL

4.1. The Art of ROM

In an embedded system, the main motivator of the system is the software in the ROMs. The embedded processor relies on this single resource for most of its intelligence, performance and abilities. Rarely does an embedded system down-load code from a disk or a data link. RAM in an embedded system tends to be limited to the buffering of data, storage of variables, and the stack. ROM is the most reliable, highest density, most cost effective storage for on board intelligence.

Development of ROM code in large quantities is generally not addressed by hardware development tools such as the processor ICE. Software development is a host based activity involving text editors, compilers, assemblers, linkers, librarians, and loader, and most desirably source level debuggers. There are plenty of tools available for developing host based applications, but for developing firmware, these tools are somewhat inappropriate. When the software development cycle is interrupted by an engineering activity, such as installing the software in ROMs or operating a sophisticated ICE, productivity tends to suffer.

The PromICE offers innovative solutions based on proven technologies. It weds ROM emulation with communication between the host and the target systems. A host based debugger can debug applications in the remote system with the help of a ROM monitor. The PromICE hardware emulates the ROM space and allows control of the target system from the host. A user can down-load and debug ROM code without leaving the keyboard.

The simplest ROM code development can be achieved by emulating the ROM and thus reducing the time it takes to install new ROM code from minutes to seconds. The ability to quickly try new code changes can substantially improve productivity. The PromICE offers the best in ROM emulation and can emulate any number, size, speed and variety of ROMs.

More sophisticated debugging can be achieved by including a ROM monitor with the target application and debugging it from the host. The ROM monitor is a software probe equivalent to the hardware probe used by the processor ICE. The PromICE offers a means of communicating with the target system through the ROM socket. This facility is exploited by several industry standard host based firmware development tools as well as by the LoadICE application.

The following is a brief overview of the ROM code development process as it may be accomplished with a PromICE system:

4.1.1. Data Preparation

You may be writing your ROM code in assembly or a higher level language by using a text editor to prepare the source files. Your source files are processed by an assembler or a compiler (or both) and produce object modules (files). Optionally, the object modules may be organized in a library by a librarian program. The final application is produced by a linker program. Sometimes you may have to use additional utilities to post-process the linker output into a form suitable for installing on the target hardware, either via a ROM programmer or an emulator. Here is an overview of the steps involved in preparation of data for the ROMs.

4.1.1.1. Cross Development Tools

When software for a system is generated by another system, the process is called *cross development*. The cross development tools generally consist of a compiler, assembler, linker and often utilities to translate linker output to proper file formats. The input to these tools are the source files and the output is a file containing data in a format suitable for down-loading to a ROM programmer. These tools operate in a manner very similar to the ones intended for native application development with the added difference that the final output will not be executable on the native (host) system.

It is assumed that you are familiar with the operation of your particular cross development tools. It should be very similar to using native application development tools. Generally the main area of concern in a cross development tool is the specifications to the linker program. A native tool generally takes a list of object modules and links them into an executable image that can be invoked simply by the host command shell. A cross linker, however, needs to know where the code (ROM) and data (RAM) space is on your target system. Typically, it needs to be supplied real addresses and it is likely to produce absolutely linked files. You have to make sure that the final output contains proper addresses for the code, data and other sections of your program for proper operation of the software on your target system.

4.1.1.2. Using Native Tools for Cross Development

When the target processor is object code compatible with the host processor, native tools can be used for developing ROM code. For example: Microsoft C or Turbo C can be used to develop firmware for 80188, 80186, NEC V-series processors on a PC. The final output of the native tools must be processed by specialized utilities to produce 'ROMable' code. These utilities are generally called *locate* utilities.

These utilities serve a function similar to the linker. They take input from you about the location of ROMs and RAM in target space and appropriately positions data, code, stack, the reset vector, etc. to the right places. You must ensure that these utilities are producing properly formatted output.

Similarly, native tools of a host based on the 68xxx processor can be used to develop code for targets based on that series of processors. Similarly, the SUN workstation can be used for developing code for embedded SPARC applications.

4.1.1.3. File Formats

The most common output of Cross development systems and other utilities such as *locate*, is a 'hex' file. It is a translation of the binary data to an ASCII format. Each binary byte is translated into two bytes encoded as characters (0-9;A-F) corresponding with the hex nibbles of the original byte. The data is then stored in a record that contains a record identifier, address, count and checksum of the data in the record. Hex files are essentially text files. Most popular formats are Motorola-hex, also called S-records and Intel hex. There are several other formats supported by the LoadICE application. Straight binary files are also supported by LoadICE.

If your target system's data bus is wider than eight bits, then the cross development tools can generate multiple output files, one per ROM, or a single output file with consecutive bytes forming word length data. LoadICE can support virtually any word size files. All files should be specified with the word size and byte order for the data they contain to ensure proper down-loading by LoadICE.

4.1.1.4. Mapping Files to ROM's

A target system generally addresses its ROM space at an address other than zero. The lower addresses are occupied by RAM, so the ROM usually starts at some higher address. If your target application is linked to absolute addresses then it most likely contains addresses in its hex records that instruct data to be loaded to these higher addresses. In some cases the translation utilities may generate records where the address is relative to the beginning of the ROM space, especially when they are generating files for each ROM. In such cases the address may start at zero and go up. LoadICE allows you to specify a file offset.

You must determine if any offset is required to properly map your data files relative to the beginning of the ROM space. This offset can be specified with each file and allows proper mapping of data in files to the emulated ROM space.

4.1.2. ROM Emulation

When code is developed by burning it into ROMs and testing it in the target system, the job is made much easier by emulating the ROM(s). The time to install and test new code can be cut from hours and minutes to minutes and seconds. The PromICE can emulate a wide variety and configuration of ROMs. The ROMs are identified to the host system by their ID number, which are their position on the serial (or the parallel) link. We will briefly discuss these concepts as they apply to the PromICE usage.

4.1.2.1. Unit IDs

PromICE units contain one or two emulation modules. The lower module is called the *master* module and the upper the *slave* module. For emulating more than two ROMs, multiple units can be daisy-chained from the same port. Each module is identified by an ID# that is dynamically assigned, starting at zero, when the host establishes the link with the PromICEs. The master module always takes the first ID# and the slave module takes the next. Successive units down the chain take successive ID#s. Target ROMs are identified to the host by the ID# of the module emulating them. Within LoadICE specifications, ROMs are referred to by their ID#s.

4.1.2.2. ROM Configurations

The simplest configuration is a single ROM in an 8-bit word. Multiple ROMs on an 8-bit target would comprise multiple banks of 8-bit words. For word sizes larger than 8-bits, each bank contains the number of ROMs equal to the size of the word in bytes. The bytes in a 16-bit word can be identified as the high and low bytes, or odd and even bytes, and in a 32-bit word, as the first, second, third and fourth byte etc. A ROM configuration describes the word size, as well as the order of bytes in the word, by using the ID# of the modules emulating the corresponding bytes. You can specify a configuration for each file separately and change the current configuration dynamically.

4.1.2.3. File Specifications

After determining the ROM configurations, you must relate the files to ROMs. The general specification is the name of the file and type of data it contains (binary etc.). For hex files, LoadICE will automatically determine the data type. The file is then related to a particular configuration and an address within that configuration.

If you have a file for each ROM, and your target is a multi-byte system, you should load the files as if your ROMs are in 8 bit mode. In other words, LoadICE can process files according to the data they contain, independent of the target's data bus width.

4.1.2.4. LoadICE Application

LoadICE is an application that will take all the information you supply about the communication link, the ROM configurations, data files and download the data properly to the PromICE units emulating the ROMs. LoadICE also controls the target system to properly operate for execution and debugging of ROM code.

4.1.3. Debugging Features

You can simply emulate the ROMs and debug your code by observing the target system's behavior as you make changes to your code. The PromICE will let you edit the ROM code to try out patches, or you may make changes in your source files, reload the code and test it. This level of debugging is usually sufficient if you are making small changes to existing code that works well as it is. However, if you are developing new code, then you will need more sophisticated debugging services. You may use either the built in facilities of LoadICE for this purpose, or use on of the third party firmware development systems for debugging your code.

4.1.3.1. AiCOM and PiCOM Protocols

Debugging interfaces within the PromICE are implemented by the AIOption or by using the basic ability of the target system to do write cycles to the ROM space and the PromICE's ability to request the target bus (BusRequest/BusGrant). AI based interface is called AiCOM. It implements a ROM based UART for communicating with the target system. The other interface is called PiCOM and is a standard part of each unit. PiCOM implements a shared memory buffer in the ROM space for communication with the target. All debugging systems that work with the PromICE use one of these two interfaces. You may choose several industry standard third party packages for debugging various targets.

4.1.3.2. Source Level Debugging

If you wish to do symbolic debugging at the source level of your program, you should use other products offered by Grammar Engine Inc. to support industry standard cross development environments on the PromICE. Currently we have products which allow debugging in C and Assembler languages for the 68xxx family and the 80x86 / NEC V series processors. Contact Grammar Engine Inc. for the latest list of supported debuggers.

With all these options, your PromICE system can perform at all levels of firmware development, adapting from custom to general needs. With the software sources supplied with your unit you can support new processors or customize the PromICE / LoadICE system for your particular needs.

4.1.4. LoadICE Application

LoadICE helps you manage a group of one of more ROMs being emulated by the PromICE unit. You can download, dump, edit or upload either hex or binary files. Further flexibility is provided through the dialog mode commands. LoadICE processes input from three separate sources: the initialization file, the command line arguments, and the dialog mode input. The initialization file is processed first, hence, the command line specifications override those in the initialization file. If you use the dialog mode then you can override most of the specifications via the interactive commands.

4.1.4.1. LoadICE Defaults

If nothing is specified in the LoadICE.ini file, the command line or dialog mode input, LoadICE will default to a baud rate of 19200, and open COM1 (on the PC; */dev/ttyb* on UNIX and *modem* on Macintosh based systems) for establishing a communication link with the PromICE. It will configure all the PromICE modules found on the link in an 8 bit configuration with each module emulating the largest ROM possible for the unit.

4.1.4.2. Planning

Unit IDs And Establishment Of The Link

Before using LoadICE you need to understand the ID# assigned to each module and its relationship to the specific ROM being emulated by the PromICE. At the same time we will describe the process of establishing the link.

LoadICE establishes the link with the PromICE by sending auto baud characters down the link. The auto baud character is a control-C (0x03). The first PromICE unit receiving this character will try to determine at what baud rate it is being transmitted. It will then change its own baud rate to match. It will then again check for the proper baud rate character. If all is well it will re transmit the baud rate character out, effectively echoing it. The host will keep transmitting the baud rate character at an interval (1/10th of a second on DOS and MAC systems; 1 second in UNIX and VMS (sleep granularity)) until it receives it back as input. The more the units, the longer it takes for them to go through the auto baud sequence. It is even longer on the UNIX and VMS systems due to the granularity of the sleep system call. Once this is done there is no other penalty for daisy-chaining the units since the host is always the master.

Once the auto baud sequence is complete the host sends an ID packet. It consists of a null (0x00) and a binary ID which is initially 0. Each PromICE receiving this packet assigns the ID in the packet to its master module, and if a slave module is present, the next ID is assigned to the slave module. Then it re transmits the ID packet with the next ID value. Eventually the host gets the ID packet back with the ID value equal to the number of emulation modules on the link. From then on, the host communicates directly to each module by using its ID#. Any module receiving a packet not matching its ID will pass the entire packet to the next unit.

If the units were left in the command mode because the previous execution of LoadICE terminated abnormally, then they will respond to the auto baud character as if it is a packet for the unit with ID=3. To avoid this the LoadICE application restarts the units by asserting the DTR and forcing a reset interrupt.

On the parallel port the auto-baud is nominal (i.e. instantaneous). LoadICE uses the INIT signal to restart the unit.

ROM Configurations And Data Files

Normally the ROMs are configured according to the word size of your target processor, or more accurately the data bus width of the target system. You may have a 16 or 32 bit processor with only an 8 bit data bus and hence one ROM. For the purposes of LoadICE in this case you have an 8 bit configuration.

You will have as many PromICE modules connected to as many ROMs as you are emulating. If you tell LoadICE nothing, then all ROMs are 8 bit. However, you can specify any word size and change the configuration, in which case the ROMs are assigned as even and odd, or high and low according to their position in the word. The order may differ from architecture to architecture (i.e. Motorola is even, odd in a 16 bit pair). However, you may want to arrange your ROMs, LoadICE will let you specify the word size and byte order explicitly. You can specify the same thing for your data files. You can have a different configuration for each file if you like and can change the configuration as often as you like.

Generally speaking, you will set a word size and byte order (default is 0, 1, 2, ...etc.). LoadICE will handle the input efficiently and you can check on its configuration to see what it thinks it's doing. You will then specify data files. There is usually no need to specify the word order with files. The important issue with files is specifying their loading address, i.e. where in the ROM they should go. Consider a few things:

When you use a cross development package or some other set of utilities to produce ROM code from your source files, the final product is a HEX file. Sometimes it can be a binary file or some other format, but in most cases the HEX file is the final product. Some tools will even split this file in ODD and EVEN bytes etc., if your system has multi-byte words. You will down load these files to a ROM programmer to blast ROMs. The same files can be downloaded into to the PromICE by LoadICE. If your files are per ROM files, then they are to be specified as 8 bit files and the configuration is in 8 bit words. If your file has longer data words, such as a 16 or 32 bit data file, then the appropriate word size should be specified. You can specify the word size and byte order explicitly with each file, or you can specify a word size and byte order for the whole system. If there is any doubt about how LoadICE is interpreting your input, you should have it display the current configuration to see the setup for yourself.

4.1.4.3. Initialization File

LoadICE will look for the initialization file in three places. It may be named *LoadICE.ini* in the current directory or it may be specified by the environment variable LOADICE. It may also be specified on the command line as the very first argument in the for of *@filename*. If the first character of the command line is a '.' then the *ini* file processing is skipped altogether.

Use the following to set LoadICE in your environment to point to the desired initialization file. In some systems you may also need to '*export*' it:

```
set LOADICE=filename*
export LOADICE

* replace '=' with ' ' when using the MPW tool
```

Generally you may specify permanent items about your setup in the initialization file. This keeps the command line short and easy to retype. Such items as the name of the serial or parallel device, baud rate, size of the ROMs being emulated and their configuration are best put in the initialization file. Filenames can also be put in the initialization file. This way the command line can be invoked after each new build of the target system software.

4.1.4.4. Command Line Arguments

You can use the command line to temporarily override the initialization settings. To avoid the initialization file all together use a '.' as the first argument (with a space after it) on the command line. Generally you can use the command line to specify all the things that may be in the initialization file. The command line can be used to override those specifications.

4.1.4.5. Aliasing and Synchronizing

Aliasing is defined as multiple PromICE units all having the same IDs and all loading and executing the same commands at the same time.

Aliasing involves the use of special daisy-chain modules which broadcast the host transmit data to all the units simultaneously as their receive data. Then only one unit's transmit data is tied back the hosts receive data. By using this method, multiple units will have the same ID numbers and will do all the operations commanded by the LoadICE application in the host. LoadICE will only hear one of the units. Switches on the special daisy chain allow selection of the unit to be listened to. The switches can be used to debug and verify the aliasing arrangements. Changing the switch setting should make no difference in LoadICE operations. Duplex units and regular daisy-chain modules can be used to alias entire configurations.

While aliasing, there may be a need to synchronize the PromICE units. Basically, this means that all the PromICE units operations remain in lock-step with each other. This allows the PromICE to be used in developing concurrent systems and other fault tolerant configurations. The following are the three issues involving the use of the PromICE in such environments:

Target RESET

The RESET control signal on the back of the PromICE should be used to start up the target. This will ensure that all the targets connected to the alias units will start at the same time. The PromICE will hold the RESET signal asserted while the units are loading and then it will issue an EMULATE command and all RESETs will be released at the same time.

AiCOM - virtual channel

A custom AI is needed to synchronize the changing of the AI status to the target's access.

Timer interrupts:

In order to ensure that there are no asynchronous events within the PromICE to alter the flow of data through the AI option, the periodic timer interrupt needs to be turned off. This interrupt occurs every 8.8 milli-seconds and is used to determine whether or not to break the transparent link on the host's request (toggling the DTR line). The PromICE units would have to be manually reset to break the AI link.

5. Sample Sessions

This section will illustrate the use of the PromICE / LoadICE system and show you how to set up various configurations and how the output for these configurations should appear. To exit the LoadICE dialog mode, type a lower case 'x' at the prompt. The LoadICE prompt is preceded by a number that indicates the number of seconds required to complete the previous command. If the command takes longer than 60 seconds then minutes are also displayed and so on. This allows simple performance measurement of LoadICE operations.

5.1. The LoadICE To PromICE Communication Link

If you simply run LoadICE on the host system with no ini file specified, LoadICE will use built in defaults to connect to the PromICE and enter dialog mode when the link comes up:

```
C:\LoadICE>LoadICE

LoadICE V 2.3a.04
Copyright (C) 1990-1993 Grammar Engine, Inc.
Opening command file 'LoadICE.ini'
File not found 'LoadICE.ini' - proceeding
Entering dialog mode
Establishing link with PromICE (please WAIT)
Opening serial device 'COM1'(3f8) @BR-19200..
Link is up

 PromICE units:
  ID 0 uCVer:7.1a MemSize=131072 EmuSize=131072 FillChar=FF
  ID 1 uCVer:7.1a MemSize=131072 EmuSize=131072 FillChar=FF
2:LoadICE:
```

There is a P2010 unit attached to the serial link 'com1'. LoadICE is using the default device name and default baud rate. Notice the emulation size default is equal to the maximum capacity device possible. The fill character is 0xFF. Following is a description of the information following the "PromICE units" line above:

`ID x`	The PromICE ID number. On a duplex unit (i.e. any unit starting with a 'P2...' in the model number) the bottom connector on the back is ID 0 and the top connector is ID 1.
`uCVer:xx`	This is the micro code version of the PromICE unit. In this case it is 7.1a.

`MemSize=`	The size of the memory physically inside the PromICE. THIS IS NOT NECESSARILY THE SIZE YOU ARE CURRENTLY EMULATING. Only if no LoadICE.ini, command line switches or dialog mode commands are issued to change it, then the emulation size will be same as the physical memory. The size is given in bits.
`EmuSize=`	This is the emulation size. This will change as you change the emulation size in your command line, LoadICE.ini, or dialog mode. Again, the size is given in bits.
`FillChar=`	The current fill character (FF).

5.2. Changing The Emulation Size

Next, lets change the baud rate and emulate 27512 ROMs (64k bytes). Rerun the LoadICE application with these two specifications on the command line (if you are using another port other than com1 place '*-o comx*' before the '-b'):

```
LoadICE -b 57600 -r 27512

LoadICE V 2.3a.04
Copyright (C) 1990-1993 Grammar Engine, Inc.
Opening command file 'LoadICE.ini'
File not found 'LoadICE.ini' - proceeding
Executing the command line '-b 57600 -r 27512'
Entering dialog mode
Establishing link with PromICE (please WAIT)
Opening serial device 'COM1'(3f8) @BR-57600..
Link is up

 PromICE units:
  ID 0 uCVer:7.1a MemSize=131072 EmuSize=65536 FillChar=FF
  ID 1 uCVer:7.1a MemSize=131072 EmuSize=65536 FillChar=FF
27:LoadICE:
```

Note that the command line is all lower case. All of the command line codes, unless otherwise described, are in lower case. LoadICE IS CASE SENSITIVE.

The 'EmuSize' is now 65536. It is now emulating a 64k byte part (divide the 'EmuSize' by 1024 to get this number).

5.3. Displaying The Configuration (dialog mode)

From the LoadICE prompt type a capital "C":

```
02:LoadICE: C


 Serial link:COM1(3F8) @57600 baud


 PromICE units:
  ID 0 uCVer:7.1a MemSize=131072 EmuSize=65536 FillChar=FF
  ID 1 uCVer:7.1a MemSize=131072 EmuSize=65536 FillChar=FF


No files are specified

 ROM config:  totalRange=[00000000-0001FFFF]
   (00A7D0):  word=8 IDs=0 addressRange=[000000-00FFFF]
(linked)
   (00A80A):  word=8 IDs=1 addressRange=[010000-01FFFF
00:LoadICE:
```

Notice that there are no files specified and the ROMs are in the 8 bit word mode by default. Also, the first ROM (master module) is from 0000-FFFF and the second one starts after the first one. LoadICE will automatically switch from the first to the second as the address crosses the boundary.

5.4. Loading A File From Dialog Mode

For this section, you will need a hex file to load into the PromICE. Run LoadICE:

```
LoadICE -b 57600 -r 27010
```

If your PromICE unit can't emulate this size, set it to a smaller emulation size (i.e. 27512).

```
LoadICE V 2.3a.04
Copyright (C) 1990-1993 Grammar Engine, Inc.
Opening command file 'LoadICE.ini'
File not found 'LoadICE.ini' - proceeding
Executing the command line '-b 57600 -r 27512'
Entering dialog mode
Establishing link with PromICE (please WAIT)
Opening serial device 'COM1'(3f8) @BR-57600..
```

```
Link is up

 PromICE units:
  ID 0 uCVer:7.1a MemSize=131072 EmuSize=65536 FillChar=FF
  ID 1 uCVer:7.1a MemSize=131072 EmuSize=65536 FillChar=FF
27:LoadICE: stop
Use 'go' later to emulate
```

Next, load the file specifying any offsets necessary:

```
27:LoadICE: l PromICE.hex fe000=0
Opening file 'PromICE.hex' for processing\
Data 00000010-00001267-
Data 00001270-000012AB\
Data 00001F00-00001F16/
Data 00001FF0-00001FF5/Done
Transferred 4781 data bytes
02:LoadICE:
```

In this case, the 'fe000=0' tells LoadICE to set the file address (fe000) to load at physical address 0 in the PromICE.

5.5. Loading A New File From Dialog Mode

Following the process from the last session, lets load a new file. First, fill the PromICE's memory to eliminate the old file:

```
02:LoadICE: f
Filling config from 0 to 3FFFF fill char=FF \
Transferred 262144 fill characters
53:LoadICE: stop
Use 'go' later to emulate
02:LoadICE:
```

If you were to dump the information from the PromICE at this time, the unit would have only 'FF's in every location (to prove this to yourself, do a 'd' at the LoadICE prompt). The next step is to specify the new file:

```
53:LoadICE: l tdrem.hex fe000=0
Opening file 'tdrem.hex' for processing\
Data 00000010-000013DE\
Data 000013E0-000013F2|
Data 00001400-00001410/
Data 00001F00-00001F39\
Data 00001FF0-00001FF5/Done
Transferred 5166 data bytes
03:LoadICE:
```

Your new file is now loaded into the PromICE. All trace of the old program have been wiped from memory. Loading multiple files into the PromICE will be discussed later.

