iRMX®
System Configuration and Administration

RadiSys Corporation
5445 NE Dawson Creek Drive
Hillsboro, OR 97124
(503) 615-1100
FAX: (503) 615-1150
www.radisys.com
07-0569-01
December 1999

December 1999

Quick Contents

Notational Conventions

This manual uses these conventions:

- All numbers are decimal unless otherwise stated.
- Bit 0 is the low-order bit unless otherwise stated.
- `Computer output and input is printed like this.`
- **System call names and command names appear in bold.**

Pathnames are given by logical name (such as *:config:*) and forward slash (/), as seen from the iRMX® Human Interface (HI) prompt. Logical names correspond to the following DOS pathnames:

Logical Name	**DOS**	**iRMX OS**
:rmx:	\rmx386\	:sd:rmx386/
:config:	\rmx386\config\	:sd:rmx386/config

See also: Logical names, *Command Reference*
Default file structure, *Installation and Startup*

iRMX system calls use prefixes to designate functions or OS layers:

- When referring to the system calls that begin with **rq**, this manual uses a shorthand notation and omits the prefix. For example, **s_create_file** means **rq_s_create_file**.
- When referring to system calls that begin with **rqe**, this manual spells out the complete names, including the **rqe** characters, for example **rqe_create_io_job**.

Note
Notes indicate important information.

CAUTION
Cautions indicate situations which may damage hardware or data.

Contents

8 Browsing/Cross Debugging an iRMX III.2.3 System from a Windows NT Host

A Keyboard and Console Information

B Multibus II Downloader

C ATCS/279/ARC/450 System Jobs

Tables

Figures

Overview of System Configuration 1

Introduction

This manual describes how to configure the iRMX® III, DOSRMX, and iRMX for PCs operating systems. Some configuration processes, such as adding users and terminals, are the same for all three operating systems (OS). Other kinds of configurations may be different from one OS to the next:

- You can configure any of the OSs with the Interactive Configuration Utility (ICU). You can configure almost every aspect of the OS with the ICU, including support for a variety of hardware components and bus architectures. You can also use the ICU to build your application into the bootable OS.

  See also: *ICU User's Guide and Quick Reference*; *Programming Techniques and Tools*

- iRMX for PCs runs on PC-compatible platforms, but without DOS interoperability. You can change the *rmx.ini* configuration file to fine-tune this OS without using the ICU; changes take effect during load-time. Each OS layer reads entries from *rmx.ini* during initialization, overriding preconfigured values.

- DOSRMX runs on PC-compatible platforms with DOS interoperability. You can change the *rmx.ini* configuration file to fine-tune this OS without using the ICU; changes take effect during load-time. Each OS layer reads entries from *rmx.ini* during initialization, overriding preconfigured values.

- For all versions of the OS you can also do run-time configuration with the **sysload** command. In DOSRMX and iRMX for PCs, the main action of the *:config:r?init* file is to submit the *:config:loadinfo* file, containing **sysload** commands for loadable jobs and drivers.

Default Configuration Files for the iRMX® III OS

The iRMX OS installation provides ICU definition files (*:icu:*.bck*) that define the default configuration for various hardware platforms. Use one of these files as a starting point in the ICU for your custom configuration.

See also: List of *.bck* definition files, *ICU User's Guide and Quick Reference*

Reasons to Change the Default Configuration

Here are some reasons for changing the default configuration:

Users — There are two users defined for the system: *Super* and *World*. The Super user is the system administrator, with access to all system files and devices. The World user has restricted access to files installed on an iRMX file system (all users have full access to files on a DOS file system). If more than one person uses the system, create a separate user logon for each. In the iRMX file system you can protect files and/or directories for each user. Even on a single-user system, you may want to log on for general work as a different user than World or Super, to keep your work distinct from these default users. Use the Super logon only for system administration duties.