5.6. Loading A List Of Files From Dialog Mode

To load multiple files into the PromICE, you must first plan where the programs will load and what there address ranges will be. You will need two small hex files for this section. First, we will get into dialog mode and load one of the files:

```
LoadICE -b 57600 -r 27010

LoadICE V 2.3a.04
Copyright (C) 1990-1993 Grammar Engine, Inc.
Opening command file 'LoadICE.ini'
File not found 'LoadICE.ini' - proceeding
Executing the command line '-b 57600 -r 27010'
Entering dialog mode
Establishing link with PromICE (please WAIT)
Opening serial device 'COM1'(3f8) @BR-57600..
Link is up

 PromICE units:
  ID 0 uCVer:7.1a MemSize=131072 EmuSize=131072 FillChar=FF
  ID 1 uCVer:7.1a MemSize=131072 EmuSize=131072 FillChar=FF
03:LoadICE: stop
Use 'go' later to emulate
```

Next, load the file specifying any offsets necessary:

```
27:LoadICE: l PromICE.hex fe000=0
Opening file 'PromICE.hex' for processing\
Data 00000010-00001267-
Data 00001270-000012AB\
Data 00001F00-00001F16/
Data 00001FF0-00001FF5/Done
Transferred 4781 data bytes
02:LoadICE:
```

The first file loaded into address 0 of the PromICE. This file ends at 0x1FF5. Now we will load the other file:

```
02:LoadICE: l tdrem.hex fe000=2000
Opening file 'tdrem.hex' for processing/
Data 00002010-000033DE/
Data 000033E0-000033F2-
Data 00003400-00003410\
Data 00003F00-00003F39/
Data 00003FF0-00003FF5/Done
Transferred 5166 data bytes
02:LoadICE:
```

In this way, you can specify as many files as you want and memory can handle. By dumping (d) the information between the two files (in this case address ranges 1FF5 to 2010) you can see were one file stops and the other starts. Later, we will discuss loading files 'per ROM'.

5.7. Dialog Mode Editing (patching)

Next, lets try editing some address locations. First, fill the ROM space with 'FF's:

```
00:LoadICE: f
Filling config from 0 to 1FFFF fill char=FF/
Transferred 131072 fill characters
26:LoadICE:
```

Edit location FFFE:

```
26:LoadICE: e fffe fe ed de af
-
00:LoadICE:
```

Next, dump that address range:

```
00:LoadICE: d fffe 1000f
Dump: word=8 ID 0 |
0000FFFE:                                                FE ED |   ..|
00010000: DE AF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF |.....|
00:LoadICE: d 1:0 1f
Dump: word=8 ID 1 \
00000000: DE AF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF |.....|
00000010: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF |.....|
00:LoadICE:
```

The command '`d 1:0 1f`' tells the PromICE to go to unit 1 and dump address 0 through 1f.

5.8. Setting Up A LoadICE.ini File

In this section, we will put some of the specifications used in previous sections in the LoadICE.ini file. Create a LoadICE.ini file and put the following lines in it:

```
output=com1
baud=57600
word=8
rom=27256
noverify
dialog
```

Now, rerun LoadICE:

```
LoadICE V 2.3a.04
Copyright (C) 1990-1993 Grammar Engine, Inc.
Opening command file 'LoadICE.ini'
Entering dialog mode
Establishing link with PromICE (please WAIT)
Opening serial device 'COM1'(3f8) @BR-57600..
Link is up

 PromICE units:
  ID 0 uCVer:7.1a MemSize=131072 EmuSize=32768 FillChar=FF
  ID 1 uCVer:7.1a MemSize=131072 EmuSize=32768 FillChar=FF
03:LoadICE:
```

Lets look at the configuration as LoadICE sees it. Type a capital 'C' at the prompt:

```
03:LoadICE: C

 Serial link:COM1(3F8) @57600 baud

 PromICE UNITS:
  ID 0 uCVer:7.1a MemSize=131072 EmuSize=32768 FillChar=FF
  ID 1 uCVer:7.1a MemSize=131072 EmuSize=32768 FillChar=FF

No files are specified

 ROM config: totalRange=[00000000-00007FFFF]
   (00A71A): word=8 IDs=0 addressRange=[000000-007FFFF]
00:LoadICE:
```

Notice that LoadICE displays the actual hardware configuration under "*PromICE units*" and then the specified configuration under the "*ROM config*".

5.9. Specifying 16-Bit Files

For this example you will need a small hex file to download to the PromICE. You will also need one duplex (Model # P2*xxx*) or two simplex (model # P1*xxx*) PromICE(s). See the installation section of this manual if you need to daisy chain two simplex PromICE's. Add the line "`fill ffff`" to the LoadICE.ini file. Then, add the '`file=`' specification to the LoadICE.ini file as follows:

```
output=com1
baud=57600
word=16
rom=27256
noverify
fill ffff
```

```
file=PromICE.hex fe000=0:0
dialog
```

The file addressing specifications may need to be changed to match the hex file that you created. Now run LoadICE again:

```
LoadICE V 2.3a.04
Copyright (C) 1990-1993 Grammar Engine, Inc.
Opening command file 'LoadICE.ini'
Entering dialog mode
Establishing link with PromICE (please WAIT)
Opening serial device 'COM1'(3f8) @BR-57600..
Link is up

 PromICE units:
  ID 0 uCVer:7.1a MemSize=131072 EmuSize=32768 FillChar=FF
  ID 1 uCVer:7.1a MemSize=131072 EmuSize=32768 FillChar=FF
02:LoadICE:C

Serial link:COM1(3F8) @57600 baud

PromICE units:
  ID 0 uCVer:7.1a MemSize=131072 EmuSize=32768 FillChar=FF
  ID 1 uCVer:7.1a MemSize=131072 EmuSize=32768 FillChar=FF

File-1  PromICE.hex type-Intel Hex offset=FFC00000
skip=00000000
          (if partial load): start/00000000 end/00000000
 data config: totalRange=[00000000-00007FFFF]
     (00A754): word=16 IDs= 0 addressRange=[000000-007FFF]

ROM config: totalRange=[00000000-0000FFFF]
    (00A71A): word=16 IDs= 0 1 addressRange=[000000-00FFFF]
00:LoadICE: stop
00:LoadICE: l
Opening file 'PromICE.hex' for processing\
Data 00000010-00001267-
Data 00001270-000012AB\
Data 00001F00-00001F16/
Data 00001FF0-00001FF5/Done
Transferred 4781 data bytes
01:LoadICE:
```

The LoadICE.ini file set the word size to 16 bits and also specified a file that is linked to absolute address 0xfe000 (and hence, the hex records say to load the data starting a 0xfe000). This is a small file and we typed 'l' at the LoadICE prompt that caused the file to be processed and downloaded. We will do a test to make sure that the file is indeed downloaded. Edit a couple of bytes and do a compare:

```
01:LoadICE: e 200 abcd
/
00:LoadICE: c
```

```
Opening file 'PromICE.hex' for processing/
compare failed - 000100: f/E8 r/AB
Data 00000010-00001267-
Data 00001270-000012AB/
Data 00001F00-00001F16\
Data 00001FF0-00001FF5/Done
Verified 4781 data bytes
07:LoadICE:
```

LoadICE reprocessed the file and instead of down-loading, it up-loaded and compared the data from the PromICE against the file data. The location modified as a word (we are in 16-bit mode) at 0x200 LoadICE reported the addresses and files versus ROM contents that did not match.

5.10. Specifying 32-Bit Files

To emulate on a 32-bit target you will need two duplex, four simplex or a combination of a duplex and two simplex PromICE's. See the installation section to daisy chain the units together. Edit your LoadICE.ini file to appear as follows then execute the LoadICE command:

```
output=com1
baud=57600
word=32
rom=27256
noverify
file=PromICE.hex fe000=0
dialog
```

```
LoadICE V2.3a.04
Copyright (C) 1990-1993 Grammar Engine, Inc.
Opening command file 'LoadICE.ini'
Entering dialog mode
Establishing link with PromICE (please WAIT)
Opening serial device 'COM1'(3f8) @BR-57600
Link is up

PromICE units
 ID 0 uCVer:7.1a MemSize=131072 Emusize=32768 FillChar=FF
 ID 1 uCVer:7.1a MemSize=131072 Emusize=32768 FillChar=FF
 ID 2 uCVer:7.1a MemSize=131072 Emusize=32768 FillChar=FF
 ID 3 uCVer:7.1a MemSize=131072 Emusize=32768 FillChar=FF
02:LoadICE:
```

The two units in this case are identical PromICE Model P2010's. They could just as easily be four P1010's or a combination of both. When daisy chaining PromICE's, you don't need to have two or more identical units. They can be different model numbers and/or even different revisions. So that we can be sure to start with clear memory, we will first fill memory with 'FF'. Now, execute a load command:

```
01:LoadICE: stop
Use 'go' later to emulate


00:LoadICE: f ffffffff
Filling config from 0 1FFFF fill char=FFFFFFFF\
Transferred 131072 fill characters
23:LoadICE: l
Opening file 'PromICE.hex' for processing\
Data 00000010-00001267\
Data 00001270-000012AB\
Data 00001F00-00002F16\
Data 00001FF0-00001FF5/Done
Transferred 4781 data bytes
01:LoadICE:
```

Next, we will load the file into a particular PromICE unit. For now, we will load the file into the third PromICE unit (ID 2). This time we will also specify the fill command in the LoadICE.ini file. Edit the LoadICE.ini file to appear as follows and execute LoadICE:

```
output=com1
baud=57600
word=32
rom=27256
noverify
fill ffffffff
file=PromICE.hex fe000=2:0
dialog
```

```
LoadICE V2.3a.04
Copyright (C) 1990-1993 Grammar Engine, Inc.
Opening command file 'LoadICE.ini'
Entering dialog mode
Establishing link with PromICE (please WAIT)
Opening serial device 'COM1'(3f8) @BR-57600
Link is up

PromICE units
 ID 0 uCVer:7.1a MemSize=131072 Emusize=32768 FillChar=FFFF
 ID 1 uCVer:7.1a MemSize=131072 Emusize=32768 FillChar=FFFF
 ID 2 uCVer:7.1a MemSize=131072 Emusize=32768 FillChar=FFFF
 ID 3 uCVer:7.1a MemSize=131072 Emusize=32768 FillChar=FFFF
03:LoadICE: l
Filling ROM[ID=0]/char=FF (via uCode)
Filling ROM[ID=1]/char=FF (via uCode)
Filling ROM[ID=2]/char=FF (via uCode)
Filling ROM[ID=3]/char=FF (via uCode)
Opening file 'PromICE.hex' for processing-
Data 00000010-00001267/
Data 00001270-000012AB-
Data 00001F00-00002F16|
Data 00001FF0-00001FF5/Done
Transferred 4781 data bytes
30:LoadICE: d 0 ffff
Dump: word=32 ID 0 1 2 3 |
00000000 FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF |.. ..|
00000010 FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF |.. ..|
00000020 FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF |..  ..|
00000030 FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF |..  ..|
00000040 FF FF FA FF FF FF FC FF  FF FF B9 FF FF FF 00 FF |..  ..|
00000050 FF FF 00 FF FF FF E2 FF  FF FF FE FF FF FF E2 FF |..  ..|
00000060 FF FF FE FF FF FF B8 FF  FF FF 40 FF FF FF 00 FF |..  @.|
00000070 FF FF 8E FF FF FF D0 FF  FF FF BC FF FF FF 30 FF |..  .0|
00000080 FF FF 06 FF FF FF B8 FF  FF FF 40 FF FF FF 00 FF |..  @.|
00000090 FF FF 8E FF FF FF C0 FF  FF FF B8 FF FF FF 27 FF |..  ..|
000000A0 FF FF FF FF FF FF 8E FF  FF FF D8 FF FF FF BE FF |..  ..|
000000B0 FF FF 00 FF FF FF 00 FF  FF FF 8B FF FF FF FE FF |..  ..|
000000C0 FF FF B9 FF FF FF 40 FF  FF FF 00 FF FF FF 2B FF |.@  .+|
000000D0 FF FF CF FF FF FF D1 FF  FF FF E9 FF FF FF F3 FF |..  ..|
000000E0 FF FF A5 FF FF FF 06 FF  FF FF 1F FF FF FF 33 FF |..  .3|
000000F0 FF FF C0 FF FF FF BF FF  FF FF 40 FF FF FF 00 FF |..  @.|
00:LoadICE:
```

The dump command can be used to dump any address range desired. The first hex value after the 'd' is the starting address and the second value is the stop address. Experiment by changing the LoadICE.ini file ID to load the file into the different ROMs.

5.11. Loading Multiple Files Into The PromICE

Edit your LoadICE.ini file as follows (if your PromICE unit isn't large enough to emulate a 27010 user smaller files and emulate the largest size possible):

```
output=com1
baud=57600
word=8
rom=27010
noverify
fill ff
file=PromICE.hex fe000=0
dialog
```

In this configuration, the file 'PromICE.hex' will load into the slave unit (notice that we are back in 8 bit emulation for now). To load another file into the PromICE, simply add another 'file=' statement to the LoadICE.ini file with your new filename and address specifications:

```
output=com1
baud=57600
word=8
rom=27010
noverify
fill ff
file=PromICE.hex fe000=0
file=tdrem.hex fe000=2000
dialog
```

The first file will load at PromICE location 0 and the second will load at PromICE location 2000. The first file ends at 1FF5. The second file will end at 3FF5. When specifying your files, make sure that the addresses do not overlap. You can specify two different starting addresses to LoadICE, but if one starting address overlaps the other file, LoadICE will not give you an error. BE SURE OF YOUR MAPPING! Let's run LoadICE with this LoadICE.ini file:

```
LoadICE V 2.3a.04
Copyright (C) 1990-1993 Grammar Engine, Inc.
Opening command file 'LoadICE.ini'
Entering dialog mode
Establishing link with PromICE (please WAIT)
Opening serial device 'COM1'(3f8) @BR-57600..
Link is up

 PromICE units:
  ID 0 uCVer:7.1a MemSize=131072 EmuSize=131072 FillChar=FF
  ID 1 uCVer:7.1a MemSize=131072 EmuSize=131072 FillChar=FF
03:LoadICE: stop
Use 'go' later to emulate

00:LoadICE: 1
Filling ROM[ID=0]/char=FF (via uCode)
Filling ROM[ID=1]/char=FF (via uCode)
Opening file 'PromICE.hex' for processing\
Data 00000010-00001267-
Data 000027E0-000012AB\
Data 00001F00-00001F16/
Data 00001FF0-00001FF5/Done
Opening file 'tdrem.hex' for processing/
Data 00002010-000033DE/
Data 000033E0-000033F2-
Data 00003400-00003410\
Data 00003F00-00003F39/
Data 00003FF0-00003FF5/Done
Transferred 9947 data bytes
08:LoadICE: d 1ff0 203f
Dump: word=8 ID 0 -
00001FF0: EA 00 00 F0 FF FF FF FF  FF FF FF FF FF FF FF FF |.. ..|
00002000: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF |.. ..|
00002010: FA FC B9 00 00 E2 FE E2  FE B8 40 00 8E D0 BC 10 |.. @.|
00002020: 08 B8 40 00 8E C0 B8 3E  FF 8E D8 BE 00 00 8B FE |@> ..|
00002030: B9 20 00 2B CF D1 E9 F3  A5 06 1F 33 C0 BF 20 00 |+. 3 |
00:LoadICE:
```

The dump 'd' of the address rang 1ff0 to 203f shows where the first file ends and the second begins.

5.12. Loading "Per ROM" Files

Using the LoadICE.ini file from the last section, set the word statement to 16:

```
output=com1
baud=57600
word= 16
rom=27010
noverify
fill ff
```

```
file=PromICE.hex fe000=0
file=tdrem.hex fe000=2000
dialog
```

Next, modify the file specifications as follows:

```
file=PromICE.hex fe000=0:0
file=tdrem.hex fe000=1:2000
```

The reason the address offsets are left in is so that the point can be demonstrated more clearly. Notice the '*0:*' and '*1:*' in the specifications. The '0:' tells LoadICE to load the 'PromICE.hex' file into the first PromICE (bottom) unit. The '1:' tells LoadICE to load 'tdrem.hex' into the second (top) unit. Since, in this configuration, each unit represents a separate ROM, one file is being loaded into one ROM and one into the other ROM.

Now, execute LoadICE:

```
LoadICE V 2.3a.04
Copyright (C) 1990-1993 Grammar Engine, Inc.
Opening command file 'LoadICE.ini'
Entering dialog mode
Establishing link with PromICE (please WAIT)
Opening serial device 'COM1'(3f8) @BR-57600..
Link is up

 PromICE units:
  ID 0 uCVer:7.1a MemSize=131072 EmuSize=131072 FillChar=FF
  ID 1 uCVer:7.1a MemSize=131072 EmuSize=131072 FillChar=FF
03:LoadICE: stop
Use 'go' later to emulate

00:LoadICE: l
Filling ROM[ID=0]/char=FF (via uCode)
Filling ROM[ID=1]/char=FF (via uCode)
Opening file 'PromICE.hex' for processing\
Data 00000010-00001267-
Data 000027E0-000012AB\
Data 00001F00-00001F16/
Data 00001FF0-00001FF5/Done
Opening file 'tdrem.hex' for processing/
Data 00002010-000033DE/
Data 000033E0-000033F2-
Data 00003400-00003410\
Data 00003F00-00003F39/
Data 00003FF0-00003FF5/Done
Transferred 9947 data bytes
08:LoadICE:
```

Do a dump of the even address' 0 to ff (`d 0:0 ff`). Notice that the file is loaded into the even bits (or unit '0'). Now do a dump of the odd address' 0 to ff (`d 1:0 ff`). Notice that there is nothing but fill character 'FF'. Next dump the odd address' 1ff0 2030 (`d 1:1ff0 2030`). This is where the second file should be located (address 2010). Experiment with different address locations ROM orders. Beware of the 'address out of range error'. This means that somewhere you have tried to load a file that either starts at an invalid address or runs into an invalid address range.

5.13. Loading Binary Files

Loading a binary file is similar to loading a hex file. For example, lets take the LoadICE.ini file from the last session and edit out the file specifications:

```
output=com1
baud=57600
word= 16
rom=27010
noverify
fill ff
                *add image spec. here
dialog
```

Next, put the following line where the file specification was:

```
image=PromICE.bin
```

Substitute the filename of the binary file and address offsets for your particular file. If you don't want to create a binary image from scratch, use a working ROM and back it up as an image through the ROM programmer. The addressing information appears the same as in the '*file=*' specification. Now, run LoadICE again:

```
LoadICE V 2.3a.04
Copyright (C) 1990-1993 Grammar Engine, Inc.
Opening command file 'LoadICE.ini'
Entering dialog mode
Establishing link with PromICE (please WAIT)
Opening serial device 'COM1'(3f8) @BR-57600..
Link is up

 PromICE units:
  ID 0 uCVer:7.1a MemSize=131072 EmuSize=131072 FillChar=FF
  ID 1 uCVer:7.1a MemSize=131072 EmuSize=131072 FillChar=FF
02:LoadICE: stop
Use 'go' later to emulate

00:LoadICE: 1
```

```
Filling ROM[ID=0]/char=FF (via uCode)
Filling ROM[ID=1]/char=FF (via uCode)
Opening file 'PromICE.bin' for processing/Done
Transferred 32768 data bytes
12:LoadICE:
```

If your file contains header information that you don't want to load you can tell LoadICE to skip this information. Let's assume that your binary file has a 14 byte load header to be skipped.

```
output=com1
baud=57600
word= 16
rom=27010
noverify
fill ff
image=PromICE.bin 14=0
dialog
```

When LoadICE is executed and the file loaded, LoadICE will skip the first 14 (hex) bytes of the file and start loading at address 0. To specify a partial load, the image specification should appear as:

```
image=PromICE.bin 14=0 (0 ff)
```

Execute LoadICE again. Execute the following commands from dialog mode:

```
02:LoadICE: stop
Use 'go' later to emulate

00:LoadICE: 1
Filling ROM[ID=0]/char=FF (via uCode)
Filling ROM[ID=1]/char=FF (via uCode)
Opening file 'PromICE.bin' for processing/Done
Transferred 256 data bytes
06:LoadICE:
```

This time the file loaded only part of the file. Do a dump of about the first 256k bytes of data:

```
06:LoadICE: d 0 10f
```

Experiment with various offsets and header skips. Watch that you don't go out of your address range when loading. Other than the differences specified above, the '*image*' is handled identically to the '*file*' specification.

5.14. Saving The PromICE Contents To A File

Let's start this section with a LoadICE.ini file from a previous section:

```
output=com1
baud=57600
word=16
rom=27010
noverify
fill ffff
file=PromICE.hex fe000=0
dialog
```

Run LoadICE:

```
LoadICE V 2.3a.04
Copyright (C) 1990-1993 Grammar Engine, Inc.
Opening command file 'LoadICE.ini'
Entering dialog mode
Establishing link with PromICE (please WAIT)
Opening serial device 'COM1'(3f8) @BR-57600..
Link is up

 PromICE units:
  ID 0 uCVer:7.1a MemSize=131072 EmuSize=131072 FillChar=FF
  ID 1 uCVer:7.1a MemSize=131072 EmuSize=131072 FillChar=FF
02:LoadICE: stop
Use 'go' later to emulate

00:LoadICE: 1
Filling ROM[ID]/char=FF (via uCode)
Filling ROM[ID]/char=FF (via uCode)
Opening file 'PromICE.hex' for processing\
Data 00000010-00001267\
Data 00001270-000012AB/
Data 00001F00-00001F16/
Data 00001FF0-00001FF5/Done
Transferred 4781 data bytes
07:LoadICE:
```

Now, lets edit some data

```
07:LoadICE: e 1f00 fec5 ffdd c426 df24
/
00:LoadICE: d 1f00
Dump: word=16 RD 0 1 \
00001F00: FE C5 FF DD C4 26 DF 24
00001F10:
00001F20:
00001F30:
00:LoadICE:
```

Since the first four bytes were all that were edited, they are the only one's shown. Next, we will save the contents of the PromICE to a file:

```
00:LoadICE: s editfile 0 1ff5
Saving to file editfile (data count=8182)\
11:LoadICE:
```

The file will be saved in a binary format. To load the file back into the PromICE, from dialog mode simply type the following command:

```
11:LoadICE: li editfile
```

This will load the file into the addresses from which it was saved.