Terminals — A set of terminal devices and types is defined by the OS. In iRMX for PCs, only one terminal is initialized by default: the main console device, **con** (system console). . In DOSRMX, only one terminal is initialized by default: the main console device, **d_cons** (DOS console When configuring terminals, initialize other devices or change keyboard and screen characteristics to match your preferences. For example, you can:

- Specify a different terminal type for a serial port
- Define a new console device, with different keyboard characteristics than the default
- Set up the COM1 and COM2 ports for use with a terminal
- Specify a terminal driver made available by a loadable device driver
- Associate a terminal with a particular user

See also: Terminal definitions, Chapter 2

Network — A set of jobs, either loadable or ICU-configurable, provides networking. You can choose from a variety of network jobs, depending on your system architecture.

Loadable Jobs and Drivers — Using loadable jobs and drivers enhances iRMX for PCs and DOSRMX during run-time, while setting up the OS to work with your hardware configuration. You can add your own application as one or more loadable jobs. Non-standard hardware is supported by loading custom device drivers. The OS treats loaded jobs and drivers as child jobs of the HI layer.

See also: Loadable jobs and drivers, Chapters 3 and 4, in this manual

System jobs, Chapter 6, in this manual,
Driver Programming Concepts

Load-time The *rmx.ini* file contains entries that match settings preconfigured into iRMX for PCs and DOSRMX. Modify the existing *rmx.ini* entries to fine-tune your configuration.

ICU-configurable Jobs
In any of the OSs, use jobs and drivers in loadable form during the debug phase, then configure them into the OS as *system jobs* using the ICU, along with layers of the OS.

Soft-Scope You can use the Soft-Scope debugger after configuring in, or loading, the C library.

See also: Chapter 8

Configuration Methods

To configure an iRMX system, run the ICU with a definition file that matches your hardware. The definition file contains parameters that define the current configuration. The ICU presents these parameters one screen at a time. Modify the system by changing parameter values. Then use the ICU to generate a new bootable OS.

See also: *ICU User's Guide and Quick Reference*

In iRMX for PCs and DOSRMX, you can configure the OS by editing configuration files and rebooting the system. If you install on the DOS file system, you have the choice of editing the files in DOS (you may have to change the attributes of hidden files), or in the iRMX OS. Use a text editor such as DOS EDIT, or AEDIT (supplied with the iRMX OS). If you use a word processor (e.g., WordPerfect), be sure to save the files as DOS text. You can also use the ICU to generate a custom system or modify an existing definition file for iRMX for PCs.and DOSRMX.

The System Administrator

You become the system administrator by logging on as Super or by invoking the **super** command after logging on as another user. The default password is `passme`. The Super user has user ID 0, providing all access rights to all files on the system.

See also: Logging on, *Installation and Startup*

As system administrator you can change the Super user password with the **password** command. You also have the right to:

- Change the access rights to any file.

- Read all data files and list all directories.
- Detach devices attached by any user.
- Delete any user from the system.
- Invoke the **shutdown** command.

The Named file driver maintains files installed on an iRMX file system. If you install on an iRMX file system, configuration files are restricted to access by the Super user only.

You edit configuration files from the iRMX prompt while logged on as Super. In three cases, you must use only an iRMX utility to modify a configuration file (don't edit the file directly), because encrypted passwords are used:

- When adding or removing users, use the **password** command.
- When configuring subnetworks within an OpenNET network, use the **modcdf** utility.
- When creating the *ccinfo* file used by the Remote Boot Server job, use the **bcl** utility.

Configuration Files

Table 1-1 lists the iRMX configuration files, which are in the *:config:* directory..

Table 1-1. The iRMX Configuration Files

Filename	Purpose	Modify With	Takes Effect
cdf	Client definition file (CDF), verifies networked systems	**modcdf** command	Immediately
udf	User definition file (UDF)	**password** command	Immediately
user/<username>	User attributes file	**password** command and/or text editor	Next logon
terminal(s) [1]	Terminal configuration file	text editor	On reboot [3]
termcap	Terminal definition file	text editor	On reboot
r?init [2]	Hidden system initialization file	text editor	On reboot
loadinfo [4]	System load file	text editor	On reboot
rmx.ini [5]	Overrides iRMX for PCs configuration defaults	text editor	On reboot
r?init2	Hidden system initialization file	text editor	On reboot

1 On the iRMX file system, this file is named *terminals*, on the DOS file system it is restricted to 8 characters.
2 From the DOS prompt this is a hidden file with the name *init*.
3 You do not need to reboot the system if you edit the files under DOS and have not yet started DOSRMX.
4 This file is submitted by *r?init* at reboot.
5 This applies only in iRMX for PCs and DOSRMX.