6. LoadICE Command Reference

6.1. OVERVIEW

LoadICE commands fall in to four major categories. They are summarized here for a quick view. You should be able to find the commands under proper category and then look up the detail in the following reference pages:

6.1.1. Host to PromICE communication

Allows you to specify the link between the host and PromICE unit(s). You may be using both the serial and the parallel link. These commands allow you to completely specify your setup:

output	serial device name
baud	baud rate
slow	to limit the PromICE transmit data rate on serial link
fast	to let PromICE transmit at full rate (older units)
high	to speed up the PromICE transmit data rate on serial link
number	number of PromICE units on daisy chained on serial link
parallel	parallel device name
pponly	parallel bi-directional device name
ppmode	parallel port operation options

6.1.2. ROM specifications and ROM operations

Next you will need to describe your ROM configuration. The number, size and arrangement of ROMs must be specified and any peculiarities of the target addressing , such as miss match between sockets and ROMs must be specified for proper operation. Finally all the operations that you may do on the ROM data etc. are specified:

rom	size of rom to be emulated
word	word size for roms being emulated
socket	target rom socket size

checksum	rom checksum specifications
fill	rom fill specification
dump	for dumping rom data
edit	for editing rom data
move	for moving rom data around
search	to look for ASCII data in rom space
find	to look for binary data in rom space
save	to save rom contents to a file
test	for testing the emulation memory

6.1.3. File specifications and file operations

Here you will be able to specify the data files on the host system, by their name, type and any configuration information such as word size of the data they contain, or special mapping of the data to ROM space. Also various loading and processing options can be specified:

file	specify hex data files
image	to specify binary data files
load	to down-load the data files
compare	to compare files with rom contents
noaddrerr	to ignore data that falls out of rom space
nochecksum	to ignore hex record checksums
noverify	save time while down-loading data
map	to turn off or on the display of data areas being loaded
save	to save rom contents to a file

6.1.4. Miscellaneous specifications

Various parameters that control the PromICE operation or affect the target directly are specified here:

dialog	enter dialog mode on startup
display	change output level detail
stop	stop PromICE units form emulating
go	set PromICE units to emulate
help	give on line help
reset	set default reset time length
ver	report LoadICE and PromICE micro-code versions
hso	define the operation and polarity of the interrupt signal
config	display configuration data in use
cursor	enable or disable the spinning cursor
nolights	disable blinking of RUN light
notimer	disable the PromICE internal timer

noautorecovery	don't try to restart link after timeout
fkey & altfkey	allows assigning commands to function keys
!system	allows to escape commands to the host shell
exit	exit LoadICE dialog mode
delay	change the time out period used by LoadICE
ulock	support the LOCKED units
status	display target status (power on; executing)

6.1.5. Debug setup commands

Generally the most common debug setup is done by the **ailoc** command. This command will let you program the PromICE (equipped with the AI option) to act as a transparent link between the host based debugger and the target resident monitor. This allows you to debug your target code without requiring addition hookups for communicating with the target system.

The rv command invokes a link up with the down-loaded minimal monitor supplied by Grammar Engine, for popular targets. This monitor can be used for testing the AI link with the target, or extend it to do minimal debugging for special cases.:

ailoc	specify the address of AiCOM virtual channel
rv	enter ROMview interface

6.2. ailoc

Set up a communication channel between the host and the target system over the same host port where the PromICE is plugged in. A Host based debugger can then communicate with the target monitor over the PromICE equipped with the AI option.

6.2.1. Command Forms

ailoc	in `LoadICE.ini` file
-a	as a command line argument to LoadICE

6.2.2. Syntax

{ailoc | -a } [*id* {:}] {*address*} {*baud*} [*break_char*] [*int_count*]

6.2.3. Use

id	(integer) A valid PromICE unit ID number (0-255), when specified it must be followed by a ':'. This number is the ID of the master module of the PromICE unit with the AI option. If you are using multiple units for 16 or 32 bit emulation etc. the other units are automatically programmed to be set in pass through mode.
address	(hex) Address of the virtual UART as seen by the target system, specified as an offset into the ROM space. Since the AI interface occupies four consecutive bytes, this address must be on a quad boundary (i.e. lowest two bits are always zero) within the single ROM through which the interface is accessed. That means for a 16 bit system the address of the AI interface must align with an octal boundary (lowest three bits are always zero) and for a 32 bit system on a 16 byte boundary (lowest nibble is always zero).
baud	(integer) A valid baud rate (1200, 2400, 4800, 9600, 19200, 57600) specifying the baud rate at which the host based debugger will communicate with the target via the PromICE. Baud rate of zero (0) will invoke transparent parallel port mode.
break_char	(hex) character used to break PromICE out of transparent mode. Used with ASCII host-target communication protocols.
int_count	(integer) cumulative number of host interrupts to ignore before PromICE breaks out of transparent mode. Used with binary host-target communication protocols. Some debuggers will toggle DTR line on startup and thus

cause the PromICE to restart and thus break the link, this option allows you to override that.

6.2.4. Default

Default *id* is 0. *address* and *baud-rate* must be specified. The *address* is normalized to map to master ROM module according to word size specification (see word command). Default for *break-character* is no break character, i.e. binary transparency, and default *int_count* is to ignore all interrupts from the host. So for most cases you need only specify the address and the baud-rate.

6.2.5. Description

ailoc is used for setting the AI link into transparent mode as the LoadICE application exits. It is generally used after down-loading a debug monitor and before starting the debugger front end (DFE). The address specified is where the target will have the interface mapped. The address is normalized automatically by LoadICE for mapping to the master ROM module. The baud-rate programs the serial link baud rate to the host. Parallel port may be used for communication with the host, in which case the PromICE bi-directional parallel port protocol must be used.

Once the AI link is in transparent mode, the DFE in the host can communicate with the ROM monitor in the target as if they were connected with a standard serial link.

6.2.6. Notes

In order to run LoadICE again, the transparent link must be broken. LoadICE does that by toggling the DTR or the INIT signal (on parallel port) 4 times in a period of 5 seconds. This allows the PromICE to recognize the LoadICE attempt to break the link. However, on UNIX systems the DTR can not be toggles fast enough (security precaution for letting the modems hang-up the connection!). So the time period must be increased to almost 30 seconds. It can be avoided by removing the toggle code in piunix.c and power-cycling the PromICE unit or pressing the reset switch on the back of the PromICE (Rev 3 only)

6.2.7. Examples

```
ailoc 8 19200
```

This example assumes that the AI link is mapped in the target space at location 0x8 in the ROM space and the serial baud-rate for communication with the host is at 19200 baud.

LoadICE will display appropriate information about the AI link at exit time.

6.3. aiswitch

- control multiple PromICE units through Applied Innovations PBX/MUX. This lets you configure large number of units from a single host port with the units getting a permanent ID assignment (switch port#s).

6.3.1. Command Forms

aiswitch	in `LoadICE.ini` file
-A	as a command line argument to LoadICE

6.3.2. Syntax

{**aiswitch** | **-A**} {*port*}

6.3.3. Use

port	(integer) A valid Applied Innovations PBX/MUX port ID number (0-255) where the PromICE you want to talk to is connected. LoadICE will automatically figure what port the host is connected to.

6.3.4. Default

You must specify a port number.

6.3.5. Description

This command allows LoadICE to communicate with the units connected to the data-pbx. Generally LoadICE will determine PromICE IDs according to how they are daisy-chained. However, when a fixed ID is desired for a given PromICE , then the data-pbx allows a constant addressing based on the port# where the PromICE is attached.

6.3.6. Notes

Custom cables must be made to connect to the data-pbx. The pbx ports must be configured with the following options:

for the port connected to the host: 2 stop bits and pass DSR to RTS
for ports connected to PromICEs : 1 stop bit, toggle RTS on connect

6.3.7. Examples

```
aiswitch 31
```

This will allow LoadICE to communicate with the PromICE connected to port#31 of the pbx. LoadICE will figure out automatically from the pbx prompt where the host is connected.

6.4. alt-fkey#

- allows assigning hot keys to LoadICE or host commands

6.4.1. Command Forms

altf#	in `LoadICE.ini` file

6.4.2. Syntax

{**altf#**=word | "string"}

6.4.3. Use

#	(decimal number) alt-function key number
word/"string"	LoadICE or system shell command. Precede the system commands with a '!' so that LoadICE will escape them to DOS or UNIX shell.

6.4.4. Default

none.

6.4.5. Description

This commands allows you to assign LoadICE commands or system commands to function keys. Thus popular commands can be invoked with a single key stroke. You can for example edit then compile your source file, without exiting out of LoadICE by assigning the command strings to function keys.

6.4.6. Notes

Assign regular word commands directly but enclose strings in double-quotes. System commands must be preceded by a '!'.

6.4.7. Examples

```
altf12=restart
```

This will issue the restart command to LoadICE when ALT key is held down and F12 key is pressed. You can then request LoadICE to restart the link with PromICE after a time-out error.

```
altf1="!edit test.c"
```

when alt-function1 key in invoked LoadICE will automatically execute the system command bringing up the editor to edit test When you exit the editor you will back at the LoadICE prompt.

6.5. baud

- specify baud rate for serial communication between the PromICE and the host. See also *output*

6.5.1. Command Forms

baud	in `LoadICE.ini` file
-b	as a command line argument to LoadICE

6.5.2. Syntax

{**baud** | **-b**} {*baud_rate*}

6.5.3. Use

{baud_rate}	(integer) A valid baud rate (1200, 2400, 4800, 9600, 19200, 57600)

6.5.4. Default

19200 baud

6.5.5. Description

This command specifies the baud_rate for use with a particular serial port specification.

6.5.6. Notes

38400 is not supported, the highest baud_rate normally available to UNIX systems.

6.5.7. Examples

-b 57600
baud 57600

will set the baud rate of the device specified to 57600

6.6. checksum

- perform checksum on ROM and store in PromICE memory

6.6.1. Command Forms

checksum	in `LoadICE.ini` file
-k	as a command line argument to LoadICE
k	in LoadICE dialog mode

6.6.2. Syntax

{**checksum** | **-k** | **k**} [*id* {:}] [*start*] [*end*] [*store*] [*function*] [*sum_size*]

6.6.3. Use

id	(integer) A valid PromICE unit ID number (0-255)
start	(hex) Address in PromICE where checksum should start
end	(hex) Address in PromICE where checksum should stop
function	(character) indicate preferred checksum function (*x* or *a*)*x* indicates exclusive OR function will be performed on data in all locations in selected address range. a indicates addition function will be performed on data in all locations in selected address range.
sum_size	(integer) size of check sum function result to be stored in bits. Must be an integral multiple of 8 and can not be larger than the data bus width emulated by the total number of daisy chained PromICE units.

6.6.4. Default

Must specify the start, end and store arguments. Default for ID is 0, default for function is x and sum_size is 8.

6.6.5. Description

Compute and store a 8, 16 or 32 bit checksum in the ROM. The checksum is performed on a per ROM or for the entire configuration. The checksum is computed inclusive of start and end addresses and the results are stored in the given store address. The checksum is displayed.

If ID is specified then do checksum on that ROM, else checksum all ROMs. If used from dialog mode then checksum the current configuration.

6.6.6. Notes

Checksum is done by uploading the data from the ROM and then stored back. It can take a bit of time for large units.

6.6.7. Examples

```
k 0 fffc fffc a 32
```

compute the checksum on 0 through fffc and store the resulting checksum as a 32 bit number at (fffc-ffff)

6.7. compare

- compares data loaded in the PromICE against files on the host

6.7.1. Command Forms

c	in LoadICE dialog mode

6.7.2. Syntax

{**c**}

6.7.3. Use

c	compare files instead of down-loading them

6.7.4. Description

This command will process the data files and then instead of down-loading the data it will upload the PromICE contents and compare it with what would have down-loaded. It is an explicit verification of file data. Difference are displayed on the screen and you have the option for continued display of differences or canceling further compare.

6.7.5. Notes

If a compare operation fails in part, it is most likely due to overlapping data areas in different files. If that is not the case then it may be overlapping data areas in the same file. If a 'write' line is connected to the target and the target was emulating then it could also have 'written' to location in the emulated space and thus cause compare failures.

If the unit is failing compare right after loading and none of the above are a cause, then there may be problems with the battery backup or memory in general.

6.7.6. Examples

c

compare now.

6.8. config

- display current PromICE configuration

6.8.1. Command Forms

config	in `LoadICE.ini` file
C	(also **config**) in LoadICE dialog mode

6.8.2. Syntax

{**config** | **C**} [link | rom | file | all]

6.8.3. Use

link	display diagnostic information about the communication link (serial or parallel)
rom	display diagnostic information about the ROM configuration
file	display diagnostic information about the list of files loaded into the PromICE
all	display complete diagnostic information about all of the above aspects

6.8.4. Default

all

6.8.5. Description

Display the current configuration data as being used by LoadICE

6.8.6. Notes

Use this command when wondering what word-size or emulation-size and file information is being used by LoadICE

6.8.7. Examples

```
C all
```

Display the entire configuration

6.9. cursor

- show or hide the spinning cursor

6.9.1. Command Forms

cursor	in `LoadICE.ini` file
cursor	in LoadICE dialog mode

6.9.2. Syntax

{cursor}	[#]

6.9.3. Use

#	(0|1) 0 - turn cursor off; 1 - turn cursor on

6.9.4. Default

toggle.

6.9.5. Description

On some systems the spinning cursor may slow down the performance of LoadICE. Certain graphic environments degrade the performance while drawing the spinning cursor.

6.9.6. Notes

WINDOWS running on some high definition CRTs and UNIX workstations have reported hang-up or slow down due to the spinning cursor.

6.9.7. Examples

cursor 0

don't display the spinning cursor

6.10. delay

- change the value of time period when LoadICE is waiting on response from PromICE. The nominal delay is 3 seconds.

6.10.1. Command Forms

delay	in `LoadICE.ini` file
delay	as a command line argument to LoadICE

6.10.2. Syntax

{delay} #

6.10.3. Use

#	(decimal number) number of seconds to delay. 0 - no delay, wait indefinitely.

6.10.4. Default

none.

6.10.5. Description

This command allows control over delay and keeps LoadICE from timing out. Must be used at your own discretion.

6.10.6. Notes

While executing certain commands where response may take variable amount of time, LoadICE internally shuts off the delay, such as when executing 'test' command to test PromICE memory.

6.10.7. Examples

delay 30

wait 30 seconds before timing out.

6.11. dialog

-must go into LoadICE dialog mode

6.11.1. Command Forms

dialog	in `LoadICE.ini` file
-d	as a command line argument to LoadICE

6.11.2. Syntax

{dialog | -d}

6.11.3. Description

This command will force LoadICE to enter interactive dialog mode.

6.11.4. Notes

Normally LoadICE will process and down-load user data files according to the configuration specified. However, if no data files are specified LoadICE will go into an interactive 'dialog' mode also. In this mode any commands, files and operations can be specified and done interactively.

6.11.5. Examples

```
-d
```

6.12. display

- change the output level of LoadICE

6.12.1. Command Forms

display	in `LoadICE.ini` file
-D	as a command line argument to LoadICE
D	in LoadICE dialog mode

6.12.2. Syntax

{**display** | **-D** | **D**} [*level*]

6.12.3. Use

level (hex) A valid number (0 - FF) indicating the level of diagnostic display desired. Bits are as follows:

0x80	- display prompts
0x40	- display progress
0x20	- display command parser data
0x10	- display config data, disk i/o and buffer transfer
0x08	- display hex record processing
0x04	- display commands and responses to/from PromICE (abbreviated)
0x02	- display full commands and responses to/from PromICE
0x01	- display actual data bytes going over the link

6.12.4. Default

0xC0 - only displays main prompt and command progress and results

6.12.5. Description

This command allows the display of extra information about processing a command.

6.12.6. Notes

Setting display level to 0x00 will shut off everything but command results. Setting level to 0xFF can cause data overflow over the serial link.

6.12.7. Examples

-D FE

display everything but the actual data going over the link.

6.13. dump

- display PromICE memory contents on the screen

6.13.1. Command Forms

d	in LoadICE dialog mode

6.13.2. Syntax

{**d** | **dump**} [[*id* {:}] [*start*] [*end*]]

6.13.3. Use

id	(integer) A valid PromICE unit ID number (0-255)
start	(hex) First PromICE address to display
end	(hex) Last PromICE address to display

6.13.4. Default

ID is zero or use the current configuration. Default for start address is 0 and end start+64 (or the last valid address in ROM)

6.13.5. Description

Dump command will display the PromICE data in hex and ASCII with 16 bytes per line. Data is displayed in the configuration that is active, i.e. you can view data as 8, 16, 32 (up to 64) bits at a time if the configuration is set so. A <cr> will simply repeat the command with next range of arguments.

6.13.6. Notes

none

6.13.7. Examples

```
d 0 ff
```
dump data from 0x0 to 0xff address range.

```
d 1:fe ff
```
dump data from unit ID-1 from 0xfe to 0xff (two bytes).

6.14. edit

- modify PromICE emulation memory, i.e edit ROM space

6.14.1. Command Forms

edit	in `LoadICE.ini` file
-e	as a command line argument to LoadICE
edit	(also **e**) in LoadICE dialog mode

6.14.2. Syntax

{**edit** | **-e** | **e**} [[*id* {:}] [*address*] [[*value*] ...]]

6.14.3. Use

id	(integer) A valid PromICE unit ID number (0-255)
address	(hex) Address in PromICE at which to start memory modification.
value	(hex) Value or list of values to be put in memory starting at *address*. Width of individual values can be any integral multiple of 8, but less than or equal to the data bus width emulated by the total number of daisy chained PromICE units.

6.14.4. Default

start editing at address 0 in the current configuration.

6.14.5. Description

Edit can edit one or more data bytes in one or more ROMs. Edit will edit words specified by current configuration. When edit is in the ini file or the command line then bytes are edited after loading any files that are specified. In dialog mode edit will enter interactive mode if no data values are specified otherwise it will simply edit the locations as specified.

6.14.6. Notes

The use of edit in the ini file or command line is intended for patching ROM data after the files have been down-loaded

6.14.7. Examples

edit 1:500 ab cd

will edit location 0x500 and 0x501 in unit 1 with values 0xab and 0xcd

6.15. exit

- exit LoadICE when in dialog mode

6.15.1. Command Forms

exit	[also **x** or **quit**] in LoadICE dialog mode

6.15.2. Syntax

{exit | x | quit}

6.15.3. Use

x	exit LoadICE

6.15.4. Default

none.

6.15.5. Description

All Done, exit LoadICE

6.15.6. Notes

Only .

6.15.7. Examples

quit

all done, exit

6.16. fast

- allows longer strobes on the parallel port on host with high performance processor and also when parallel interface is on the mother-board.

6.16.1. Command Forms

fast in `LoadICE.ini` file

6.16.2. Syntax

{fast} [#]

6.16.3. Use

#	To lengthen the parallel port strobes, by a factor of #

6.16.4. Default

as fast as it goes.

6.16.5. Description

This command allows you to lengthen the strobe by varying amount.

6.16.6. Notes

If the parallel port transfer hang and fail, use this command to lengthen the strobe . The strobe is lengthen by inserting a dummy loop between the assertion and removal of various strobe signals. The loop factor is multiplied by the user given number.

6.16.7. Examples

```
fast    5
```

says use 5 times longer strobes.

6.17. file

- specify hex record file information for down-loading or compare operation

6.17.1. Command Forms

file	in `LoadICE.ini` file
<no arg>	as a command line argument to LoadICE
file	in LoadICE dialog mode

6.17.2. Syntax

{**file** | <no arg>} [[*file_name*] [*link_address*] [*id* {:}] [*ROM_offset*] [*data_width*] [[*byte_n*] ...]] [(a1,a2)]

6.17.3. Use

file_name	(string) Name of hex record file to be loaded.
link_address	(hex) Absolute starting address of the file *file_name* , produced by the linker, that is stored incrementally in each hex record of the file *file_name.* This address is normally used by hex record decoding programs to determine where in memory to place the data.
id	(integer) A valid PromICE unit ID number (0-255)
ROM_offset	(hex) An offset value within the PromICE emulated ROM space. The start of the PromICE emulated ROM space is assigned to be offset address 0. This offset is used to determine the location within the PromICE emulated memory where the *link_address* should be assigned.
data_width	(integer) Data bus width associated with width of data objects in file *file_name.* Must be an integral multiple of 8 and can not be larger than the data bus width emulated by the total number of daisy chained PromICE units.
byte_n	(integer) nth byte in a data bus that has a width that is an integral multiple of 8 bits. Bytes are assigned to successive PromICE units (0 - 255) based on the selected order of *byte_n* specifications.
(a1,a2)	(hex addresses) transfer data only from address a1 through address a2, inclusive. This specification allows partial loading of data files.

6.17.4. Default

load the file in current configuration starting at address zero.

6.17.5. Description

This specification allows you to specify the file for down-loading to the PromICE. The file type can be specified if it is binary, else it is assumed to be hex. The command allows for provision to specify whether the file data needs to mapped or relocated within the ROM space. Word width of the data in the file as well as byte order within the word can be specified. Finally partial loading of data can be specified.

6.17.6. Notes

Only one set of partial loading address can be specified.

6.17.7. Examples

```
file=myfile.hex 400000=0 16 1 0
```

says load myfile.hex with 0x400000 in the file to go to 0x0 in the ROM and that the file contains 16 bit data with the first byte (even) to go to PromICE unit ID-1 and the second byte (odd) to go to PromICE unit ID-0.

6.18. fill

- fill PromICE memory with repeating data (fill pattern)

6.18.1. Command Forms

fill	(also **ffill**) in `LoadICE.ini` file
-f	(also **-ff**) as a command line argument to LoadICE
f	in LoadICE dialog mode

6.18.2. Syntax

{**fill** \| **f** \| **-f**} [[*id* {:}] [*start*] [*end*] [*data*]]

6.18.3. Use

id	(integer) A valid PromICE unit ID number (0-255)
start	(hex) First PromICE address to fill
end	(hex) Last PromICE address to fill
data	(hex) Data value with which to fill range. Width of data value can be any integral multiple of 8, but less than or equal to the data bus width emulated by the total number of daisy chained PromICE units.

6.18.4. Default

Fill all of the ROM space with the default fill character (0x0FF)

6.18.5. Description

Fill is used for filling all or part of ROM space with fill character. Individual ROM or entire configurations can be filled. When the configuration or multibyte (16-64 bits) then fill character is also 16, 32 or 64 bits long (64 bit data is specified as two 32 bit items)

6.18.6. Notes

filling is done prior to loading the file data.

6.18.7. Examples

f 200 300 ab
fill from 0x200 to 0x300 with data value 0xab

6.19. find

- find binary data patterns in the PromICE memory.

6.19.1. Command Forms

find | **F**	in LoadICE dialog mode

6.19.2. Syntax

{**find** | **F**} [[*id* {:}] {*start*} {*end*} {*size* } {*data bytes*}

Use:

id	(integer) A valid PromICE unit ID number (0-255)
start	(hex) First PromICE address to start search
end	(hex) Last PromICE address to search
size	(integer) number of bytes to find
data bytes	(hex) Data values to look for. The sequence of bytes are looked for in the specified address range.

6.19.3. Default

all information except for the ID must be specified.

6.19.4. Description

Find allows you to search the PromICE memory for binary data patterns. You can search individual ROMs or ROM configurations.

6.19.5. Notes

The limit on number of data bytes to find is ?

6.19.6. Examples

```
F 0 1ffff 4 de ad fe ed
```

looks for the pattern 0xdeadfeed in a 128kB ROM space (0x-0x1ffff)

6.20. fkey#

- allows assigning hot keys to LoadICE or host commands

6.20.1. Command Forms

f#	in `LoadICE.ini` file

6.20.2. Syntax

{f#=word | altf#="string"}

6.20.3. Use

#	(decimal number) function key number
word|"string"	LoadICE or system shell command. Precede the system commands with a '!' so that LoadICE will escape them to DOS or UNIX shell.