Backup copies of these files are installed in the *:config:default* directory. Save the backup copies to preserve the original syntax.

HI Initialization and Logon

Configuration files run automatically on system bootup, and during logon and logoff. Understanding the Human Interface (HI) initialization helps you decide which configuration changes to make and how those changes affect the system. The HI uses several configuration files while performing these steps:

1. Creates the Super and World users (performs all subsequent steps as Super).
2. Sets the system date and time to values read from the global battery-backed clock.
3. Reads the *:config:terminals* file in preparation for terminal initialization later.
4. Executes command lines in *:config:r?init*. The execution occurs in the context of the first terminal defined in the *:config:terminals* file. You can add commands to the *r?init* file directly. The default command line is:

   ```
   submit :config:loadinfo over :config:loadinfo.log
   ```

   This executes command lines in *:config:loadinfo* (and sends output to *:config:loadinfo.log*). The default *loadinfo* file contains a series of **sysload** commands that load jobs and file and device drivers. Comment characters (semicolons) in the file determine which commands execute. Adding **sysload** commands enables you to load your application, or any other commands you want to run at this point.
5. Brings up terminals designated in *:config:terminals*. This file defines each terminal as either *static* or *dynamic*. A static terminal has a unique user name and ID associated with it, regardless of how many different people use it. There is no logon or logoff; rather, the HI starts the user job and displays the command interface prompt. At dynamic terminals, the HI prompts for a logon before doing the same.
6. Concurrently with Step 5, executes any commands in *:config:r?init2*, if it exists.

Note

You can change the system configuration file (SCF) from the default of *:config:r?init* to any other file in the *:config:* directory using one of these methods:

- SCF parameter on the HI screen of the ICU
- SCF parameter in the *rmx.ini* file (see Chapter 5)
- `rq_hscf` BPS parameter (see *MSA for the iRMX Operating System)*

If you change the SCF, the HI does not execute commands from *:config:r?init2* in Step 6. Instead it attempts to execute from a file with the same name as the new SCF, but with the number 2 appended to the filename.

The HI runs the SCF (and any files submitted by it, such as *loadinfo*) as the Super user, so they affect all users in the system. The **submit** command that runs at this time is an internal HI version of the command, not the CLI version. So there is no CLI support, including support for command aliases.

See also: **submit**, CLI, *Command Reference*

What Happens During Logon

When a user logs on, the HI scans the *:config:udf* file, verifies the password, gets the user ID, and:

1. Checks the appropriate user attributes file (*:config:user/<username>* where *<username>* is the logon name). This file tells the HI which initial program to load (iRMX CLI or a custom command interface you have written), and other information related to this user.
2. Invokes the initial program. The default CLI displays a sign-on message, runs the logon command file *:prog:r?logon* that submits the user-modifiable *:config:alias.csd* and *:prog:alias.csd* files, and then issues a command line prompt.

End a session at a dynamic terminal with the **logoff** command. The CLI submits the *:prog:r?logoff* file. As with the logon file, users can modify their own individual logoff file.

Logon Error Messages

The HI allocates a memory pool for each static user and for each dynamic user who logs on. If the HI cannot allocate the amount of memory requested, but has enough memory to activate the CLI, the HI allocates all the free memory available and displays this warning on the user's screen:

```
*** WARNING: The system cannot provide the minimum memory
*** you requested.  You will come up with all the memory
*** that is currently available in the system.  If
*** this is a problem, contact the system administrator.
```

If there is not enough memory to create the user's job, this message appears:

```
*** HI LOGON ERROR: Insufficient user memory available at
*** this time.  Try to logon again later.
```

The Resident/Recovery User

You can use the ICU to specify a resident/recovery user, with Super user access rights. The resident/recovery user supplies access to the system if the HI fails to initialize. There is no logon for this user. If the HI detects a problem while initializing terminals, the resident/recovery user is enabled on the first terminal defined in the *:config:terminals* file, and the HI displays this error message:

```
*** Recovery User created
```

Logging On and Remote File Access

If your system is configured to include the iRMX-NET client job, any user on the system who gains access to the HI by logging on, automatically becomes a verified user. In an OpenNET network system, a verified user can access files on remote systems through iRMX-NET. User IDs for static terminals are not verified users.