6.20.4. Default

none.

6.20.5. Description

This commands allows you to assign LoadICE commands or system commands to function keys. Thus popular commands can be invoked with a single key stroke. You can for example edit then compile your source file, without exiting out of LoadICE by assigning the command strings to function keys.

6.20.6. Notes

Assign regular word commands directly but enclose strings in double-quotes. System commands must be preceded by a '!'.

6.20.7. Examples

f12=restart

This will issue the restart command to LoadICE when F12 key is pressed. You can then request LoadICE to restart the link with PromICE after a time-out error.

f1="!edit test.c"

when function1 key in invoked LoadICE will automatically execute the system command bringing up the editor to edit test When you exit the editor you will back at the LoadICE prompt.

6.21. go

- instruct PromICE to go into emulation mode

6.21.1. Command Forms

go	in LoadICE dialog mode

6.21.2. Syntax

{**go**}

6.21.3. Description

Allows you to start emulation from the dialog mode. The status of the PromICE emulation state is reflected in the load light on the front panel. Upon the go command the load light should go out (unless the target is not powered up)

6.21.4. Notes

You can also use the [ESC] key to switch from load to emulate and vice-versa.

6.21.5. Examples

```
go
```

PromICE will go into emulation mode.

6.22. help

- obtain help about a LoadICE command

6.22.1. Command Forms

?	on the command line as the only argument
help	in LoadICE dialog mode

6.22.2. Syntax

{**help** | **?**} [*command*]

6.22.3. Use

command	(string) Any valid LoadICE command

6.22.4. Default

display the list of command on which help is available

6.22.5. Description

help will give you on-line help for any command. When invoked with '?' as the only argument on the command line, it will give you help on all the items that can be specified in the ini file. When invoked as '?' or 'help' in dialog mode, it will display a list of all the commands available. Further help then can be obtained on each individual command.

6.22.6. Notes

if there is any help file distributed with LoadICE then it should be in the current directory or the proper environment variable must be set to indicate the location of the help file

6.22.7. Examples

help find

in dialog mode this will give information on how to use the 'find' command.

6.23. high

- let PromICE send the response back on the serial link at full rate

6.23.1. Command Forms

high	in `LoadICE.ini` file
-h	In LoadICE command line

6.23.2. Syntax

{high -h}

6.23.3. Use

(high I h	full tilt transmission

6.23.4. Default

on Rev 2 units the default if to simulate 9600 baud rate
on Rev 3 units the default is to send at full rate.

6.23.5. Description

On older hosts the data coming back at full baud rate of 567600 etc. would overflow in the host resulting in LoadICE timeouts. This option lets you change the transmission to full rate on faster hosts (16+ MHz 386, 486 etc.)

6.23.6. Notes

Remove this statement if serial link seems to hang on your host .

6.23.7. Examples

high

transmit response at full baud rate

6.24. hso

- program the interrupt signal to the target (on PromICE back panel).

6.24.1. Command Forms

hso	in `LoadICE.ini` file
-l	in LoadICE dialog mode

6.24.2. Syntax

{hso | -l} {#}

6.24.3. Use

#	this number specifies the polarity of the interrupt signal as well as allows you to toggle it when LoadICE connects with the PromICE 0 - interrupt is low asserted, it is raised at this time 1 - interrupts is high asserted, it is lowered at this time 2 - interrupt is high asserted and at startup lower-raise-lower the interrupt line. 5 - interrupt is low asserted and at startup raise-lower-raise the interrupt line A - interrupt is high asserted and at start up lower-raise-lower the interrupt line, but raise it for 1 second. D - interrupt is low asserted and at startup raise-lower-raise the interrupt line, but lower it for 1 second.

6.24.4. Default

signal is low asserted.

6.24.5. Description

By using this command you can define the interrupt signal on the back of the PromICE panel to be high or low asserted. Besides that you can also specify if to toggle this signal when LoadICE first connects with the PromICE. This allows you to alert the target that LoadICE is talking to the PromICE.

6.24.6. Notes

On Rev-3 unit both polarity of the signal are available at the back panel.

6.24.7. Examples

hso 1

defines the interrupt to the target high asserted

6.25. image

- specify binary file information for down-load or compare operation

6.25.1. Command Forms

image	in `LoadICE.ini` file
-i	as a command line argument to LoadICE
image	(also **i**) in LoadICE dialog mode

6.25.2. Syntax

{**image** | **-i** | **i**} [[*file_name*] [*skip count*] [*id* {:}] [*ROM_offset*] [*data_width*] [[*byte_n*] ...]

6.25.3. Use

file_name	(string) Name of binary file to be loaded etc.
skip_count	(hex) any data to be skipped from the front of the file. Occasionally there is a 14 byte header that may need to be skipped.
id	(integer) A valid PromICE unit ID number (0-255)
ROM_offset	(hex) An offset value within the PromICE emulated ROM space. The start of the PromICE emulated ROM space is assigned to be offset address 0. This offset is used to determine the location within the PromICE emulated memory where the first data byte of file *file_name* should be assigned.
data_width	(integer) Data bus width associated with width of data objects in file *file_name.* Must be an integral multiple of 8 and can not be larger than the data bus width emulated by the total number of daisy chained PromICE units.
byte_n	(integer) nth byte in a data bus that has a width that is an integral multiple of 8 bits. Bytes are assigned to successive PromICE units (0 - 255) based on the selected order of *byte_n* specifications.

6.25.4. Default

load the file in its entirety starting at 0 in the PromICE configuration.

6.25.5. Description

This command allows you to specify a binary data file to be down-loaded to the PromICE. Furthermore you can specify where the file should be loaded exactly and if any bytes need to be skipped from the start of the file. The word size and byte order in the file can also be specified.

6.25.6. Notes

When loading multiple binary files you must specify where they go in the emulated space or else they all get loaded on top of each other.

6.25.7. Examples

```
image=myfile.bin 0=10000 16 0 1
```

load the file 'myfile.bin' at location 0x10000. The file contains 16 bit data and the first byte is to be loaded in unit-0 and the second byte in unit-1.

6.26. load

- file loading associated with LoadICE dialog mode.

6.26.1. Command Forms

load	in `LoadICE.ini` file
-l	as a command line argument to LoadICE
l	in LoadICE dialog mode

6.26.2. Syntax

{**load** | **-l** | **l**} [filename etc.]

6.26.3. Use

	when specified in the ini file or the command line, then the instruction to LoadICE is to down-load the file before entering the dialog mode.
filename	if you wish to load a specific file (in dialog mode only).

6.26.4. Default

load the pre-specified file list

6.26.5. Description

'load' allows you to down-load your files on demand. All the specified files are processed and down-loaded.

When 'load' appears in ini file or '-l' on the command line, it implies that LoadICE should down-load the files before entering the dialog mode.

6.26.6. Notes

'dialog' mode is entered only if LoadICE is instructed to do so with 'dialog' in ini file or '-d' on the command line. 'load' allows you to first load the file when this is the case.

6.26.7. Examples

l

load the files now

6.27. map

- control the display of address range being loaded during file processing

6.27.1. Command Forms

map	in `LoadICE.ini` file
map	in LoadICE dialog mode

6.27.2. Syntax

{map} [#]

6.27.3. Use

#	0 - turn display of map information off
	1 - turn display of map information on

6.27.4. Default

on.

6.27.5. Description

LoadICE will display the range of addresses to which data is being loaded. This command lets you turn that feature off

6.27.6. Notes

When the data file has hex records that have real fragmented data then there may be a good stream of address ranges displayed. You can use this command to turn that display off .

6.27.7. Examples

map 0

don't display the map information

6.28. move

- copy bytes of memory in PromICE

6.28.1. Command Forms

move	in `LoadICE.ini` file
m	in LoadICE dialog mode

6.28.2. Syntax

{**move** | **m**} [[*id* {:}] [*start*] [*end*] [*destination*]]

6.28.3. Use

id	(integer) A valid PromICE unit ID number (0-255)
start	(hex) starting address of source data block from where data is to be copied
end	(hex) ending address of source data block from where data is to be copied
destination	(hex) starting address of destination location to where source data block is to be copied.

6.28.4. Default

All three parameters must be specified.

6.28.5. Description

Allows you to move data (copy) from and to within PromICE memory.

6.28.6. Notes

Data is up-loaded and then down-loaded to new location. However, it is uploaded in 4096 bytes chunks, therefore overlapping moves can have interesting results.

6.28.7. Examples

```
m 100 120 300
```

move data from 0x100 to 0x120 to 0x300 (to 0x320)

6.29. noaddrerr

- ignore address out of range errors during file loading

6.29.1. Command Forms

noaddrerr	in `LoadICE.ini` file
-z	as a command line argument to LoadICE
noaddrerr	(also **z**) in LoadICE dialog mode

6.29.2. Syntax

{**noaddrerr** | **-z** | **z**} [1]

6.29.3. Use

1	view records in hex record file that have address values out of range

6.29.4. Default

Stop down-loading further data when address error is encountered.

6.29.5. Description

If attempt is made to load any data byte to an address that is not within the specified emulated space, i.e. the address is larger than the ROM size then LoadICE will stop processing further data and report this error message. You can specify to skip offending records and keep going with processing.

6.29.6. Notes

Generally if you have properly mapped your data files then this error is caused by initialized data intended for the RAM. This option will allow you to skip these records.

6.29.7. Examples

noadddrerr 1

skip records that have address out of range but display then nonetheless.

6.30. noautorecovery

- do not reset corrupted link with PromICE after time-out error.

6.30.1. Command Forms

noautorecovery in `LoadICE.ini` file

6.30.2. Syntax

{**noautorecovery** | **auto**}

6.30.3. Description

If LoadICE times out while communicating with PromICE it will reestablish the link. This is called auto-recovery. To disable this you can use this command.

6.30.4. Notes

You might want to disable this when using LoadICE in batch mode, this way it won't keep trying to recover the link over and over if some strange error has occurred (like no power on the PromICE)

6.30.5. Examples

noautorecovery

do not try to re-establish the link after any time-out.

6.31. nochecksum

- do not process record checksums in hex record files

6.31.1. Command Forms

nochecksum	in `LoadICE.ini` file
-x	as a command line argument to LoadICE

6.31.2. Syntax

{**nochecksum** | **-x**} [1]

6.31.3. Use

1	(integer) allows you to look at the offending records

6.31.4. Default

Stop processing further data when checksum error is encountered.

6.31.5. Description

Normally LoadICE will stop processing further data when checksum error is encountered. This option allows you to continue processing the data including the offending record. In fact this option disable computation and comparison of checksum stored in the record.

6.31.6. Notes

If you really are getting checksum errors in your file then something is terribly wrong with your data file. However, this option allows you to manually patch your hex file without worrying about the checksum and then down-load the modified file.

6.31.7. Examples

nochecksum

do not process the checksum in the hex records.

6.32. nolight

- turn off the blinking RUN light feature of newer versions of PromICE micro-code

6.32.1. Command Forms

nolight	in `LoadICE.ini` file
-N	In LoadICE command line

6.32.2. Syntax

{nolight | -N}

6.32.3. Use

nolight | -N	leave the light alone, i.e. ON

6.32.4. Default

blink to indicate different states of the PromICE.

6.32.5. Description

On newer versions of PromICE micro-code the RUN light shows the various states the PromICE is in

ON - unit is not connected to any host
blink once every three seconds - unit is connected to a host
blink rapidly - AI circuit is armed but not accessed
blink twice every three seconds - AI is in use or has been accessed by the target.

6.32.6. Notes

When using the AI and setting up the debugger to work, the RUN light state can be handy for knowing if the AI is armed and if and when the target starts accessing it.

6.32.7. Examples

nolight

don't mess with the RUN light

6.33. notimer

- turn off the internal timer interrupt in the PromICE

6.33.1. Command Forms

notimer	in `LoadICE.ini` file
-T	In LoadICE command line

6.33.2. Syntax

{notimer | -T}

6.33.3. Use

notimer | -T	turn off the use of the timer (nolight is implied)

6.33.4. Default

timer is on and interrupts every 8.9 milli-seconds.

6.33.5. Description

This command is useful only when using multiple PromICEs with synchronized targets. Since the timer interrupts are asynchronous the different PromICE units can loose lock-step operation.

6.33.6. Notes

When using the AI circuit on fault tolerant systems debugging this option help keep the units synchronized .

If the 'reset' command is used then the timer is turned back on to drive the reset signal. It is turned back off if this option is in effect.

6.33.7. Examples

notimer

no timer no lights....

6.34. noverify

- do not request response from PromICE during download

6.34.1. Command Forms

noverify	in `LoadICE.ini` file
-v	as a command line argument to LoadICE

6.34.2. Syntax

{**noverify** | **-v**}

6.34.3. Description

Normally LoadICE will compute a checksum for each block of data sent to the PromICE. PromICE will return the checksum it computes and the two are compared. If they fail to match then some transmission error occurred.

Noverify option allows you to skip this comparison and speed up the down-loading process.

6.34.4. Notes

We recommend that once your connection is set up and you can reliably down-load data, you may consider turning off this option.

Also when using the Turbo mode of down-loading with the parallel port, this option is required to achieve very fast down-load speed.

To verify down-loaded data in its entirety use the 'compare' command

6.34.5. Examples

noverify

do not verify down-loaded data as it is loaded

6.35. number

- specify number of PromICE modules

6.35.1. Command Forms

number	in `LoadICE.ini` file
-n	as a command line argument to LoadICE

6.35.2. Syntax

{**number** | **-n**} [*total_modules*]

6.35.3. Use

total_modules	(integer) Total number of PromICE master and slave modules daisy chained together (0-255)

6.35.4. Default

number is 1

6.35.5. Description

This option is needed only for UNIX systems. This allows the LoadICE application to send out enough auto-baud character to ensure that all the units would have gone through the auto-baud sequence and some of the characters would be back in the system before a read is issued. If a read were to be issued before any characters have arrived, then the read will sleep (forever).

6.35.6. Notes

Even though the serial line is opened with option to not wait on the read, i.e. return if nothing to read, the UNIX manual states that on a communication line the first read will block if no characters are available to be read. This command allows us to get around this problem.

6.35.7. Examples

number 3

Allows enough auto-baud characters to be transmitted for three units to auto-baud completely.

6.36. output

- specify serial output device for connection with PromICE

6.36.1. Command Forms

output	in `LoadICE.ini` file
-o	as a command line argument to LoadICE

6.36.2. Syntax

{**output** | **-o**} [[*dev_name*] [*address*]]

6.36.3. Use

dev_name	(string) Standard operating system name for the desired serial port. Paths may be required.
address	(hex) On the PC or compatibles, LoadICE will lookup the port address in the BIOS built table in low memory. If you have a non standard serial port, then you may explicitly give its address to LoadICE, in which case LoadICE will not search for it in low memory.

6.36.4. Default

COM1 for PC
/dev/ttyb for UNIX
modem for Macintosh

6.36.5. Description

This command specifies the serial link for the LoadICE to use to talk to PromICE. Optionally the port address may be specified for PC systems.

6.36.6. Notes

When using the AI in transparent mode (ailoc command) this option specifies the link for use in host communication after transparent mode is established.

6.36.7. Examples

-o com3

use COM3

6.37. parallel

- specify parallel port in unidirectional mode

6.37.1. Command Forms

parallel	in `LoadICE.ini` file
-p	as a command line argument to LoadICE

6.37.2. Syntax

{**parallel** | **-p**} [[*dev_name*] [*address*]]

6.37.3. Use

dev_name	(string) Standard operating system name for the desired parallel port. Paths may be required.
address	(hex) On the PC or compatibles, LoadICE will lookup the port address in the BIOS built table in low memory. If you have a non standard serial port, then you may explicitly give its address to LoadICE, in which case LoadICE will not search for it in low memory.

6.37.4. Default

LPT1 - for PCs only.

6.37.5. Description

This command will use the parallel link for down-load only. Serial link information also must be specified. Both links are used to communicate with the PromICE.

6.37.6. Notes

This specification is used mostly when multiple units are daisy chained on the parallel port. This allows fast loading of the units. Serial link is used for all commands and bi-directional communication. Parallel link is used for down-loading the data files

6.37.7. Examples

parallel lpt2

use LPT2 for down-loading data

6.38. ppmode

- specify parallel port operating mode

6.38.1. Command Forms

ppmode	in `LoadICE.ini` file
-P	as a command line argument to LoadICE

6.38.2. Syntax

{**ppmode** | **-P**} {*mode*}

6.38.3. Use

mode	(number) 0 - specifies standard mode compatible with older LoadICE and PromICE. Requires parallel b_ack handshake after each transfer 1 - specifies the fast parallel transfer mode. The b_ack is done only if and when a response is to be sent (i.e. parallel port is bi-directional) 2 - specifies Turbo transfer mode. LoadICE uses inline code to transfer data as fast as possible over the parallel port. Verification and timeouts are turned off.

6.38.4. Default

mode 0 or 1 is automatically selected. You can force other modes.

6.38.5. Description

You can use this command to invoke Turbo transfer mode for the PromICE parallel port, or slow down the transfer altogether by using mode 0.

6.38.6. Notes

Make sure that parallel port is working reliably before engaging Turbo mode. Let LoadICE automatically determine whether you have an old style parallel port or fast parallel port. The difference is only in the micro-code of the PromICE unit.

6.38.7. Examples

ppomode 2

Run the parallel link in turbo mode.

6.39. pponly

- specify parallel port in bi-directional mode

6.39.1. Command Forms

pponly	in `LoadICE.ini` file
-q	as a command line argument to LoadICE

6.39.2. Syntax

{**pponly** | **-q**} [[*dev_name*] [*address*]]

6.39.3. Use

dev_name	(string) Standard operating system name for the desired parallel port. Paths may be required.
address	(hex)On the PC or compatibles, LoadICE will lookup the port address in the BIOS built table in low memory. If you have a non standard serial port, then you may explicitly give its address to LoadICE, in which case LoadICE will not search for it in low memory.

6.39.4. Default

LPT1

6.39.5. Description

If only one PromICE unit is attached, then the parallel port can be used in bi-directional mode. Specially modified parallel port protocol is used to talk to the PromICE.

6.39.6. Notes

If you have any security devices on the parallel port (dongles!) then you must remove them. Also sometime the A/B switches also will not work.

6.39.7. Examples

pponly lpt2

use LPT2 as the bi-directional parallel port to talk to the PromICE.

6.40. reset

- specify duration of target reset signal (RST)

6.40.1. Command Forms

reset	in `LoadICE.ini` file
-r	as a command line argument to LoadICE

6.40.2. Syntax

{**reset** | **-r**} [*milliseconds*]

6.40.3. Use

milliseconds	(integer) Number of milliseconds (0-3000) to activate target reset signal (RST) from PromICE during the next reset event.

6.40.4. Default

500 milliseconds

6.40.5. Description

Normally LoadICE/PromICE will operate in auto-reset mode. This means that whenever the PromICE is in 'load' mode the target reset signal is asserted and whenever PromICE is in 'emulate' mode then the reset is released (goes tri-state). However, reset may be asserted on command by using this command.

6.40.6. Notes

'reset' with time of 0 will disable auto-reset and then the target must be reset by giving it an explicit reset command.

6.40.7. Examples

r 500

reset the target with a 500 milli-second pulse.

6.41. restart

- restart the LoadICE/PromICE interface

6.41.1. Command Forms

restart	in LoadICE dialog mode

6.41.2. Syntax

{restart}	restart the link

6.41.3. Use

restart	to restart the link

6.41.4. Default

none.

6.41.5. Description

This command lets to reestablish the link with PromICE without restarting the LoadICE application.

6.41.6. Notes

If you have reset or powercycled the PromICE then you can reestablish the link by using this command. If LoadICE has timed out waiting for PromICE then this command can be used as well. However, LoadICE will try to recover the link automatically unless the 'noautorecovery' command was specified. .

6.41.7. Examples

restart

regardless of what state PromICE was in, restart the link with PromICE

6.42. rom

- specify ROM emulation memory size

6.42.1. Command Forms

rom	in `LoadICE.ini` file
-r	as a command line argument to LoadICE
R	in LoadICE dialog mode

6.42.2. Syntax

{**rom** | **-r** | **R**} [[*id* {:}] [[[*size*] [k]] | *part_number*]]

6.42.3. Use

id	(integer) A valid PromICE unit ID number (0-255)
size	(integer) Size of emulated memory, measured in bytes. Specified number must be an integral power of 2 and cannot be larger than the total memory emulated by the total number of daisy chained PromICE units.
k	Indicates *size* is measured in kilobytes.
part_number	(string) JEDEC standard ROM part number. Number must begin with a 27.

6.42.4. Default

The emulation size is same as the amount of memory in a given PromICE module.

6.42.5. Description

This command allows you to specify the size of ROM you wish to emulate. It must be less than or equal to the amount of memory in the PromICE module.

6.42.6. Notes

see 'socket' command if your target is wired for a socket larger than the size of ROM you are emulating. This will usually be the case only for 1, 2 and 4 Mb ROMs. A socket wired for 4Mb can be used with a 1 or 2 or 4 Mb ROMs without any jumper changes.

6.42.7. Examples

rom=27010 | rom=131072 | rom=128k

They all specify a 1mbit ROM to be emulated

6.43. rv

- engage the ROMview interface of LoadICE

6.43.1. Command Forms

rv	in LoadICE dialog mode

6.43.2. Syntax

{**rv** } {address} [async]

6.43.3. Use

address	engage ROMview interface with the target using the given address as the interface address as seen by the target.
async	set to 1 if using async mode of transfer.

6.43.4. Default

address must be specified and the default for *async* is 0.

6.43.5. Description

This command will request that LoadICE establish communication with a monitor down-loaded on the target system. This monitor is generally supplied by Grammar Engine, you may have one of your own. If LoadICE is successful in setting up the link with the monitor then the prompt will change to ROMview:

Once you are in ROMview you can still do most of the LoadICE commands, except that they apply to the target's address space. So you can dump, edit, fill, find, search, and download and upload target's memory. This interface can be expanded to implement custom debuggers.

6.43.6. Notes

You can change the display level (D command) to FE before using rv command. This will let you see the commands going to the target and help you diagnose the problem if link is not working.

6.43.7. Examples

```
rv 200
```

Establish the link with interface address at 200

6.44. save

- save PromICE memory contents to a file on the host

6.44.1. Command Forms

save	in LoadICE dialog mode

6.44.2. Syntax

{**save**} {*file_name*} [[*id* {:}] [*start*] [*end*]]

6.44.3. Use

file_name	(string) Name of host file in which PromICE memory contents are to be saved. Path names may be included.
id	(integer) A valid PromICE unit ID number (0-255)
start	(hex) If a portion of PromICE memory is to be saved, this specifies the starting address of that portion. Address must be an offset value within the PromICE emulated ROM space. The start of the PromICE emulated ROM space is assumed to be address 0. Data bus width multiples (8,16,32 bit) do not need to be taken into account.
end	(hex) If a portion of PromICE memory is to be saved, this specifies the ending address of that portion. Address must be an offset value within the PromICE emulated ROM space. The start of the PromICE emulated ROM space is assumed to be address 0. Data bus width multiples (8,16,32 bit) do not need to be taken into account.