Configuring System Jobs and Device Drivers

In any of the OSs, you can configure system jobs and file and device drivers by adding loadable jobs and drivers. You can also configure most system jobs and device drivers in the ICU. Most ICU-configurable jobs and drivers are also provided in their loadable form.

Use the provided loadable jobs and drivers as starting points in the development of your own application system. Use the **sysload** command to load these drivers to allow testing and debug during run time. After development, use the ICU to configure jobs and devices into the OS.

See also: **sysload** command, Chapter 3,
Loadable Jobs and Drivers, Chapter 4,
ICU-configurable jobs, Chapter 6,
IDEVS screen for drivers, *ICU User's Guide and Quick Reference*

Load-time Configuration

In iRMX for PCs and DOSRMX, drivers, and jobs are preconfigured with default values. You can override some of these values with entries in the *rmx.ini* file. Changing values in the *rmx.ini* file changes the configuration. The default *rmx.ini* file contains strings for values that are the same as those preconfigured into the OS. If you change any values, the next time you boot the OS, those values are set as each layer loads.

See also: Default configuration, *Programming Concepts for DOS*

Note

The iRMX III OS does not include the *rmx.ini* file, and loadable jobs in these systems cannot take advantage of such a file. However, in a Multibus II system you can do some load time configuration of an iRMX III system by making changes in the BPS file.

See also: BPS parameters, *MSA for the iRMX Operating System*

❑❑❑

Configuring Users and Terminals 2

Adding Users to the System

Use the **password** command to modify entries in the User Definition File (UDF) and create a user home directory and a *user attributes file*. If you delete a user with the **password** command, only the UDF entry for that user is deleted; all other user files remain intact.

See also: **password**, *Command Reference*, for creating and deleting users

User Definition File

The UDF (*:config:udf* file) contains the logon name, user ID, and encrypted password (dynamic terminals only) for all users. iRMX-NET uses this file to validate user access to the network.

Passwords in *:config:udf* are encrypted to prevent unauthorized access. The **password** command maintains the format of the file and automatically encrypts the passwords. Do not edit the *:config:udf* file directly.

User Home Directory

When you add a user to the system, the **password** command creates a user home directory unless you tell it not to. The home directory is the user's working directory immediately after logon. The default directory created by **password** is *:sd:user/*<*username*>, where <*username*> is the logon name of the new user. In this directory, **password** creates a *prog* subdirectory containing three files: *alias.csd*, *r?logon* and *r?logoff*. These files determine actions that happen automatically when that user logs on and off the system.

See also: What happens during logon, Chapter 1

The **password** command installs *alias.csd*, *r?logon* and *r?logoff* files from copies in the *:config:default* directory. To change the default logon and logoff actions for all users, modify these files before creating users. If you want the same changes to apply to the World and Super users, edit the same files in the *:sd:user/super/prog* and *:sd:user/world/prog* directories.

User Attributes File

For each user on the system, the **password** command creates a user attributes file in the *:config:user* directory. The filename is *:config:user/<username>*. You can modify the file with a text editor to change the attributes. This file has these entries:

`minimum_pool_size`
: Specifies the minimum size, in Kbytes, of the memory pool that the HI assigns to the interactive job for this user.

`maximum_pool_size`
: Specifies the maximum size, in Kbytes, in the user job's memory pool. The difference between the maximum and minimum values is borrowed from the parent job, if necessary.

`max_task_priority`
: Specifies the maximum priority (numerically lowest) that any task associated with this user can have (from 0 to 255). This value determines the priority of the initial task (CLI or a custom interface). Recommended values are a user priority lower (numerically higher) than 141, and a Super user priority of 141. The World user's default priority, and the priority of any user created with the **password** command, is 142.

`default_prefix_pathname`
: This user's default home directory, establishing the *:home:* and initial *:$:* directory. This is normally the *:sd:user/<username>* directory. The directory specified in this field must exist or the user will be unable to access the HI.

`initial_program`
: Pathname of the file containing the user's initial program. If there is no entry here, the iRMX CLI is used as the default.

  You can specify your application program as the initial program for a particular user. Depending on the purpose of the application, it may not need a CLI.

  See also: Chapter 7

For example, here is a user attributes file with the pathname *:config:user/steve*:

```
1024,2048,150,:sd:user/steve
```

The attributes are separated by commas and no spaces. Since a command interface is not specified (there is no final field), Steve's command interface is the iRMX CLI.

User Job Priority and DOS Priority in DOSRMX

In DOSRMX, DOS operations are fixed at priority 254. The default user `max_task_priority` is 142. With these priorities, activity at the iRMX prompt preempts the DOS command interface. The DOS command interface polls the processor, rather than being interrupt-driven, so DOS uses processor time even when idle.

If you set the user `max_task_priority` to 253, DOS and iRMX commands have equal priority, sharing the processor in equal time slices. Although DOS priority is 254, 253 is correct for users because the CLI lowers the maximum priority by one for child jobs and their initial tasks. In other words, commands entered at the CLI prompt operate at a priority of one lower (numerically higher) than `max_task_priority`.

See also: Child jobs, *System Concepts*

The exceptions to this rule are the **submit** and **background** commands. The **submit** command lowers the priority one unit below that set by the CLI, for each invocation (nested **submit** commands each lower the priority by one). The **background** command lowers the priority by three units below that set by the CLI.

If you were to set the user `max_task_priority` numerically greater than 253, the DOS task would preempt all iRMX user commands. For the **submit** and **background** commands, this would occur even for higher user priorities, depending on the nesting level of these commands.

To avoid this situation, the OS prevents any iRMX tasks from using priority 255, forcing them to priority 254. This means:

- iRMX tasks created at priority 254 or 255 can share processor time with DOS
- Commands invoked from the HI at priority 253 or 254 can share processor time with DOS
- The **background** command fails with an E_PARAM exception if invoked by an HI user at a priority lower (numerically higher) than 251
- Background programs cannot share processor time with DOS
- Any iRMX tasks that share the processor with DOS will take longer to execute than if they operated at a higher priority. This is because they operate in round-robin fashion with the DOS task, which is always in the ready state.

Configuring Terminals

Adding or changing information in the following files defines new terminal types or defines how terminals operate:

- Terminal configuration file, *:config:terminals*
- Terminal definition file, *:config:termcap*

Terminal Configuration File

The terminal configuration file, *:config:terminals*, specifies which terminals the HI attaches to the system. This file contains the terminal device names, the terminal type, and the names of users associated with static terminals. You can edit this file directly to add or delete terminals from the system. There is only one terminal configuration file per application system, and its name is fixed.

The terminal configuration file on the system device (*:sd:*) contains information about local terminals and, if it is an iRMX-NET server, can also contain information about terminals for remote diskless systems. During initialization, the HI determines whether the system device is local or remote. If it is local, the HI initializes local terminals defined in the first part of the *:config:terminals* file. If the system device is remote, the HI initializes terminals defined in a subsequent section of the file.

Note

You can force the HI to use a different file than *terminals* for terminal initialization. Specify the terminal configuration file in the TCF parameter of the *rmx.ini* file or in the rq_hterm parameter of a BPS file on a Multibus II system.

See also: Chapter 5 for *rmx.ini* parameters
BPS, *MSA for the iRMX Operating System*

When Changes to the Terminal Configuration File Take Effect

Each static terminal is configured for a specific user. When the HI starts running, it stores the information about the terminal in memory. If you change information about a static terminal's user or attributes for the user, the change does not take effect until you reboot the system.

If you change the terminal type of a dynamic terminal in the terminal configuration file, the change takes effect the next time a user logs on to the terminal. However, changes made to the number of terminals or to the device names do not take effect until you reboot the system.