6.44.4. Default

save the whole ROM or the current ROM configuration

6.44.5. Description

The contents of the PromICE memory are uploaded and saved to a binary file

6.44.6. Examples

```
save newfile 0:100 3fff
```

save the data from unit-0 from 0x100 to 0x3fff to a file called 'newfile' on the host

6.45. search

- search PromICE memory for ASCII data pattern

6.45.1. Command Forms

search (also **S**) in LoadICE dialog mode

6.45.2. Syntax

{**search** | **S**} [[*start*] [*end*]] {*pattern*}

6.45.3. Use

start	(hex) If a portion of PromICE memory is to be searched, this specifies the starting address of that portion. Address must be an offset value within the PromICE emulated ROM space. The start of the PromICE emulated ROM space is assumed to be address 0. Data bus width multiples (8,16,32 bit) do not need to be taken into account.
end	(hex) If a portion of PromICE memory is to be searched, this specifies the ending address of that portion. Address must be an offset value within the PromICE emulated ROM space. The start of the PromICE emulated ROM space is assumed to be address 0. Data bus width multiples (8,16,32 bit) do not need to be taken into account.
pattern	(string) Data pattern to be searched. Please enclose the string in "s

6.45.4. Default

search the whole of current configuration

6.45.5. Description

This command allows you to search the PromICE memory for an ASCII string.

6.45.6. Notes

The entire specified range is searched even if there are multiple occurrence of the string.

6.45.7. Examples

```
S 0 1000 "Enter new value:"
```

search from 0x0 to 0x1000 for the string given

6.46. serial#

- display PromICE mirco-code serial #

6.46.1. Command Forms

serial	in LoadICE dialog mode

6.46.2. Syntax

{serial}

6.46.3. Use

serial	display the serial number in the micro-code

6.46.4. Default

serial number is 0.

6.46.5. Description

The micro-code has place for a 32 bit serial number that can uniquely identify a particular unit. This command shows you what that number is. Most units have the number 0x00000000 as the serial #.

6.46.6. Notes

This allows PromICE to be bundled with debuggers that will only work on given unit. . If you would like some specific serial number in the units you want, contact Grammar Engine.

6.46.7. Examples

serial

report serial #

6.47. slow

- slow down the response transmission rate from the PromICE to the host

6.47.1. Command Forms

slow	in `LoadICE.ini` file
-S	In LoadICE command line

6.47.2. Syntax

{slow | S}

6.47.3. Use

slow | -S	slow down the serial rate as if it were 9600 baud

6.47.4. Default

on Rev2 - slow
on Rev3- full rate.

6.47.5. Description

Allows you to slow down the data coming back from a PromICE when the PromICE is connected to a slow host (16MHz of less CPUs)

6.47.6. Notes

If LoadICE hangs when communicating with the PromICE over the serial link, try this option to avoid data over runs. .

6.47.7. Examples

slow

send back responses at simulated 9600 baud rate

6.48. socket

- specify the capacity of the ROM socket on the target

6.48.1. Command Forms

socket	in `LoadICE.ini` file

6.48.2. Syntax

{socket}	{romsize}

6.48.3. Use

romsize	(part#) specified as 27040 etc.
	(K bytes) specified as 512K
	(decimal number) specified as 524288

6.48.4. Default

same as rom size.

6.48.5. Description

If your target is wired to accept a range of ROM devices then this statements allows you to specify the largest size the target can address as it is currently configure. Then if you are emulating a size smaller than the socket size, LoadICE will appropriate masks to make sure that both the host (LoadICE) and the target 'see' the same ROM space.

6.48.6. Notes

The 27010, 27020 and the 27040 parts (128k; 256k and 512k bytes) can be plugged into a socket wired for the 27040 part without changing any jumpers. This will cause a problem specially when AI circuit is in use. This command allows LoadICE to use appropriate mask for unused address lines for such case.

6.48.7. Examples

```
socket=27040
rom=27010
```

lets you reliable emulate the 1MBit ROM in a socket wired for the 4MBit ROM.

6.49. status

- display the target's status

6.49.1. Command Forms

status (also ' ' (space)) in LoadICE dialog mode

6.49.2. Syntax

{**status** | ' ')

6.49.3. Use

hit the space bar and <CR> to display status at the LoadICE: prompt

6.49.4. Default

none.

6.49.5. Description

PromICE keeps track of the target power state as well as it tries to determine if the target is active by monitoring the data bus. If the data bus is changing then the target is assumed to be alive.

6.49.6. Notes

Only newer versions of the micro-code support these features.

6.49.7. Examples

```
status

Power is on
Target is running |halted
```

6.50. stop

- stop PromICE emulation mode

6.50.1. Command Forms

stop (also **<ESC>** key) in LoadICE dialog mode

6.50.2. Syntax

{**stop** | **<ESC>**}

6.50.3. Description

Force the PromICE units to stop emulating and go in to load mode. If auto-reset is active then reset signal is asserted.

6.50.4. Notes

crashes the target when you do that if target is executing out of the emulated space, unless reset is attached to the target.

6.50.5. Examples

stop

load light should come on and you can do commands that modify PromICE data

6.51. !system

- escape command to DOS or UNIX shell

6.51.1. Command Forms

!command	in LoadICE dialog mode

6.51.2. Syntax

{!command}

6.51.3. Use

command	is the command word
"string"	is the command string to be executed by the system shell. After the command completes control is returned to LoadICE

6.51.4. Default

none.

6.51.5. Description

This command allows you to execute arbitrary system command without leaving LoadICE

6.51.6. Notes

Does not work on Macintosh or VMS system .

6.51.7. Examples

!dir

show the files in current directory

!edit myfile

run the editor and edit 'myfile'. When you exit the editor you will be back at LoadICE: prompt.

6.52. test

- test PromICE emulation memory.

6.52.1. Command Forms

test	(also **t**) in LoadICE dialog mode

6.52.2. Syntax

{**test** | **t**} [[*id* {:}] [*pass_count*]]

6.52.3. Use

id	(integer) A valid PromICE unit ID number (0-255)
pass_count	(integer) number of times a full memory test is to be performed.

6.52.4. Default

test unit with ID-0 once.

6.52.5. Description

It runs a simple RAM test within the PromICE memory. If the test fails then the offending address is reported.

6.52.6. Notes

destroys all data. A failed test may indicate several problems. It can be a damaged buffer interfering with the test (most likely), it may be weak battery causing write failures to the RAM, or it may be bad RAM chip (not very likely)

6.52.7. Examples

```
t 3
```

test unit-3 once

6.53. ulock

- support LOCKED PromICE units

6.53.1. Command Forms

ulock	in `LoadICE.ini` file
-M	In LoadICE command line

6.53.2. Syntax

{ulock | -M}   #

6.53.3. Use

#	(decimal number) Unit ID that has the LOCK feature

6.53.4. Default

none.

6.53.5. Description

Special micro-code allows a pair of signals on the back of the PromICE to be using for LOCKING a PromICE unit. A LOCKed unit can not be loaded with new code. The REQ and ACK lines when shorted together will lock a unit. Opening the lines will UNLOCK the unit

6.53.6. Notes

This feature is custom implemented in PromICE micro-code to allows the user at the target site to LOCK the unit with a switch attached to the REQ and ACK lines.

This feature is not available when AI is in use, since AI uses the two signals in question. .

6.53.7. Examples

ulock   0

treat the unit with ID 0 as a LOCKable unit. I.e. LoadICE will not let you down-load code or do any operation that will make the unit stop emulating if the switch is thrown.

6.54. version

- report version number of controlling program embedded in PromICE as well as that of LoadICE

6.54.1. Command Forms

version	(also **v**) in LoadICE dialog mode

6.54.2. Syntax

{**version** | **v**} [*id* {:}]

6.54.3. Use

id	(integer) A valid PromICE unit ID number (0-255)

6.54.4. Default

Display LoadICE version # as well as micro-code version of unit ID-0

6.54.5. Description

This is to check the version of software and micro-code you are running. This information is helpful in doing support.

6.54.6. Notes

Please have this information handy if calling for support

6.54.7. Examples

```
v 3
```

report version of the micro-code in unit-3

6.55. word

- specify emulated data bus width

6.55.1. Command Forms

word	in `LoadICE.ini` file
-w	as a command line argument to LoadICE
word	(also **w**) in LoadICE dialog mode

6.55.2. Syntax

{**word** \| **-w** \| **w**} [[*data_width*] [[*byte_n*] ...]]

6.55.3. Use

data_width	(integer) Data bus width of PromICE emulated memory. Must be an integral multiple of 8 and can not be larger than the data bus width emulated by the total number of daisy chained PromICE units.
byte_n	(integer) nth byte in a data bus that has a width that is an integral multiple of 8 bits. Bytes are assigned to successive PromICE units (0 - 255) based on the selected order of *byte_n* specifications.

6.55.4. Default

Word size is 8 bits

6.55.5. Description

This command is used for setting the word size configuration of the PromICE modules. If no word-size information is specified with the files than this information is applied to such files as well.

6.55.6. Notes

When doing interactive command the word size specification is used to dump, edit, fill, save PromICE memory.

6.55.7. Examples

```
w 16 1 0
```

Specifies that word of 16 bits is to be used and that the first byte of data (all the even numbered bytes) is to be loaded in unit ID-1 and the second byte of data (odd numbered bytes) to be loaded in unit ID-0.

7. QUICK TROUBLESHOOTING GUIDE

7.1. BEFORE YOU BEGIN

If this section doesn't resolve your problem, see the **Troubleshooting** section.

WARNING: If proper ESD precautions have not been taken you may have damaged buffers. If your PromICE appears to load fine but will not emulate or will emulate to a point in code and hang up reliably, there is a good possibility that the buffers have been damaged.

WARNING: Do NOT open the PromICE. All warranties are void if unit is opened!

YOU MUST CONTACT GRAMMAR ENGINE SUPPORT AT (614) 899-7878 FOR ALL SUPPORT RELATED INFORMATION.

CONTACTING YOUR SALES ENGINEER WILL SIMPLY DELAY IN YOUR GETTING NECESSARY HELP.

7.2. COMMUNICATING WITH THE PromICE

7.2.1. Multifunction board warning (ATI multifunction cards)

These cards have integrated serial/parallel, floppy/hard drive controllers, and/or video on a single board. Some of these cards cause conflicts between the I/O ports and the other devices (video, floppy, etc.). Replace the card with separate I/O, video and floppy/hard drive controllers.

7.2.2. Using the TTY as a diagnostic tool

If you are still having problems try using a TTY interface to connect to the PromICE. Connect the PromICE unit to a serial port. Set the TTY's parameters to 9600 baud, 8 data bits, No parity bit, and 1 stop bit. Enter the interface and send a couple of carriage returns to the TTY. Something like the following should appear:

```
V7.1c(C)93GEI
M>
```

The PromICE is communicating if the TTY can see it. If the TTY can see the PromICE then the problem is in the configuration. If the TTY can't see the PromICE then the unit may be damaged. Contact Grammar Engine Technical Support for assistance (see *If All Else Has Failed* at the end of this section).

7.2.3. Cannot connect with the PromICE (serial connection)

Causes	Solutions
No power to PromICE	• Make sure that the run light is on. If it isn't check the power connections to the unit. • If using parasitic power try running with external power. The target may not be able to source the current load.

Conflicting device drivers	• Check your config.sys and autoexec.bat for drivers that may effect the port being used.
Incorrect port	• Check the host connection. Make sure that the adapter is connected into the same port that LoadICE is using. • Check the LoadICE.ini file "com=" setting or your command line arguments.
Incorrect port address	• The port address may be non-standard. Specify the port address as: output=com1:*xxx*, where *xxx* is the port address.
Windows DOS box not exclusive	• See your Windows documentation to open a DOS window exclusively.
Network conflicts	• Log off the network. • A slower baud rate or running LoadICE parallel may help.
Extension cables connected	• Use only cables supplied with PromICE.
Poor serial connection	• Check PromICE phone jack for bent pins. Gently bend pins back into place. • Slow the baud rate. • Move cable away from power supplies, other cables, etc. • Make sure that your cable and adapters have the GEI logo on them. Contact Grammar Engine Sales if you need to obtain new cables or adapters. Pinouts are available in the Appendix for the various cable adapters.
Switch boxes	• Connect PromICE directly to host system. • Use the cables supplied with the PromICE unit only.
Gender changers	• Disconnect gender changer.

	• Adapters are available from GEI.
Bad serial port	• Use another host system

7.2.4. Loses connection with the PromICE (serial connection)

Causes	Solutions
Windows DOS box not exclusive	• See your Windows documentation to open a DOS window exclusively.
Network conflicts	• Log off the network. • A slower baud rate or running LoadICE parallel may help.
Baud rate	• Slow baud rate to 19200. • Check cable routing (see "Bad Connection" below).
Display mode set too high	• Set display mode lower. • "fe" is maximum display mode. • "c0" is default.
Conflicting device drivers	• Check your config.sys and autoexec.bat for drivers that may effect the port being used.
Extension cables	• Disconnect extension cable. • Use cables supplied with the PromICE only.
Switch boxes	• Connect PromICE directly to host system. • Use only cables supplied with PromICE.
Parasitic power loss / drop	• Check target power. It should be above 3.3 volts. • If power is close to threshold voltage (3.3V), jumper "EXT", "ROM", and "28" or "32" selections. POWER TARGET FIRST, THEN PromICE.

	• Contact Grammar Engine Inc. if this fails to solve the problem.
Poor Serial Connection	• Check PromICE phone jack for bent pins. Gently bend pins back into place. • Slow the baud rate. • Move cable away from power supplies, other cables, etc.
Bad serial port	• Use another host system

7.2.5. Cannot connect with the PromICE (parallel connection)

Causes	Solutions
No power to PromICE	• Make sure that the run light is on. If it isn't check the power connections to the unit. • If using parasitic power try running with external power. The target may not be able to source the current load.
Security keys (dongles)	• Disconnect security keys.
Switch boxes	• Connect PromICE directly to host system. • Use only cables supplied with PromICE.
Extension cables	• Disconnect extension cable. • Use cables supplied with the PromICE only.
Conflicting device drivers	• Check your config.sys and autoexec.bat for drivers that may effect the port being used.
Incorrect port	• Check the host connection. Make sure that the adapter is

	connected into the same port that LoadICE is using. • Check the LoadICE.ini file "pponly=" setting or your command line arguments.
Incorrect port address	• The port address may be non-standard. • Specify the port address as: pponly=lpt1:*xxx*, where *xxx* is the port address.
Switch boxes	• Connect PromICE directly to host system. • Use only cables supplied with PromICE.
Gender changers	• Disconnect gender changer. • Adapters are available from GEI.
Bad parallel port	• Try connecting with the serial port. • Use another host system

7.2.6. Loses connection with the PromICE (parallel connection)

Causes	Solutions
Parasitic power loss / drop	• Check target power. It should be above 3.3 volts. • If power is close to threshold voltage (3.3V), jumper "EXT", "ROM", and "28" or "32" selections. POWER TARGET FIRST, THEN PromICE. • Contact Grammar Engine Inc. if this fails to solve the problem.
Switch boxes	• Connect PromICE directly to host system.

	• Use only cables supplied with PromICE.
Gender changers	• Disconnect gender changer. • Adapters are available from GEI.
Conflicting device drivers	• Check your config.sys and autoexec.bat for drivers that may effect the port being used.
Extension cables	• Disconnect extension cable. • Use cables supplied with the PromICE only.
Bad parallel port	• Try connecting with the serial port. • Use another host system

7.2.7. Cannot connect with the PromICE (Duplex serial / parallel connection)

Causes	Solutions
See Serial and Parallel connection problems for more information.	
Incorrect connection	• Reverse serial or parallel cables.

7.3. COMMUNICATING WITH THE TARGET

Target doesn't emulate

Causes	Solutions
Incorrect emulation size	• Emulate the largest size the target is wired for. Check the "socket=" statement in the LoadICE.ini or in the command line.
Incorrect Byte order	• Make sure that your code is being loaded into the correct ROM. • For multiple files being loaded "per ROM" specify: file =filename 0=*x*:0 where *x* is the PromICE ID of the ROM the file is to be loaded. • For 16 bit or larger files, change the word specification to include the byte order: word=16 0 1 2 3 ....
Mapping	• Loading out of address range. • If loading data into RAM space include "noadderr" in the LoadICE.ini file. • When running LoadICE, verify that the code is being loaded into the correct location. • If loading multiple files, the data may be overlapping with the previous file data. • See "*Mapping*" in the Troubleshooting section of this manual for more information.
Parasitic power loss / drop	• Check target power. It should be above 3.3 volts. • If power is close to threshold voltage (3.3V), jumper "EXT", "ROM", and "28" or "32"

	selections. POWER TARGET FIRST, THEN PromICE. • Contact Grammar Engine Inc. if this fails to solve the problem.
Noise	• Reroute cables away from other cables and/or power supplies. • Use the shortest cables possible. • Externally power to the PromICE. • You may need shielded cables. • See also "*Parasitic power loss / drop*" above.

Target stops emulating

Causes	Solutions
Mapping	• Loading out of address range. • If loading data intended for RAM space include "noadderr" in the LoadICE.ini file. • When running LoadICE, verify that the code is being loaded into the correct location. • If loading multiple files, the data may be overlapping with the previous file data. • See "*Mapping*" in the Troubleshooting section of this manual for more information.
Parasitic power loss / drop	• Check target power. It should be above 3.3 volts. • If power is close to threshold voltage (3.3V), jumper "EXT", "ROM", and "28" or "32" selections. POWER TARGET FIRST, THEN PromICE. • Contact Grammar Engine Inc. if this fails to solve the problem.

Noise	• Reroute cables away from other cables and/or power supplies. • Use the shortest cables possible. • Externally power to the PromICE. • You may need shielded cables. • See also "*Parasitic power loss / drop*" above.

8. TROUBLESHOOTING

8.1. BEFORE YOU BEGIN

WARNING: If proper ESD precautions have not been taken you may have damaged buffers. If your PromICE appears to load fine but will not emulate or will emulate to a point in code and hang up reliably, there is a good possibility that the buffers have been damaged.

WARNING: Do NOT open the PromICE. All warranties are void if unit is opened!

YOU MUST CONTACT GRAMMAR ENGINE SUPPORT AT (614) 899-7878 FOR ALL SUPPORT RELATED INFORMATION.

CONTACTING YOUR SALES ENGINEER WILL SIMPLY DELAY IN YOUR GETTING NECESSARY HELP.

8.2. HOST TO PROMICE COMMUNICATION

8.2.1. Using the TTY interface as a diagnostic tool

If you are still having problems try using a TTY interface to connect to the PromICE. Connect the PromICE unit to a serial port. Set the TTY's parameters to 9600 baud, **8** data bits, **N**o parity bit, and **1** stop bit. Enter the interface and send a couple of carriage returns to the TTY. Something like the following should appear:

```
V7.1c(C)93GEI
M>
```

The PromICE is communicating if the TTY can see it. If the TTY can see the PromICE then the problem is in the configuration. If the TTY can't see the PromICE then the unit may be damaged. Contact Grammar Engine Technical Support for assistance (see *If All Else Has Failed* at the end of this section).

8.2.2. Windows

Do not run Windows for the duration of the diagnostic procedure. You can always try using Windows again later. If exiting Windows solves the problem, you must not have the LoadICE window opened exclusively. See your Windows documentation for more information about exclusive applications.

8.2.3. Network

Log off of any network you may be on. If this solves the problem, your network may be conflicting with the PromICE's operation. Sometimes, depending on the situation, slowing the baud rate can solve the problem. This, however, is not guaranteed to work. The best solution is not to run LoadICE while logged onto the network. Whether this is the problem or not, stay off the network until the problem is solved. The network may mask another problem. You can always log back in after the other problem(s) are taken care of.

8.2.4. LoadICE Version

Check your LoadICE version. Free upgrades can be downloaded from the GEI Bulletin Board System. See the "*If All Else Has Failed*" section for BBS protocol information.

8.2.5. Switch boxes / extension cables/ software keys

Disconnect any switch boxes, extension cables and/or software keys (dongles). Connect the PromICE to the host via the supplied cables only. Extension cables and switch boxes are not recommended for use with the PromICE. Software keys may intercept code intended for the PromICE and stop communication. Don't reconnect any of these cables or boxes even if disconnecting them doesn't help. They can always be reconnected once the PromICE is communicating.

8.2.6. Port specification

Make sure that you are connected to the right port. If the port has a non-standard address specify it in the command line or LoadICE.ini file as follows:

```
output=com1:3f8
```

OR

```
pponly=lpt1:378
```

8.2.7. Baud rate

Make sure you selected a valid baud rate. The PromICE supports 1200, 2400, 4800, 9600, 19200 and 57600 baud. Try using a slower baud rate. If this solves the problem you probably have a noise related problem. Rerouting the cabling away from power supplies and power cables may help this problem.

8.2.8. Serial/parallel communication

If you are using the serial and parallel port together, slow the serial port to 19200 baud. The PromICE cannot handle data at the full baud rate in this configuration.

8.2.9. Serial/parallel daisy chain

Make sure the daisy chain modules are connected properly. If the serial connection is the reverse of the parallel connection, the PromICE won't communicate (see also "Serial/parallel communication" above).

8.2.10. Noise

Move the communications lines away from any power cables, monitors, power supplies, etc.

8.2.11. Display mode

If you are setting the display mode in LoadICE too high the data will overflow the host system during communications. The maximum display mode is "fe".

8.2.12. Conflicting device drivers/TSR's

Check your config.sys and autoexec.bat files for any drivers or TSR's that may conflict with LoadICE. Anything that effects I/O ports can affect LoadICE. If in doubt you can rename your config.sys and/or autoexec.bat files. Reset the system and try LoadICE without any other drivers loaded. If this solves the problem then one of the drivers is effecting LoadICE communications. Disable one driver at a time until the problem is found. Some memory driver configurations have been known to cause problems.

8.2.13. Poor connection (damaged cable)

Take the phone cable out of the PromICE unit and out of the DB9 (DB25) connector. Look inside the PromICE and DB9 (or DB25) "phone jack" modules to see if any of the connection wires are bent or damaged. If a connector is bent, using a small hook, gently pull the line down to where is should be. If the line won't stay down, contact Grammar Engine Inc. for a replacement. Avoid this problem by using only the cables supplied with your PromICE unit. Disconnect any extension cables, switch boxes or software keys (dongles) that may be attached to the PromICE. Use only the cables supplied with the unit.

8.2.14. Damaged serial/parallel port

If communication is impossible through the serial port, try using the TTY interface as described earlier in this section. If you can connect with the TTY but not through LoadICE, you probably have a bad serial port. Try using another port preferably on another host system.

8.2.15. ATI multifunction cards

These cards have integrated serial/parallel, floppy/hard drive controllers, and/or video on a single board. Some of these cards cause conflicts between the I/O ports and the other devices (video, floppy, etc.). Replace the card with separate I/O, video and floppy/hard drive controllers.