Configuration of Local Terminals

The first line in the *:config:terminals* file is an integer specifying the number of local terminals the HI initializes. Each succeeding line specifies attributes of a single terminal device as follows:

```
device_name,[user_name],reserved,[terminal_type]
```

Where:

`device_name`
: The physical device name (DUIB name) of the terminal as specified in the ICU or in a loadable terminal device driver.

  See also: DUIB names for terminal devices, Chapter 4

`user_name`
: If there is only a comma here (as a placeholder), this is a dynamic terminal. Any user verified in the UDF can log on. Placing a user logon name here, three to eight characters long, specifies a static terminal. On initialization, the HI automatically logs this user on to this terminal. The static user must have an entry in the UDF and have a user attributes file.

`reserved`
: Reserved; use a comma as a placeholder.

`terminal_type`
: Specifies a terminal name from the terminal definition file. The default value is `ANY`, which applies to all terminal types. Entering a name for a specific terminal type enables the full screen-editing features of the CLI.

During HI initialization, memory for all static terminals is allocated in the order they appear in the *:config:terminals* file. To ensure that high-priority terminals have access to the system, list static terminals in order of their importance.

In DOSRMX, DOS applications can access any serial device not attached by the iRMX OS. When the iRMX OS attaches a serial device, it traps the device's interrupt; DOS applications cannot use that interrupt or the associated I/O port(s). If you intend to use a serial device from DOS, do not enter the device name in the *:config:terminals* file or attach it with the **attachdevice** command.

CAUTION

The *:config:terminals* file is only for initializing user logon devices. If a mouse is attached to a terminal that is specified in the *:config:terminals* file (such as COM1 or COM2), the system will hang. This occurs either during HI initialization or when the mouse is accessed from DOS.

Configuration of Remote Terminals

Use the *:config:terminals* file to initialize terminals on remote diskless systems that use the file server as their system device. In this case, the file contains several additional lines:

```
number_of_terminals_for_local_host
local_terminals
//
number_of_terminals_remote_host#1, remote_host#1_name
remote_host_terminals
//
number_of_terminals_remote_host#2, remote_host#2_name
remote_host_terminals
//
.
.
.
//
```

Where:

`number_of_terminals_for_local_host`

`local_terminals`
: The number of terminals and associated terminal configuration entries for the file server. `Local_terminals` lines are the same as described previously.

`//`
: A delimiter between groups of terminal configurations.

`number_of_terminals_remote_host#x`
: The number of terminals to attach for a remote host.

`remote_host#x_name`
: The network node name of the remote host that gets system device services from this system. This is the name cataloged with the iRMX-NET Name Server.

  See also: Name server, *Network User's Guide and Reference*

`remote_host_terminals`
: The terminal configuration lines for the remote system.

Terminal Configuration File Examples

This example *:config:terminals* is for an DOSRMX system with three terminals: two are dynamic and one is static.

```
3
d_cons,,,PC
com1,,,any
com2,ted,,VT100
t550_0,,,any
```

The `d_cons` device is the DOS console device provided by the loadable *keybd.job*. The second and third devices are the COM1 and COM2 serial ports on a PC, resident in the OS. COM2 is set up as a static terminal for user `ted`. There is a fourth device: the *h550drv* loadable device driver; if you change the first line of this *:config:terminals* file to 4 instead of 3, this device would be initialized on reboot.

System console output during initialization is always sent to the first terminal in *:config:terminals*. In this example, the output from *:config:r?init* (and *:config:loadinfo*) does not appear because `d_cons` is loadable and not yet available.

See also: HI Initialization and logon, Chapter 1,
Loading and unlocking terminal devices, Chapter 3,
keybd.job, Chapter 4

The next example is for a Multibus II system running the iRMX III OS. The board that attaches the hard disk is the local host. The remote hosts are other boards in the system (named `slot3`, `slot4`, `slot5`, and `slot6`) that use terminals associated with their names:

```
1
t0,,,wys50
//
1,slot3
t279_1,,,rgi
//
1,slot4
t279_2,,,rgi
//
1,slot5
t279_3,,,rgi
//
1,slot6
t279_4,,,rgi
//
```

Terminal Definition File

The terminal definition file, *:config:termcap*, defines the terminal types and their characteristics. There is only one terminal definition file in the system. Terminal types used in *:config:terminals* must be defined in the *:config:termcap*. You can edit this file directly to add or delete terminal types from the system. The configuration commands in the *:config:termcap* are used by the CLI and the AEDIT editor, and include parameters and control sequences specifying how the CLI function keys operate. Following are some common terminal types; refer to the *:config:termcap* file on your system for other terminal types that may also be defined.