8.3. PromICE TO TARGET COMMUNICATION

NOTE: Emulation problems show up in one of two ways. The target either never emulates, or stops emulating. This section of troubleshooting is split into two separate sections. Many of the problems experienced with emulation may show up in a number of ways. For example: If the target never emulates the problem could be noise related (in the "Never emulates or stops emulating section") or byte order related (in the "Never emulates" section).

If your target doesn't (emulate and never did) see the "Never emulates" section first. Then, if the problem persists, see the "Never emulates or stops emulating" section. Remember to log off the network and exit Windows. If this solves the problem, see the "Windows" and/or the "Network" sections below.

If, after completing the section(s) below, the target still won't emulate, contact Grammar Engine Inc. Technical Support.

8.3.1. Never emulates or stops emulating

8.3.1.1. Windows

Do not run Windows for the duration of the diagnostic procedure. You can always try using Windows again later. If exiting Windows solves the problem, you must not have the LoadICE window opened exclusively. See your Windows documentation for more information about exclusive applications.

8.3.1.2. Network

The next step is to log off of any network you may be on. If this solves the problem, your network may be conflicting with the PromICE's operation. Sometimes, depending on the situation, slowing the baud rate can solve the problem. This, however, is not guaranteed to work. The best solution is not to run LoadICE while logged onto the network. Whether this is the problem or not, stay off the network until the problem is solved. The network may mask another problem. You can always log back in after the other problem(s) are taken care of.

8.3.1.3. Mapping

If the file specification in the LoadICE.ini or command line is specified incorrectly, the program will not run or will run to a point and then stop (crash). First, if you don't have any code that you are loading into RAM space on purpose, make sure that noaddrerr is NOT in your LoadICE.ini or command line. This way, if you are loading out of address range, LoadICE will show the error and location. If the file address and/or the ROM address are specified incorrectly the program won't run or will crash. Make sure you know where your file's starting address is and where in ROM you want it to load.

Example: Assume you have a file called mad.hex that starts at 0x400000 (the hex records in the file say to start loading data at this address). Also, assume that the file has 16 bit data and the first byte is emulated by module ID1 (slave unit) and the second by module ID0 (master unit). The specification would be:

```
file=mad.hex 400000=0 16 1 0
```

The "16 1 0" is optional if "word=16 1 0" has been specified. Binary images aren't much different.

Example: Assume you have a file called mad.bin that has a 14 byte load header. Assume that the file has 16 bit data and the first byte is emulated by module ID0 (master) and the second by module ID1 (slave). The specification would be:

```
image=mad.bin 14=0 16 0 1
```

Again, the "16 0 1" are optional if the "word=16 0 1" has already been specified. You can always be sure by specifying all the parameters.

8.3.1.4. Emulation size

Emulate the largest size that your target IS WIRED FOR. If the ROM you are emulating is a 27512, but your target's socket can handle a 27020, emulate the 27020. If you don't know for sure whether your target can emulate larger sizes or not, start by emulating the largest size ROM that your PromICE can emulate. Work your way down from there until you find the correct emulation size.

8.3.1.5. Noise

Check the cabling between the PromICE and the target. Make sure that the cable isn't near any power supplies or cables. If your target is sensitive to noise or is noisy itself, it may not emulate (or at least not reliably). Make sure that none of your cables are anywhere near a source of noise (i.e. monitors, power supplies, power cables, etc.). Keep the cables as short as is possible and route them as far away as possible from noise sources. If the target itself is noisy it may become necessary to obtain shielded cables. Contact Grammar Engine if this becomes necessary. See also "Parasitic power" below. You may also need shielded cables.

8.3.1.6. Parasitic power

The PromICE unit is designed to run either parasitically (power comes from target ROM socket) or externally. The PromICE draws about 100 mA per board inside. If you have a duplex (2 PromICE's in a box) with an AI interface, the unit will draw approximately 300 mA from your target if you are running the PromICE parasitically. This may be enough power drain on the target system to make it fail. In addition, whether powered parasitically or externally, if the target voltage drops below a certain voltage threshold the PromICE will go into protected mode. During this time, the PromICE will stop emulation and turn off the buffers to protect its memory. For Revision 3 units the voltage is approximately 3.3 volts. Although the voltage may be high enough at startup, there may be a function that enables a peripheral that takes power away from the target. This drain may drop the target voltage below the threshold. To resolve this problem, the voltage sense can be forced to 5v:

Remove power from the target and PromICE. Place jumpers on the "EXT" and "ROM" positions. Place another jumper on the 28 or 32 pins depending on the adapter you are using. Only the 28 pin DIP cable will use the "28" pin jumper. All other PLCC adapters and 40 pin DIP adapters will use the "32" jumper. Connect the reset line on the PromICE to the target reset line. If you have a reset button on the target, you can use it instead of the reset line on the PromICE.

WARNING: You will not be able to power down your target to initiate a reset. In this configuration, powering down the target without first powering down the PromICE can damage the unit.

Power up the target, THEN the PromICE. Rerun LoadICE.

If you are using parasitic power (PromICE gets its power from the ROM socket) and the target stops emulating at certain points, the PromICE is probably going into protected mode. The PromICE stops emulation and turns off the buffers that go to the target when the target voltage drops below a certain voltage. If your target locks up while performing a certain function the target voltage may be dropping below this threshold. Try powering the PromICE externally. If this still doesn't help you may want to try to force the voltage sense. See the "**Voltage sense**" section under "**NEVER EMULATES**" to do this.

8.3.2. Never emulates

8.3.2.1. Incorrect byte order

If you are emulating more than one ROM and/or are in a 16 or larger emulation, check the byte order in the LoadICE.ini file or your command line. If the byte order is reversed the code is going into the wrong ROMs. In duplex PromICE's, the bottom unit is ID 0 (usually the even bit) and the top is ID 1 (usually the odd bit).

8.3.2.2. Emulation size

Check your emulation size. Make sure the socket statement is set correctly and that the ROM size is not set larger than the socket size. If this is set incorrectly, the PromICE will not emulate. If you are not sure what your target is wired for, set the socket size to emulate a 27080 and then work your way down.

8.4. IF ALL ELSE HAS FAILED

If the problem cannot be solved from the information supplied in this manual then you may contact Grammar Engine Technical Support. Support is available via Phone, Fax or BBS.

If you choose to use the phone option be prepared with the following information:

- revision number
- model number (On front of PromICE. It starts with a "P")
- Contents of the LoadICE.ini file or command line
- Exact error message being encountered
- Target information:
  - CPU type
  - Clock rate
  - ROM access time
  - ROM part number
- All switch and/or jumper settings on the PromICE
- Host type and type of connection to PromICE (serial, parallel or both)

Be at your computer with the PromICE and target configured and ready to go. This will help us get you up and running faster.

The preferred method of obtaining Technical Support is through the BBS or Fax. Fax or mail the previous information to the BBS Sysop or Fax it attention Technical Support. Include your company name voice phone number and Fax number. The more detail you can give us about your problem and what you have done to try to resolve it will help us solve the problem faster.

We strongly recommend the use of the Grammar Engine BBS service, since it offers same day turnaround of problem reports, sample software and LoadICE updates in a timely and efficient manner. Grammar Engine's BBS service is available 24 hours a day 7 days a week for access to the latest versions of LoadICE and other software. The BBS is checked 8 times daily for messages. You can also leave detailed messages and possible output and/or data files to help us solve your problems. This method is specially encouraged if you wish to add some new capability to our software or modify it for some unique purpose.

Our numbers are:

Technical Support	(614) 899-7878
Fax	(614) 899-7888
BBS	(614) 899-6230
INTERNET	support@gei.com
COMPUSERVE	71121,1603

8.5. EMERGENCY REPAIRS

You may be in a situation where you must repair the PromICE if you can and continue with your work. In such cases the instructions provided here may help. Warning: This will definitely void any warranty and you assume the full risk of applying any fixes here. Any mistakes may further damage the unit or your target system. If you are very careful and take full precautions against static damage, you can proceed. Grammar Engine will not be liable for any consequence of following these instructions.

8.5.1. Replacing ROM Interface Buffers

Many of the problems of the PromICE can be traced back to damaged buffers on the ROM interface. If you are experiencing failure to emulate, specially when the unit was operating correctly before. Or you begin to get the Memory Size Zero message from LoadICE when you connect with the PromICE. Then you most likely have partially of completely damaged buffers on the ROM interface. One or more damaged buffers can keep some address line from switching and cause the above mentioned problems.

You may also run in to a situation where the unit works on one target but not the other. In such cases the ROM interface buffers may need to be changed with some of another technology such as CMOS, FCT etc.

To replace the buffers do the following:

1. Open the PromICE unit by removing the flat head screw on the back panel that is attached to a heat-sink. Use #1 Phillips screw driver and remove the two screws from the bottom portion of the PromICE box.

2. Slide out the circuit boards and the panel all together. Remove the panels and gently pry apart the circuit boards, if you have duplex unit, or AI option installed.

3. The address input buffers are marked U11, U12 and U13 and are located behind the ROM cable header. These are 20 pin devices in sockets. These are generally 74ALS244 chips. They can be replaced by equivalent known good chips. If you do not have enough chips then replace them one at a time, until the defective chip is replaced.

4. The data buffer is marked U14 and is a 74ALS245. Replace with similar chip.

5. The interface chips are marked the same on both the master board (the one with main PromICE circuit on it) and the slave board (mounted on the master board).

6. After the chip are replaced, reassemble the boards carefully lining up the 44 pin headers on both sides of the board. You may refrain putting the unit in the box until the problem is completely fixed.

7. For complete reassemble, hold the panels in position with the boards and gently slide the whole assembly into the bottom portion of the box. Attach the heat sink back to the back panel with the flat head screw. Slide the top half of the box and close it with the Phillips screws through the bottom half of the box.

8.5.2. Replacing the Parallel Port Buffers:

If you are having problems with the parallel port, and it can not be fixed by any of the things mentioned in regular trouble shooting section, then the parallel port buffers may be damaged.

Once again, when the unit is open. Check the two buffers right behind the parallel port connector. These are also 20-pin devices and are in sockets. They are 74LS244 and 74ALS244 chips. Replace them with equivalent chips.

8.5.3. Replacing the Battery

After disassembling the unit, you should check the power on the SRAM backup battery. You can conveniently check the power on the RAM chips, or the empty RAM socket. These are the 32 pin socket and there are four of them. Check the power (with no power applied to the PromICE) between pin 16 (GND) and pin 32 (VCC). There should be at least 2.7VDC of power from the battery. The NV controller chip (usually Dallas Semiconductor: DS1221 chip) will not let you access RAM reliably if the battery power is low.

Desolder the battery from the board and install a similar battery.

If the battery power is okay but power at the SRAM is non-existent, the DS1221 may be damaged.

8.5.4. Replacing Other Parts

Theoretically you can replace all the parts but the PAL. But if the problem can not be solved by replacing the buffers, then the only other parts that are easy to replace are the ones in buffers, or some of the discretes. Here is a list of parts that you may replace:

1. The 8-pin OpAmp part (the only 8-pin IC on board, and in socket). The unit will not go into emulation and the target has enough power. This OpAmp will force emulation to shut down if it thinks the target power is going down.

2. The 7805 regulator. If PromICE will not maintain 5VDC power, the regulator may have gone bad. Look for discoloration from excessive heat. Desolder the regulator, solder a new one, reattach the heat sink.

3. The 74HCT373. This chip is soldered on to the board. If it has failed, it will not let PromICE correctly address the memory. It will have to be desoldered and replaced. Do not replace with an HC part, must be HCT or LS or ALS etc.

4. The 74HCT125. This is a 14 pin device in a socket. This chip drives the target RESET and INTERRUPT lines on the back pin rst(- +) and int(- +). If these signals are not functioning properly, then this chip can be replaced with an equivalent part. LS or ALS will also do and even an HC may be used.

5. The last chip is the LT1081, a 5VDC only RS232 interface device. If the problem seems to be RS232 then this chip can be desoldered and replaced with an equivalent chip.

Note: Both the Dallas DS1221 and Linear Technology LT1081 (or LT1281) have pin compatible parts made by BenchMark and Maxim, respectively. Carefully check the part pinouts before substituting.

GOOD LUCK!!

8.6. TECHNICAL SUPPORT

PRIORITY FAX

To: **GRAMMAR ENGINE TECHNICAL SUPPORT**
Fax: (614)899-7888

From: ______________________________
Company: ____________________________
Fax: (   ) ______________________
Voice: (   ) ______________________

PromICE revision
_____ Rev 1 (1 light on front panel)
_____ Rev 2 (4 lights on front panel, DIP switches on back)
_____ Rev 3 (4 lights on front panel, reset button on back (no switches))

Model # **P**_______________ (Please include complete number)
Serial # _________________ (Below model number)
(*model and serial numbers are located on the front or bottom of the unit*)

Jumper settings on the PromICE:
Revision 3: **EXT ROM 32 28 24**
Revision 1 and 2: **EXT 32 28 24**

Switch settings on the PromICE (if applicable): 1 2 3 4 5 6 7 8 9 10 (circle if on)

LoadICE version number: ____________ (Run LoadICE to get full version #)
Contents of the LoadICE.ini file:

______________________ ______________________
______________________ ______________________
______________________ ______________________
______________________ ______________________

Command line arguments: ______________________________________
Error messages: ____

Target CPU: __________ Clock Rate: _____ ROM access time: ______
ROM part # __________ ROM wait states: _____ Bus width: **8 16 24 32**

Host type and speed: ______________ Switchbox: **YES / NO**
Connection to PromICE: **Serial / Parallel / Both** Extension cables: **YES / NO**
Debugger used: ___________________________________

Additional Information:

____
____
____

8.7. RMA INFORMATION

SHIP TO: Grammar Engine, Inc.
921 Eastwind Dr.
Suite 122
Westerville, OH 43081

ATTN. RMA# ______________ (Contact Grammar Engine Technical Support for RMA)

PromICE **MODEL #** (On front panel or bottom of unit):
P____________________________

PromICE **SERIAL NUMBER** (Under model number):

____

Purchase Order number
(If out of warranty and/or for special shipping): ______________

Return Shipping instructions:

____

Return Address: ____________________________
____
____
____
____
____

Contact Name: ____________________________
Contact Phone # ____________________________

Reason for Return: _____ **Upgrade** _____ **Repair**

Problem Description:

____
____
____
____
____
____

9. ERROR MESSAGES

Normal messages from LoadICE inform you of the progress and show you the data per your request. The following messages, however, are error messages that are displayed when errors are encountered. Unless you are in dialog mode, errors will terminate LoadICE. In dialog mode the control is returned to you for further command input.

Error messages are always identified as such, and auxiliary data is also displayed to give you a little more information about errors. If the global variable 'errno' is set, then a system error message is also displayed. It may not always be meaningful but, it usually means that some system call that LoadICE has issued has failed, producing the error message. LoadICE usually displays the last user input it was processing. In some cases it may not really apply to the error condition. In other cases the error may have occurred in processing the input after part of it had been successfully processed. Following, is the list of errors you may get from LoadICE:

9.1. LoadICE parser internal error #1

Causes	Solutions
Modifying the script files to	
an incorrect character in the script.	•customize LoadICE. You
entered Correct the input script and	
	retry the command.

9.2. Invalid input

Causes	Solutions
Spelling	• Check spelling and retry.
Capitalization	• All commands, unless otherwise noted should be lower case.

9.3. LoadICE parser internal error #2

Causes	Solutions

This is an internal error unless you have added a script that parses to many numbers. You can change a definition in the *piconfig.h* file to get around this restriction.

9.4. Too many numeric items in input

Causes	Solutions

This message is similar to the above except it relates to items in a list of numbers. See the script processor for detail.

9.5. Expected item not in input

Causes	Solutions
Required input was left out	• Check your configuration for information that may be needed. • Most commands have default values for unspecified input.

9.6. Filename error

Causes	Solutions
Filename doesn't exist in directory or specified path	• Make sure the file is in the same directory as LoadICE.
Incorrect filename	• Check the name of the file to be loaded and it's extension.

9.7. Invalid baud-rate

Causes	Solutions
Invalid baud rate specified	• Specify a supported baud rate i.e. 1200, 2400, 4800, 9600, 19200, or 57600.

9.8. Invalid ROM size

Causes	Solutions
ROM size specified is not a valid size	• Specify a generic part number such as 27512, the size in k bytes such as 64k, or an actual decimal number such as 65536.

9.9. Invalid word-size

Causes	Solutions
Word size is not a multiple of 8	• Specify a valid word size, such as 8, 16, 32, 64...

9.10. Invalid ID list

Causes	Solutions
Attempting to load into more units than exist	• Check your configuration for an id that is larger than the number of physical PromICE units. • Check file specification(s) and/or id list for incorrect id listing.

	• See Quick Start section "*Software Configuration*" for more information.
Word size too large	• Divide the wordsize by 8. If this number is larger than the number of PromICE units, you either need more units or the wordsize is incorrect.
	• See Quick Start section "*Software Configuration*" for more information.

9.11. Open failed

Causes	Solutions
File name not specified correctly	• Make sure the file and it's extension are correctly spelled and are in lower case.
File not in path	• Either the file should be in the same directory as LoadICE or the file path should be specified.

9.12. Unable to skip file data

Causes	Solutions
Skipping more data than is in file	• Check your skip count. It should not be larger than your file size.

9.13. Device I/O error

Causes	Solutions
This can happen with either the file I/O or the I/O on the link to the PromICE. Check the system error message for a more meaningful description.	

9.14. Bad port name

Causes	Solutions
Port doesn't exist	• Check the port name. • If the port does exist an address may need to be specified.
Non-standard port address	• Check the address of the host port. • It may be necessary to specify the address of the port. • See the Quick Start section "*Software Configuration*" for more information.

9.15. End-o-File

Causes	Solutions
Corrupt record in hex file	• Check the file, remake it.

9.16. Bad parameters to picmd()

Causes	Solutions
Count field larger than allowed.	• Should only occur if LoadICE has been modified. • Check the function prototype definition in *"pidriver.h"* for the

	maximum number of arguments.

9.17. Communication error

Causes	Solutions

LoadICE could not establish a link with the PromICE or the link failed. This error effectively means that LoadICE and the PromICE are confused about their current state. Cycle power on the PromICE and restart LoadICE.

9.18. Name too long

Causes	Solutions
File or device name too long	• Shorten the name to no more than 128 characters long.

9.19. Timed out waiting for PROMICE

Causes	Solutions
LoadICE cannot talk to the PromICE(s)	• See the QuickShoot section *"COMMUNICATING WITH THE PromICE"* .

9.20. FEATURE NOT IMPLEMENTED YET!

Causes	Solutions
Internal error	• Contact Grammar Engine Technical Support for more information.

9.21. Verify failed

Causes	Solutions
Noisy connection with the PromICE	• Move serial and/or parallel cables away from monitors, cables and other sources of noise. • Usually accompanied with error 19. • If problem persists contact Grammar Engine Technical Support for assistance.
Bad RAM in PromICE	• Rerun LoadICE and run the "test" command. This will test the PromICE memory. • Contact Grammar Engine Technical Support if test fails.

9.22. Invalid unit ID

Causes	Solutions
ID specified is larger than total number of PromICE units	• Check configuration for an incorrect ID. • Check file and word specifications for an invalid ID. • See the QuickStart section *Software Configuration* for more information.

9.23. Address out of range

Causes	Solutions

Incorrect offset	• ROM size being emulated is too small. • Offset is set incorrectly. • See *"file"* in the COMMAND REFERENCE for more information.
Some code being loaded is intended for RAM	• add *"noaddrerr"* to ignore this error.

9.24. LoadICE internal error #3

Causes	Solutions
Not enough memory to process config	• Recompile LoadICE with larger parameter in "*piconfig.h*".

9.25. Bad arguments

Causes	Solutions
Some command has invalid input	• Check input values in LoadICE.ini file, command line or dialog mode. • See the COMMAND REFERENCE section of this manual to verify syntax for a particular command.

9.26. Bad checksum in record

Causes	Solutions
File has been manually edited (patched)	• Use the nochecksum or *-x* option to process and load the edited records.

9.27. Feature not supported by the Host system

Causes	Solutions
"! command" attempt on MAC or VMS system	• "!command" only works on DOS systems.

9.28. Bad argument for driver call

Causes	Solutions
Corrupt LoadICE application	• Contact Grammar Engine Technical Support or BBS for a LoadICE replacement.

9.29. Bad data in the hex record

Causes	Solutions
The file processor found data that it doesn't know how to handle.	• Recompile file and retry. • The error string will contain info about the error. Correct any problems and retry.

9.30. Unit is LOCKED

Causes	Solutions
On units with an "M" in the model number only.	• Remove jumper from across the "*ack*" and "*req*" on the PromICE. *NOTE:* This is a custom feature only available on custom units with an "M" in the model number. Placing a jumper across these pins on any other unit will cause

damage. If there is any question as to whether or not you have one of these units, contact Grammar Engine Technical Support.

9.31. Not enough units to emulate the word-size

Causes	Solutions
Not enough PromICE units to emulate the word size.	• An attempt was made to emulate a word size larger than the number of PromICE units in the config. • Divide the word size by 8. There should be the same number of PromICE units as the number you come up with. PromICE models beginning with "P2" count as two PromICE units and models beginning with a "P1" count as one unit. Refer to the QUICK START section "*Software Configuration*" for more information.

9.32. Memory size ZERO! Cycle power on PromICE

Causes	Solutions
Using parasitic power to PromICE	• Power the unit externally. This will usually solve the problem.
Unit is failing (rare case)	• Contact Grammar Engine Technical Support for more information.

9.33. Operation terminated by user

Causes	Solutions
A command was terminated during processing.	• If you didn't stop the operation yourself it is possible that noise on the line interfered with the operation. This is extremely rare. If it occurs often refer to the QUICK SHOOT section of this manual.

9.34. Data over-run

Causes	Solutions
Interference from a network.	• Disable the network, including any network drivers and retry.
The "*high*" statement has been specified.	• Take the statement out of the LoadICE.ini file or out of the command line. Insert 'slow' statement in the ini file

9.35. Failed to establish link with RemoteView

Causes	Solutions
Internal error	• Contact Grammar Engine Technical Support for more information or see the section "*If All Else Has failed*" in the Troubleshooting section of this manual for more information.

9.36. Unable to communicate with RemoteView

Causes	Solutions
Internal error	• Contact Grammar Engine Technical Support for more information or see the section "*If All Else Has failed*" in the Troubleshooting section of this manual for more information.

9.37. There is no link with the target

Causes	Solutions
Internal error	• Contact Grammar Engine Technical Support for more information or see the section "*If All Else Has failed*" in the Troubleshooting section of this manual for more information.

9.38. Emulating!! do a 'stop' first

Causes	Solutions
A command was issued that requires the PromICE to be in "load" mode	• Issue a "*stop*" command or press the escape key (The escape key will toggle the PromICE between emulation and load modes the same as issuing a "stop" or a "go" command). • When emulation is stopped, the target will crash and will need to be reset after the "go" command is issued.

9.39. Unit is not emulating!! do a 'go' first

Causes	Solutions
A command was issued that requires the PromICE to be in "emulation" mode	• Issue a "*go*" command or press the escape key (The escape key will toggle the PromICE between emulation and load modes the same as issuing a "stop" or a "go" command). • If the reset line is not connected from the PromICE to the target, the target will have to be reset after the "*go*" is issued.

10. Technical Specifications

10.1. Identification

PromICE models are identified as P1xxx or P2xxx as follows:

P1*nnn*- Simplex: Single (master) module for emulating 1 ROM.

P2*nnn*- Duplex: Two modules (master and slave in one box) for emulating 2 ROMs.where *nnn* is one of the following indicating the maximum capacity of the ROM:

512	Emulates 2716 - 27512	(64KBytes)
010	Emulates 2716 - 27010	(128KBytes)
020	Emulates 2716 - 27020	(256KBytes)
040	Emulates 2716 - 27040	(512KBytes)
080	Emulates 2716 - 27080	(1MBytes)
160	Emulates 2716 - 27160	(2MBytes)

Further affixes that may be added to indicate the following options:

nnn: If faster than standard speed, then speed in nano-seconds is *nnn.*
AI: Analysis Interface for special firmware development features.

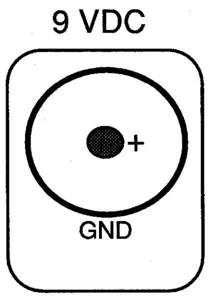
10.2. Power Consumption

Power consumption will vary depending on the buffers on the ROM emulation interface. Also faster models consume more power due to faster SRAMs and faster buffers. The figures below are based on a 4 Meg unit built with 4 1Mbit SRAMs and the ALS buffers.

PromICE Master: +5VDC < 200mA
PromICE Slave : +5VDC < 150mA
PromICE Analysis Interface: +5VDC < 70mA

Power Jack:

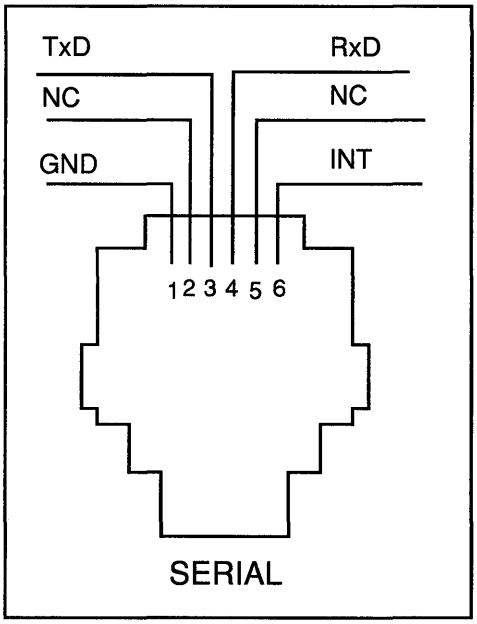
Pin and sleeve plug, with pin as +ve and sleeve as ground. External supply provides 9 VDC ~500 MA (unregulated)



10.3. Interfaces

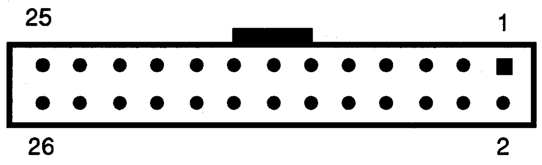
10.3.1. Serial Interface / RJ11 jack

RS232-C, connects to the host or a terminal via 6 conductor modular cables. Pinout (pins are numbered from left to right when looking at the front of the unit) is GND-1, nc-2, TxD-3, RxD-4, nc-5, INT-6. GND is signal ground. TxD is data out of the PromICE and RxD is data into the PromICE. INT is the interrupt from the host and is normally connected to DTR. Among other things the INT signal is used for restarting the PromICE from a known state.



10.3.2. Parallel Interface / 2x13 male header

Centronics a compatible parallel printer port configured for direct connection to DB25 connector on the back of PC or Compatibles. It can operate bidirectionally by using 'error status lines' for sending data back 4 bits at a time.



PIN#	SIGNAL	SIGNAL	PIN#
1	STROBE	AUTOFEED	2
3	D0	ERROR	4
5	D1	INIT	6
7	D2	SELECTIN	8
9	D3	GND	10
11	D4	GND	12
13	D5	GND	14
15	D6	GND	16
17	D7	GND	18
19	ACK	GND	20
21	BUSY	GND	22
23	PE	GND	24
25	SELECT	GND	26

The data transfer protocol is modified so that the parallel port can be used bidirectionally. STROBE is used to send data from the host to the PromICE. BUSY is asserted by the PromICE to indicate that it either has not read the previous data from the host or it has data to send to the host. For this reason, the SELECTIN signal is used to acknowledge the unasserted state of BUSY to the PromICE. This ensures that the sense of BUSY is never confused by the host.

For sending data back, the PromICE places the data 4-bits at a time on the ACK, PE, SELECT, and ERROR bits and asserts BUSY signal.

When two PromICE units are daisy-chained on parallel port, the AUTOFEED is used as STROBE for the second unit, and PE used as BUSY. Because of this the port is used for download only and the serial daisy-chain also must be used.

10.4. Indicators

On rev3 units there are four indicators, RUN - programmable run light, blinks during connect sequence; Rx & Tx - two indicators for received and transmitted data signals (serial data only); LOAD - indicates when the unit is in load mode, i.e. not emulating.

10.5. Enclosure

5.08" Wide, 1.5" High w/o rubber feet, 5.25" Deep, Impact-resistant, ABS-molded, Grade DFA/R Plastic.

10.6. Environmental Restrictions

Operating Temperature: 5 to 32 degree C (41 to 90 degrees F)
Storage Temperature: -40 to 70 degrees C (-40 to 158 degrees F)
Humidity: 90% maximum without condensation.

10.7. Accessories

10.7.1. External Power Supply

A wall mount power-supply is able to provide 9VDC unregulated and up to 500ma of current. The voltage is regulated by an internal 5V regulator (7805 or equivalent). The plug on the power supply has the sleeve as ground and the pin as positive.

10.7.2. Standard ROM Cables

A standard (.6") DIP plug on 12" 24 and 28 Conductor Ribbon cable with a 34-position Female Header for mating with connector on the back of the unit. See pinout for the header on the PromICE and DIP plug on cable ahead.

10.7.3. ROM Socket

JEDEC 24/28/32 pin DIP socket w/150ns access. Non-JEDEC and non-DIP footprints are handled via custom cables.

10.7.4. Modular Adapters

Modular to DB25 Male or Female and Modular to DB9 Male or Female are available. The pin out at the DB25 side of the adapter is RD-3, TD-2, RTS-4, CTS-5, GND-7, DTR-20 and the DB9 side of the adapter is RD-2, TD-3, RTS-7, CTS-8, DTR-4, GND-5.

10.7.5. Mini-Clip

The pin sleeve for attaching to the PromICE pins and a micro-hook for attaching to the target system. Clip provided for reset line.

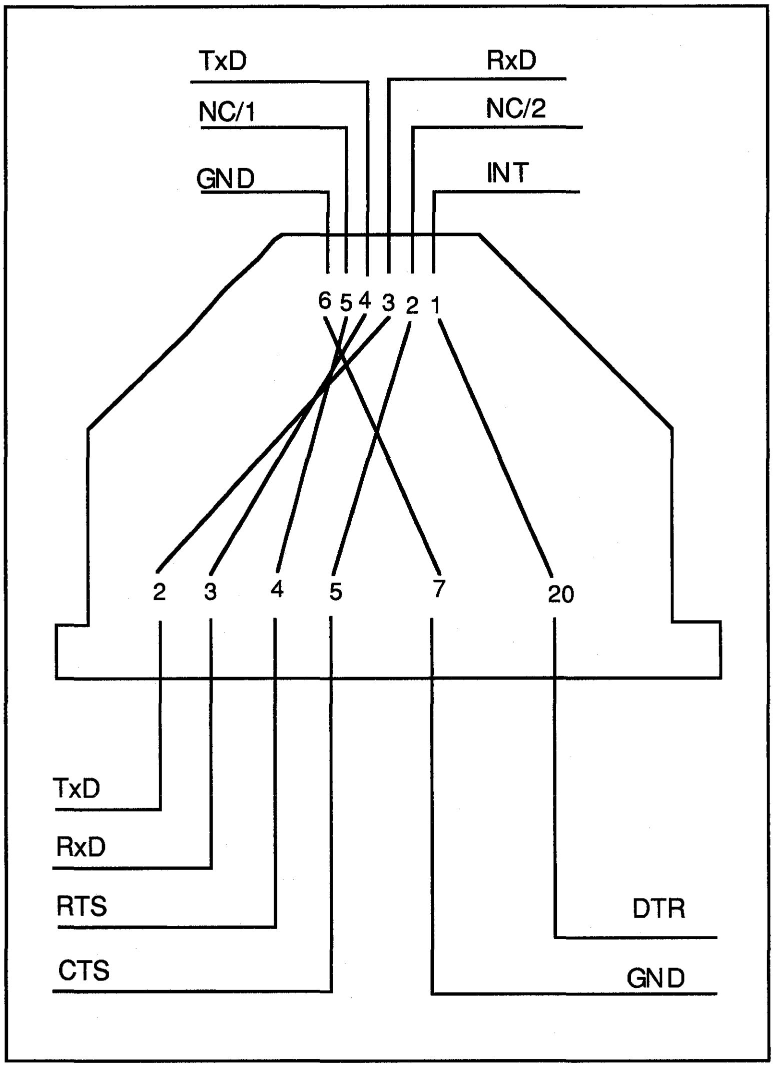
10.7.6. Custom ROM Cables

All types of custom cables are available for emulating PLCC or 16-bit wide or .3" center ROMs. Inquire about the adapter for custom ROMs.

10.8. Adapter Pinouts

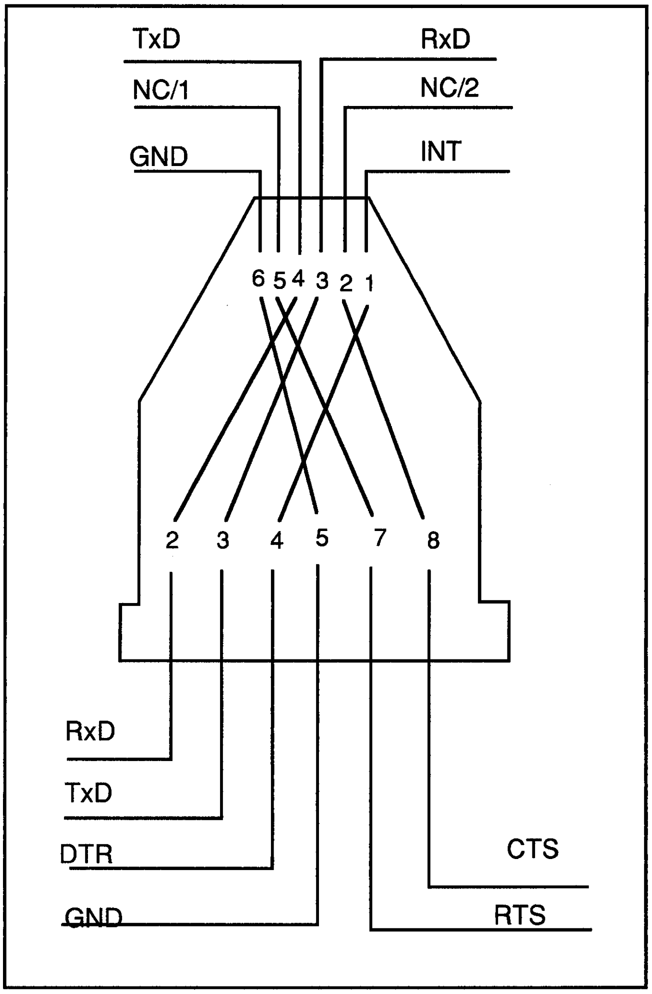
10.8.1. Female and Male DB25 Adapter

The RJ11 is oriented as a mirror image of the PromICE connector.



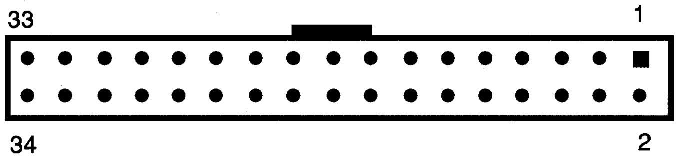
10.8.2. Female and Male DB9 Adapter

The RJ11 is oriented as a mirror image of the PromICE connector.



10.8.3. ROM Cable Header Pinout

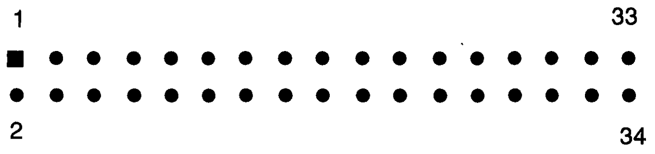
Pinout for the PromICE ROM cable header as seen from the back of the unit.



PIN#	SIGNAL	SIGNAL	PIN#
1	GND	A0	26
2	A20	A1	24
3	VCC-32	A2	22
4	A19	A3	20
5	A18	A4	18
6	A16	A5	16
7	A17/vcc-28	A6	14
8	A15	A7	12
9	A14	A8	13
10	A12	A9	15
11	A13/vcc-24	A10	21
12	A7	A11	17
13	A8	A12	10
14	A6	A13/vcc-24	11
15	A9	A14	9
16	A5	A15	8
17	A11	A16	6
18	A4	A17/vcc-28	7
19	OE_	A18	5
20	A3	A19	4
21	A10	A20	2
22	A2	CS_	23
23	CS_	OE_	19
24	A1	D0	28
25	D7	D1	30
26	A0	D2	32
27	D6	D3	33
28	D0	D4	31
29	D5	D5	29
30	D1	D6	27
31	D4	D7	25
32	D2	VCC-32	3
33	D3	GND	34
34	GND	GND	1

vcc-*nn* -> is the vcc for the appropriate pin device when parasitic

Pinout for the PromICE ROM cable header seen from the circuit side of the master board



PIN#	SIGNAL	SIGNAL	PIN#
1	GND	A0	26
2	A20	A1	24
3	VCC-32	A2	22
4	A19	A3	20
5	A18	A4	18
6	A16	A5	16
7	A17/vcc-28	A6	14
8	A15	A7	12
9	A14	A8	13
10	A12	A9	15
11	A13/vcc-24	A10	21
12	A7	A11	17
13	A8	A12	10
14	A6	A13/vcc-24	11
15	A9	A14	9
16	A5	A15	8
17	A11	A16	6
18	A4	A17/vcc-28	7
19	OE_	A18	5
20	A3	A19	4
21	A10	A20	2
22	A2	CS_	23
23	CS_	OE_	19
24	A1	D0	28
25	D7	D1	30
26	A0	D2	32
27	D6	D3	33
28	D0	D4	31
29	D5	D5	29
30	D1	D6	27
31	D4	D7	25
32	D2	VCC-32	3
33	D3	GND	34
34	GND	GND	1

vcc-*nn* -> is the vcc for the appropriate pin device when parasitic

10.8.4. Standard ROM Cable Pinouts

32-PIN DIP

PIN#	SIGNAL	SIGNAL	PIN#
1	A19	VCC	32
2	A16	A18	31
3	A15	A17	30
4	A12	A14	29
5	A7	A13	28
6	A6	A8	27
7	A5	A9	26
8	A4	A11	25
9	A3	OE_	24
10	A2	A10	23
11	A1	CE_	22
12	A0	D7	21
13	D0	D6	20
14	D1	D5	19
15	D2	D4	18
16	GND	D3	17

28-PIN DIP

PIN#	SIGNAL	SIGNAL	PIN#
1	A15	VCC	28
2	A12	A14	27
3	A7	A13	26
4	A6	A8	25
5	A5	A9	24
6	A4	A11	23
7	A3	OE_	22
8	A2	A10	21
9	A1	CE_	20
10	A0	D7	19
11	D0	D6	18
12	D1	D5	17
13	D2	D4	16
14	GND	D3	15

24-PIN DIP

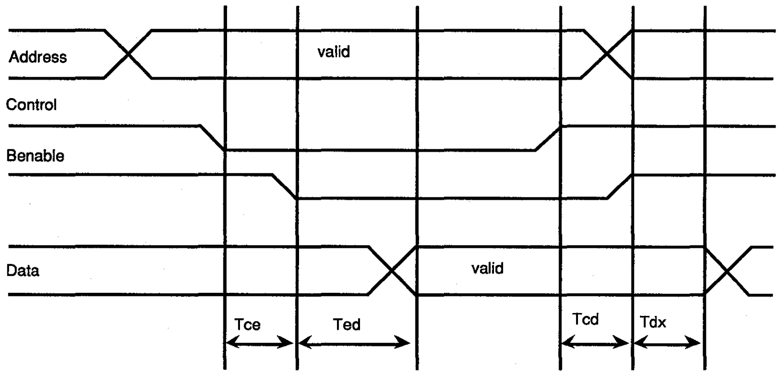
PIN#	SIGNAL	SIGNAL	PIN#
1	A7	VCC	24
2	A6	A8	23
3	A5	A9	22
4	A4	A11	21
5	A3	OE_	20
6	A2	A10	19
7	A1	CE_	18
8	A0	D7	17
9	D0	D6	16
10	D1	D5	15
11	D2	D4	14
12	GND	D3	13

10.9. Timing Diagrams

Here is how the target signals are received by the PromICE. The *address bus* and the *chip_select* and *output_enable* control signals are received by uni-directional buffers (74ALS244) and the *data bus* is connected to a bi-directional buffer (74ALS245) . When PromICE is in 'load' mode, these buffers are turned off and their outputs are tri-stated.

When PromICE is emulating these buffers are turned on and the address buffers supply address to the emulation memory within the PromICE. Data buffer is enabled by a combinations of *chip_select* and the *output_enable* signals. Direction of the data buffer is controlled by a target supplied *write* signal.

Emulation memory is made up of SRAM chips that are selected directly by the address supplied by the target system (in other words without using *chip_select* or the *output_enable* signals). This allows faster access to data. The data is placed on the target's data bus by the data buffer. This achieves a faster response from the PromICE to a target driven ROM read cycle.



Tce - time from chip_select & Output_enable to internal buffer enable.

Ted - time from buffer enable to data valid (delay through 74xxx245)

Tcd - time from chip_select & output_enable to buffer disable

Tdx - time from buffer disable to data bus tri-state

These times will vary according to the buffers used. The typical values are: Tce - 35ns / Ted - 20ns / Tcd - 35ns / Tdx - 20ns

11. Operating the AI as a Transparent Link

A typical monitor based debugging system involves blasting the monitor into a ROM and placing it on the target. This monitor has support for some on board serial I/O. The serial I/O is connected to a *comm* port on the host. The host resident portion of the debugger talks to the monitor on the target, and down-loads and debugs the applications on the target system. Many off-the-shelf cross development systems can operate in this manner. Many users also have their own debugging systems that operate this way.

When the target system cannot spare any I/O or does not have enough RAM to down-load applications, then it is desirable to find and alternate solution. The PromICE with the AI option offers a solution that can solve both problems. The monitor and the application can be down loaded into the PromICE. The AI option is used to provide the communication path. The AI circuit is initialized as a transparent link (sending host data to target and target data to host, as if it were a preprogrammed UART). The host based debugger is then invoked to commence debugging.

Using the LoadICE to set AI in to transparent mode:

This method of entering the transparent mode can be invoked from the LoadICE application as the application exits. This way you can use LoadICE to down-load the code and immediately go to your debugger. All the debugger that currently support PromICE, such as Turbo, (Paradigm Debug/RT) SDSI's SingleStep; SSI's CV/Tools, MRI's XRAY, all use the AI as a transparent link between the debugger front end in the host and the debug monitor running in the target system ROMs.

You will need to include the following specifications to LoadICE, either in the initialization file or on the command line. The address should be computed at the location of the AI as mapped into the master emulation module.

```
ailoc addr,baud-rate,[break-char,hints]
-a addr,baud-rate[,break-char,hints]
```

The first two arguments are required and specify the address of the AI interface and the baud rate at which to program the serial link.

You must carefully determine the address of ailoc. Generally this is a label in the monitor code and can be found in some map. The ROM space occupied by the AI interface is filled with 0xcc. The address you specify here is the address where the target is going to access ailoc. This should be specified as an offset from the beginning of the ROM space. So if the target addresses the ROMs at 0x8000 and ailoc is at 0x8008 then specify the address in ailoc command as 0x8.

This address is normalized (i.e. divided by two if word=16 and divided by 4 if word=32) by LoadICE to address within the master ROM module of the unit containing the AI option.

The AI maps into four contiguous bytes of ROM within the master module of the unit containing the AI. Therefor the ailoc in the target address must always be on a four byte boundary. I.e. the lowest two bits of ailoc must always be zero. If your target is a 16 bit system then the master is emulating only one of the ROMs and hence the ailoc offset must be on an octal boundary inorder be properly addressed through the master ROM module.

You may be emulating ROMs on a 32 bit system or otherwise using multiple PromICE units. If you are wishing to use the AI in one of the units, the ailoc command will automatically set the other PromICE units in the daisy chain to pass through mode.

On the newer units with newer micro code, the RUN light will begin to flash rapidly when the AI circuit is properly armed. It will switch to blinking twice every 2.3 seconds when the target accesses the AI. This can be used as diagnostic feed back.

The optional arguments to ailoc allow you to specify a break character and the last argument allows you to break the link by toggling the DTR or the INIT (on parallel port) signals. Here is some more detail on breaking the link out of transparent mode.

The PromICE units can be reset from the host by toggling the DTR line (called host interrupt). If LoadICE conditions the AI to be a transparent link, when the debugger runs, very often it will open the serial link and assert DTR. This will reset the PromICE unit and destroy the setup. If you specify the 'hints' parameter, the PromICE is conditioned to ignore as many of the host interrupts as specified by this parameter. The break character is then ignored and the unit will exit the transparent mode if it is power cycled, or the reset count is exhausted. Without this specification, the specified break character (or null if none specified) will break the transparent mode.

When this specification is (ailoc) present, LoadICE will first send the break character so it can get the PromICE to do a reset, It will also toggle the DTR line at an interval to break out of that mode and proceed with down-loading the files. At the exit it will send a command to the PromICE to recondition the serial link and set the AI to the proper mode.

Using TTY interface to put AI in transparent mode:

The AI option can be configured as a transparent link, by manually using the '.a addr bchr' command of the PromICE tty interface. However, it is often desired to run the LoadICE application to down-load the monitor and have the AI link be set in transparent mode. For both cases you will need to determine the address where the AI circuit is mapped. This is the four byte area in the ROM where the AI circuit's registers will be mapped. This area is called the ACB, or the AI Control Block and consists of four contiguous bytes on a quad boundary within the ROM space emulated by the master module. You should look at the data generated by your cross development tools for computing the proper location for the ACB. In the module RV68K.A68 the variable AILOC defines the position of the ACB. If you have a system that is 16 or 32 bit wide the ACB offsets also change by a factor related to wordsize. Effectively, the address given to the PromICE is the address seen by the ROM being emulated by the master module. We will describe both methods. First the manual method.