Terminal Type	Description
ANY	Default ANSI terminal for terminals not listed
1510E	Hazeltine 1510 with escape lead-in
1510T	Hazeltine 1510 with tilde lead-in
ADM3A	Lear Seigler ADM-3A
AT386	PC system console
RMXPC or PC	PC system console
QVT102	Qume QVT102, in QVT102 mode
RGI	SBX 279 Graphic Subsystem
S120	PC system console
TV910P	Televideo 910 Plus
TV950	Televideo 950
VT100	DEC VT100, VT101 (also for Wyse 75 and Wyse 85)
VT102	DEC VT102
VT52	DEC VT52
WYSE50 or WY50	Wyse 30, Wyse 50
XTERM	X-Windows terminal
ZENTEC	Zentec Zephyr and Cobra

The format of a terminal definition in *:config:termcap* is shown below. Use either a semicolon (;) or a space as a separator.

```
name = terminal_name;
code = value; code = value; [...]
//
```

Where:

`name = terminal_name`
: Up to seven characters specifying a name for this terminal type.

`code`
: An input or output code, as shown in Table 2-1.

`value`
: A hexadecimal value for the ASCII character(s) that correspond to this code. Do not follow the value with an H.

`//`
: A delimiter for each terminal definition in the file.

Terminal Control Codes

Table 2-1 lists the control codes used in *:config:termcap*. Some are used to define CLI function keys, others are used by the AEDIT editor.

See also: CLI function keys, *Command Reference*;
Function keys, *AEDIT User's Guide*

Table 2-1. Terminal Control Codes for CLI Function Keys

INPUT CODES	
Codes	**Meaning**
AB= *hhhh*	Sets <Esc>
AFCL= *hhhh*	Sets <Left-Arrow>
AFCR= *hhhh*	Sets <Right-Arrow>
AFCU= *hhhh*	Sets <Up-Arrow>
AFCD= *hhhh*	Sets <Down-Arrow>
AFCH= *hhhh*	Sets <Home>
AR= *hhhh*	Sets <Rubout> (<Backspace> or <Del>)
AFXF= *hhhh*	Sets delete character <DelCh>
AFXA= *hhhh*	Sets delete right <DelR>
AFXX= *hhhh*	Sets delete left <DelL>
OUTPUT CODES	
Codes	**Meaning**
AFMB= *hhhh*	Moves cursor to start of line
AFML= *hhhh*	Moves cursor left
AFMR= *hhhh*	Moves cursor right
AFEK= *hhhh*	Erases entire line
AFEL= *hhhh*	Erases to the end of the line
BELL= *hhhh*	Beeps the terminal bell

hhhh represents a 1 to 4-byte hexadecimal number (2 to 8 characters)

You can also specify a null value for a control code. Then, the CLI tries to bypass the missing output character by simulating its function. For example, if a terminal has no rubout character, specify `AR=;`.

If a function is not available on a terminal, set the corresponding code to `FF`.

If you are not sure which terminals the system includes, or for compatibility with previous iRMX releases, use the `ANY` terminal type defined in the *termcap* file.

Terminal Definition Example

This is an example terminal definition used to access the iRMX OS from a UNIX X-Window with remote (virtual terminal) software. The first half of the definition is used by the CLI and the second half is used by the AEDIT editor.

```
NAME = XTERM;
AFCL = 1B5B44; AFCR = 1B5B43; AFCU = 1B5B41; AFCD = 1B5B42;
AFML = 1B5B44; AFMR = 1B5B43; AFMB = 0D;
AFXA = 01; AFXF = 06; AFXX = 18;
AFEK = 1B5B324B; AFEL = 1B5B4B;
BELL = 07;

AFMU = 1B5B41; AFMD = 1B5B42; AFMH = 1B5B48;
AFES = 1B5B324A; AFER = 1B5B4A;
AFDL = ; AFIL = ;
AFRV = 1B5B376D; AFNV = 1B5B306D;
AI = T; AC = T;
AH = VT100;
AB = 1B; AR = 08; AFCH = 1B5B48;
AV = 24;
//
```