1) Use a terminal emulator, or your host based debugger and talk directly to the PromICE. This can be accomplished by sending a couple <CR> characters to the PromICE. When the prompt from the PromICE is received, set the mode as follows:

```
.m 68 ss
```

The mode is changed to '68' which sets it to "no prompt; no echo; no break" from tty mode. You may choose different values to set the mode to a desirable value.

The 'ss' value is the emulation size of the ROM you are emulating encoded as a hex number. The upper nibble sets the size for the slave module and lower nibble for the master. Usually the two numbers will be the same. They are encoded as follows.

1-2k; 2-4k; 3-8k; 4-16k; 5-32k; 6-64k; 7-128k; 8-256k; 9-512k; A-1m

If you are emulating a pair of 27512s, the value will be 66, and the whole command will be '.m 68 66'

Since part of the mode command argument was to turn off the echo, you will not see the '66' part or any other input you type until you change the mode back to the echo input and display the prompt. Effectively, you want to condition the PromICE to operate as close to a transparent link as possible. Now you will instruct it to enable the AI as a tty; which is done by the following command: (again, it will not be echoed on your terminal emulator)

```
 a addr [bchr]
```

You will specify the address within the master ROM module where the AI circuit is mapped. Or in a 0x03 at this address, i.e. the low two bits are always specified as 1s. and optionally specify a break character. If you set the 'NoTTBreak' bit in the mode previously then you don't need this option, the default value for the break character is a null (0x00) or control-shift-2 on the keyboard.

The AI will now go into the transparent tty mode. Any characters you send to the PromICE will be sent to the target system and any data from the target system will be sent to you. You can now invoke the debugger. If 'NoTTBreak' option is *not* in effect, then a break character will cause the PromICE to exit to the transparent mode. The target system must be running and reading the host data for the break character to break the link, i.e. data must be flowing through the link for the PromICE to break the link on a break character.

12. PromICE Internal Memory Addressing Scheme

The PromICE can address up to 1 megabyte of memory per module. Master and slave modules are addressed by switching the internal circuit to select either module. There a total of 20 address lines required to access the 1 megabyte of memory. When less than 1 meg of memory is present the higher address lines are pulled high. The exact reason for this is that the micro controller in the PromICE can set its I/O lines to off state and the internal pull up resisters will pull the signals up. The emulation size switches on the back of the unit let the user connect those lines that are supplied by the target system through the PromICE internal bus. The unused address lines will remain pulled up. Therefore, internally the PromICE addresses memory by using this map:

address 0	last address	Emulated ROM size
1F F8 00	1F FF FF	2K
1F F0 00	1F FF FF	4K
1F E0 00	1F FF FF	8K
1F C0 00	1F FF FF	16K
1F 80 00	1F FF FF	32K
1F 00 00	1F FF FF	64K
1E 00 00	1F FF FF	128K
1C 00 00	1F FF FF	256K
18 00 00	1F FF FF	512K
10 00 00	1F FF FF	1M

When using a terminal or a terminal emulator to talk to the PromICE the actual address used by the PromICE is displayed. An example would be a unit equipped with 128k bytes of memory when emulating a 32k byte ROM will display address 0F8000 for address zero. This will compute to be the highest one addressed of the four chunks of 32k that comprise the 128k unit.

This also explains why the LoadICE software must be told exactly what size ROM you are emulating. It ensures that the data is loaded at the proper place in the internal space. Also, the switches on the back of the unit must be set for proper emulation size, otherwise, the target system will access the wrong emulation space.

13. PromICE ASCII/TTY Interface Specifications

When the PromICE detects '0D' (carriage return) code at the auto baud time, it decides that the user is communicating from a terminal. No daisy chaining is supported in this mode. The PromICE will take commands interactively from the user and will directly load hex records. Only Motorola S1, S2 and S3 records and Intel hex (standard only) records are supported. All commands begin with a '.' Following, are all the valid commands:

.a address [bchr]	operate AI as a tty with UART mapped at the given address (control-shift-2 (null) or 'bchr' to exit).
.d address data	deposit the data at the given address.
.e address range	examine one or more bytes at given location. For slave unit the high nibble contain a 1.
.g	go into emulation mode.
.h	print a very short help message
.m nn ss	report or set mode and size, also reports size*
.p address data	do AI patch function with given data to be patched at given address.
.q	quit emulation, go in to load mode
.r nn	reset the target for nn•8.9 milliseconds
.t	test emulation RAM
.f dd	fill RAM with 'dd'
.x	switch from master to slave or vice-versa
q	*restart interface*

*mode command argument nn is bit encoded as follows:

Slow-NoPrompt-NoEcho-LoadSlave|NoTTBreak-x-x-LoadMaster

Slow - The PromICE will transmit data back at full baud. To avoid overflow on a host computer set this bit to have the PromICE transmit data at ~9000 baud rate.

NoPrompt - Set this bit to stop the PromICE from displaying the prompt. This is useful when the AI option is in transparent tty mode.

NoEcho - Set this bit to stop the PromICE from echoing your input. Another useful command for AI tty mode of operation.

LoadSlave - Set this bit to put the slave module in load mode.

NoTTBreak - When the AI is in the transparent tty mode, a break character (definable in .a command to be other than a *null*) will break the PromICE out of that mode. Set this bit if you don't want any break characters. This will be handy if you are doing communication with the target in binary data. PowerCycle the PromICE to break out of this mode.

x - unused bit reserved for future.

LoadMaster - Set this bit to put the master in load mode.

The memory and emulation size (in K-bytes) are encoded as follows:

1-2k; 2-4k; 3-8k; 4-16k; 5-32k; 6-64k; 7-128k; 8-256k; 9-512k; A-1m; B-2m

In the above commands all numbers are in *hex*. The *examine* and *deposit* commands will open the next location if a *line-feed* is typed instead of a *carriage-return*. The *Test* command can be interrupted by hitting any key. typing 'q' will restart the interface allowing re-established communication links at a different baud rate or via *LoadICE* program.

The AI board, if present in the system, can be operated from the terminal interface to emulate a transparent link to the target. If you have a piece of code running on the Target system that can communicate to a serial line, then it is trivial to interface it to operate via the AI system. The '.a' command is used to specify the mapped address of the AI and enable this transparent mode. For all practical purposes the PromICE serial port can be used directly to communicate with the target system. You can also specify a break character for the PromICE to break the AI out of this loop. The break character by default is a null (0x00).

There is no particular reason why the terminal interface cannot be operated from the parallel port when it is used bidirectionally. However, it requires that some host based terminal emulator should be able to handle the bi-directional parallel protocol.

14. PromICE Binary/Host Interface Specifications

PromICE communicates with the host system over the serial RS232 interface or the parallel Centronix compatible interface. The serial interface supports daisy chaining PromICE units for attaching and controlling multiple units from a single host serial port. The parallel port may also be used in conjunction with the serial arrangement for down loading of data faster than over the serial port alone.

14.1. PromICE RESET

PromICE is in idle state on power up or after pressing the reset switch on the back of the unit. In this state the PromICE unit is emulating. A target system can access the ROM. PromICE also asserts the reset to target on the back panel.

PromICE can also be brought into the idle state from any other state by toggling the DTR signal on the serial interface or the INIT signal on the parallel interface. PromICE will enter the idle state but will not assert the reset pins on the back panel, as it does if it is powerup or manually reset.

14.2. Establishing communication with the host

14.2.1. Determining the serial baud rate

Once the PromICE is in the idle state the host may establish communication with it. Host starts by sending to the PromICE the auto-baud character (0x03). The auto-baud character is sent on an interval (1/10 second) until it is received back at the host. The PromICE has determined the baud rate now..

On the parallel port only one auto-baud character is sent and the PromICE echoes it back immediately. This also establishes the parallel port as a bi-directional connection to the host. PromICE sends data back by using the parallel port in a pseudo bi-directional mode by sending a nibble of the data at a time over the error and status signals of the parallel port.

14.2.2. Establishing unit IDs

The host after receiving the 0x03 character will send the PromICE the ID packet which consists of a 0x00 as packet identifier and the first ID which is initially 0x00. PromICE will send back the 0x00 as soon as it receives it and then will receive the ID. The ID is incremented by 1 and if a slave module is present then the ID is incremented again and is sent out as the next available ID.

When multiple units are daisy chained to the serial port, the next unit in the chain will receive the ID packet and will assign and increment the ID number in the ID packet. Eventually the host will receive the ID packet back with the ID number in the packet reflecting the next available ID. This number establishes the number of emulation modules (master and slaves together) present on the serial or the parallel port.

14.2.3. Communicating with PromICE

Once the IDs are established, the host communicates with the PromICE modules by using their respective ID numbers. The communication takes place in form of command packets with ID number in each packet. If a unit receives a packet with an ID that does not match its master's or it's slave's ID, it is forwarded to other units down the chain.

All command packets have the following format:

```
ID | COMMAND | DATA_LENGTH | data - data - .......|
```

where ID is the binary ID of the module the command is being sent to and COMMAND is the command byte being sent. DATA_LENGTH is the binary byte containing the number of data bytes to follow. This way up to 256 data bytes can be sent in a single packet with DATA_LENGTH=0x00 meaning 256

PromICE modules respond to the command packet by performing the command and generating a response packet. The response packet is identical to format with the command packet. The DONE bit is set in the command byte to indicate that the command has been completed. An ERROR bit may also be set if the command failed. The data in the response packet either contains data requested as a result of the command or a completion code. On most successful commands the completion code is generally 0x00.

14.3. COMMANDS and RESPONSES

Here are all the commands that the host may send to the PromICE and the possible responses the PromICE may generate. The commands are only four bits long. The high nibble of the command byte contains modifier that have different meanings for different commands. However, one modifier is common to all the commands and it is the NO RESPONSE bit.

Any command masked with 0x20 will not return a response. The only exception to this rule is the RESTART command, no response is ever generated.

14.3.1. 00: LOAD POINTER

Loads a 24 bit point for read and write operations. The pointer is incremented as READ and WRITE commands are executed. But the internal pointer is only 16 bits long. The extended address byte acts as a bank address. It is the host's responsibility to reload the pointer everytime it goes to all zeros.

```
ID 00 03 EX HI LO
ID 80 01 00
```

14.3.2. 01: WRITE DATA

Writes up to 256 bytes of data in a single command. The response data is an exclusive OR of all the data bytes.

```
ID 01 NN DD.DD.DD....
ID 80 01 CK
```

14.3.3. 02: READ DATA

Return a response packet with up to 256 bytes of data.

```
ID 02 01 NN
ID 82 NN DD.DD.DD.....
```

14.3.4. 03: RESTART

Restart the PromICE interface. LoadICE sends this command to each unit when it exits. No response is generated. When the AI is being programmed for transparent mode (see *ailoc* command of LoadICE), this command carries additional information to either program the PromICE AI channel for transparent mode of operation, or set it is pass through mode as passive member of a multi unit configuration. (See appendix on putting AI in transparent mode)

Standard format:

```
ID 03 01 00
```
NO RESPONSE IS GENERATE TO THE RESET COMMAND

Request to enter AI transparent mode, AILOC address is specified as EX, HI and LO. Unused bits of the AILOC must be masked for size of ROM that is being emulated, as well as for the size of SOCKET that the target is addressing. The interface to work correctly the internal comparator which is loaded with the fill 24 bit address must match it with the target access cycles. BR is the baud rate for the serial link. If BR=FF then use parallel port in bi-directional mode for link to the host. BR is encoded as follows:

BR=00	1200 baud
BR=01	2400 baud

BR=02	4800 baud
BR=03	9600 baud
BR=04	19200 baud
BR=05	57600 baud
BR=FF	PARALLEL PORT

```
ID 03 05 96 EX HI LO BR
```
NO RESPONSE IS GENERATE TO THE RESET COMMAND

Request to enter pass through mode. BR is the baud rate ORed with 0x80. BR values are same as above

```
ID 03 01 80|BR
```
NO RESPONSE IS GENERATE TO THE RESET COMMAND

14.3.5. 04: SET MODE

Sets the operating mode for PromICE. It is most often used to switch the units between load and emulate mode. However, MODE command can be used for many different things. The following are all the things MODE can do:

• Set various modes and control parallel port mode of operation

```
ID 04 01 MM
```
ID 84 01 SS

Where MM is decoded as follows:

Bit 7	FAST	run the serial link transmitter at full baud rate
Bit 6	XTND	send target status in each response packet
Bit 5	MORE	more mode data in the next byte
Bit 4	PPON	turn on the parallel port for data receive
Bit 3	PPOFF	turn off the parallel port for data receive
Bit 2	PPFAST	do fast parallel port transfers (no back_ack)
Bit 1	AUTO	enable auto-reset (disabled by RESET 0)
Bit 0	LOAD	put the unit into load mode

If MORE was set then another byte of data follows and command format is

```
ID 04 02 MM MO
```
ID 84 01 SS

Where MM is same as above and MO is decoded as follows

Bit 7 NOLIGHT	disable PromICE blinking RUN light
Bit 6 NOTIMER	disable internal timer

```
Bits5-0         UNSUED
```

SS is the real size of the memory in PromICE units. The size is only four bits long. If both the nibbles contain data then the response is from a master module of a duplex PromICE unit. S is encoded as follows and indicates the real amount of memory on the module in concern.

S=1	2K Bytes	2716
S=2	4K Bytes	2732
S=3	8K Bytes	2764
S=4	16K Bytes	27128
S=5	32K Bytes	27256
S=6	64K Bytes	27512
S=7	128K Bytes	27010
S=8	256K Bytes	27020
S=9	512K Bytes	27040
S=A	1M Bytes	27080
S=B	2M Bytes	27160

• Set or clear AI special bits for debugger control

```
ID 14 02 AI DD
```
`ID 84 01 00`

where AI is the address of the special bit and DD is 0 or 1 for clear or set it. AI values can be as follows:

AI=BD	overflow bit in AI status reg (clears automatically)
AI=BE	DD=0 enable target writes to slave, 1 disable writes
AI=BF	DD=0 enable target writes to master, 1 disable writes

• Set AI transparent mode parameters

```
ID 44 03 BB BC SI
```
`ID 84 01 00`

Where BB=00 if no break character in effect (binary transparency), BB non zero then use BC as the break character and SI is number of DTR (or INIT) toggles to skip. If SI=FF ignore all toggles, host can still break the transparent mode by toggling the DTR or INIT about 4 times in less than 30 seconds. NOTE: the PromICE does not go into transparent mode until RESET command is issued.

• Set new emulation size

Allows the host to set a new emulation size, usually the host does not set emulation size, it rather uses address masks to address the memory so that the target and the host will see the same area of memory.

```
ID 84 01 EE..
ID 84 01 SS
```

Where EE is the size code for the new emulation size (you can specify both the slave and master size at the same time). The SS in response is the new emulation size. The normal MODE command will still return the SS in its response as the real memory size

14.3.6. 05: TEST RAM

Tests the PromICE memory. It can also be used for filling the memory with a fill character.

• Test the PromICE memory PC times

```
ID 05 01 PC
ID 85 01 00
ID 95 03 EX HI LO
```

If the test fails then the longer response is generate with the ERROR bit set. The 24 bit address is returned where the error occurred. The address will have the high order bits masked depending on the size of the memory.

• Fill the PromICE memory FC character

```
ID 15 01 FC
ID 85 01 00
ID 95 03 EX HI LO
```

14.3.7. 06: RESET TARGET

Generates a reset pulse to the target. It can also be used to turn off auto-reset. If RL=0 then auto-reset flag is turned off. No reset is generated and as unit switches from load to emulate and vice-versa via the MODE command, no reset is generated then either. MODE command can be used to turn the auto-reset back on.

If RL is non zero then a reset pulse of RLx8.9 milli-seconds is generated on the *int-* and the *int+* pins on the back of the PromICE unit.

```
ID 06 01 RL
ID 86 01 00
```

14.3.8. 07: SPECIAL READ/WRITE

Does a read or write of a single byte by using the REQ and ACK pins on the back of the PromICE as handshake lines. The target bus is requested by asserting REQ and when ACK is received, the unit is put into load mode and a byte of data is read or written. The REQ is then released and the unit is put back into load mode. The read operation takes a minimum of 7 micro-seconds and write takes a minimum of 13 micro seconds. The wait increments are 4 micro seconds length. A time out error is returned after a maximum wait of 2.3 seconds.

• For doing a read: dd going in is dumped and *dd* in response is data read. If a time out error occurs (timed out waiting for ACK) then response FD indicates timeout error.

```
ID 47 01 dd
ID 87 01 dd
id 97 01 FD
```

• For doing a write dd is the data written and response is zero or FD as above

```
ID 07 01 dd
ID 87 01 00
id 97 01 FD
```

If the command contained the 0x10 bit or the GXINT was set by MODE then an interrupt to the target is also generated on the back pins of the PromICE, if the operation was successful (i.e. no timeout error).

14.3.9. 08: ESTABLISH LINK WITH TARGET

Establish a communication link with a target monitor supporting the AiCOM or the PiCOM interface of the PromICE.

• For PiCOM protocol, timeout error is returned if failed to initialize due to no ACK

```
ID 88 03 EX.HI.LO
ID 88 01 00
ID 98 01 FD
```

• For AiCOM protocol, error return is made when the AI option is not installed

```
ID 08 03 EX.HI.LO
ID 88 01 00
ID 98 01 FF
```

If the command contained the 0x10 bit (INTT) or the GXINT was set by MODE then an interrupt to the target is also generated on the back pins of the PromICE, if the operation was successful (i.e. no timeout error).

14.3.10. 09: WRITE MESSAGE

Send a message to the target system. A maximum of 39 (decimal) bytes long message can be sent over the AiCOM or a 256 bytes long message over the PiCOM interface. The AiCOM message in internally buffered and then sent to the target asynchronously. The PiCOM message is inserted in shared buffer in the ROM space by using the mechanism described in

SPECIAL READ/WRITE (07) command above. *nn* is the data count and *dd* are the data bytes

• For PiCOM, if error then standard response *sr* and interface status *es* are returned

```
ID 89 nn dd.dd.dd.....
ID 89 01 00
ID 99 02 sr ee
```

• For AiCOM, if error then standard response *sr* and interface status *es* are returned

```
ID 09 nn dd dd.dd.....
ID 89 01 00
ID 99 02 sr ee
```

If the command contained the 0x10 bit (INTT) or the GXINT was set by MODE then an interrupt to the target is also generated on the back pins of the PromICE, if the operation was successful (i.e. no timeout error).

14.3.11. 0A: READ MESSAGE

A message is read from the target if there is one. The AiCOM message from the target is limited to 39 bytes and the PiCOM message can be up to 256 bytes long. The PiCOM message is read from the shared buffer in the ROM space by using the mechanism described in SPECIAL READ/WRITE (07) command above.

• For PiCOM, then the requested data length is ignored and available data is sent. *nn* is the length of the target message. If error then standard response *sr* and interface status *es* are returned

```
ID 8A 01 dd
ID 8A nn dd dd dd.....
ID 9A 02 sr ee
```

• For AiCOM, the request data length is ignored and the message from the target is sent, *nn* is the message length. If error then standard response *sr* and interface status *es* are returned

```
ID 0A 01 dd
ID 8A nn dd dd dd......
ID 99 02 sr ee
```

If the command contained the 0x10 bit (INTT) or the GXINT was set by MODE then an interrupt to the target is also generated on the back pins of the PromICE, if the operation was successful (i.e. no timeout error).

14.3.12. 0B: SET COMM MODE:

Sets the various conditions and parameters for AiCOM and PiCOM communication. Mode is only changed if the CHANGE bit is set in the command else the command is used to simply generate an interrupt to the target under program (host) control.

• Simply interrupt the target. if MM = 1 then another byte LL follows and defines the target interrupt length as LLx2 micro-seconds. The maximum can be 255x2 or ~ half a milli second. If MM=0 the default is 28 micro seconds or whatever the length was set previously.

```
ID 0C 01 MM
ID 0C 02 01 LL
```
ID 8C 01 00

• Set other mode bits: PiCOM - if MM has 01 bit on the LL is the target interrupt length. same as above.

```
ID CC 01 MM
ID 0C 02 MM LL
```
ID 8C 01 00

The mode bits in MM are as follows:

Bit 7 LINK - 1 = bring the link up; 0 = bring the link down
Bit 6 GLOBAL ASYNC READ: allows us to generate a read response whenever target has data for the host, this way the host does not have to issue a CREAD command. However, the host better have some way to reliably accept asynchronous data.
Bit 5 REQ HIGH: Specifies that the REQ signal (PiCOM only) is high asserted. REQ is immediately lowered.
Bit 4 ACK HIGH: Specifies that ACK is high asserted input.
Bit 3 UNUSED
Bit 2 GLOBAL READ INT: allows us to interrupt the target any time a read command completes (i.e. we have taken target data and he can send more).
Bit 1 GLOBAL COMMAND COMPLETION INT: this allows us to interrupt the target on any communication command's (cmds 07-0B) completion.
Bit 0 TARGET INTERRUPT LENGTH: (LL) The default interrupt length when DOXINT causes target interrupt is 28 micro-seconds. However it can be set by CMODE in increments of two micro-seconds, from minimum of 6 to maximum of 516 micro-seconds.

If the command contained the 0x10 (INTT) bit or the GXINT was set by MODE then an interrupt to the target is also generated on the back pins of the PromICE, if the operation was successful (i.e. no timeout error).

14.3.13. 0C: AI PATCH

Allows patching of the ROM data with the help of target monitor by using target read operation alone. This allows setting of break-points in those targets where ROM write cycles can not be done. NOTE: All debuggers currently supported by the PromICE use the wrt pin on the back of the PromICE and require that the target be able to do write cycles to the ROM space in order to set and remove break points in the ROM space. AI interface provides special circuit for the host to be able to disable the writes to ROM under program control. This way the writes can be enabled when the debugger is in control and can be disabled when the application being debugged is running. This way the ROM space can be protected from potential bugs in the application that may otherwise trash the ROM with stray writes. However, this feature is also not currently exploited by the third party debuggers.

Patch ROM - 24 bit address is given and DD is the data with TT as the time out value. TT=0 means to timeout to occur. Units of TT are 8.9 milli seconds.

```
ID 0D 04 EX.HI.LO.DD TT
```
ID 8D 01 00

14.3.14. 0D: UNUSED

Reserved for hardware trap function.

14.3.15. 0E: UNUSED

Reserved for interface extension.

14.3.16. 0F: GET VERSION

Returns a 4 byte micro-code version #. LoadICE uses the version number to determine what features the PromICE supports. This command can also be used to receive the 32-bit serial number.

- Return version:

ID 0F 01 00
ID 8F 04 V1.V2.V3.V4

- Return serial number

ID 1F 01 00
ID 8F 04 S1.S2.S3.S4

15. Index