How the CLI Uses Terminal Support Code

The Terminal Support Code (TSC) is a support library for terminal device drivers that use the iRMX BIOS. The CLI uses some TSC features and you can use the TSC in the design of your own command interface. This section describes TSC features used by the CLI: type-ahead buffers, default control characters, and escape sequences.

See also: Terminal support code, *Driver Programming Concepts* for a complete specification of TSC features

When you enter characters at a terminal, the TSC sends the first line to the OS for processing and stores additional lines in a type-ahead buffer. It sends the next line in the buffer to the OS after the OS finishes with the first line. If the type-ahead buffer becomes full, the TSC sounds the terminal bell and refuses to accept input.

The TSC also provides control character support that the CLI uses by default. Table 2-2 summarizes these characters.

Table 2-2. Default TSC Control Characters

DEFAULT INPUT CONTROL CHARACTERS	
Character	**Results**
<carriage return> or <line feed>	Terminates current line with a <CR>-<LF> combination and puts cursor at start of next line
<Rubout>	Deletes a single character; depending on TSC configuration, it removes the character or echoes it to the display surrounded by # characters
<Ctrl-P>	Removes any special meaning from immediately following input control characters; has no effect on output control characters such as <Ctrl-C>
<Ctrl-R>	Reprints a line with editing performed (not used by CLI, except with terminal type “ANY”)
<Ctrl-U>	Discards contents of the type-ahead buffer
<Ctrl-X>	Discards the current input line
<Ctrl-Z>	Specifies an end-of-file
DEFAULT OUTPUT CONTROL CHARACTERS	
Character	**Results**
<Ctrl-O>	Toggles the terminal in and out of discarding mode
<Ctrl-Q>	Resumes previous output mode, reversing the effects of a <Ctrl-S>, <Ctrl-W>, <Ctrl-T>, or <Ctrl-O>
<Ctrl-S>	Stops output display; has no effect after a <Ctrl-O>
<Ctrl-T>	Scrolls output one line at a time
<Ctrl-W>	Scrolls output one screen at a time
<Ctrl-C>	Aborts currently executing program (but not background jobs)

The TSC also accepts escape characters that enable you to further define a terminal. For example, you could set the scroll count or switch your terminal into transparent mode so that control characters have no effect. You can enter these escape characters from the terminal, or you can write them to the terminal from a program. The CLI does not use TSC escape sequences other than those for codes defined in Table 2-1.

Configuring Terminals for a Modem

This section explains how to set up a modem for use on the iRMX end of a modem link.

See also: Documentation for your modem or terminal for how to connect or use a modem on the terminal end

Before attempting to use a modem with an iRMX system, ensure that the port to which you connect the modem is configured as a modem link:

- In ICU-configurable systems, use the ICU to set the Modem Control (MC) parameter of the terminal driver to `yes`.
- In DOSRMX and iRMX for PCs systems, you must change the loadable terminal driver so that the Unit Information (UINFO) table is set for modem control; set bit 3 of `terminal_flags` to 1. There are two ways to do this:

  _ Use the BIOS **a_special** system call to set the terminal attributes (Function 5). Set bit 3 of the `terminal_flags` field to 1 (this method is not for dynamic terminals).

  See also:**a_special**, *System Call Reference*

  _ Change the source code for the loadable driver's configuration file and rebuild the driver. For example, the source code for the HOSTESS 550 terminal driver includes a configuration file named *h550cfg.a38*. This file defines a single UINFO table named `UINFO_550` that is used by all the DUIBs for the driver (T550_0 through T550_7). `UINFO_550` sets `term_flags` to 0101H. Create a different UINFO table with `term_flags` set to 0109H. Specify your new UINFO table name in the DUIB(s) to be used with the modem.

  You could also create new DUIB names (for example T550_0M rather than T550_0) that reference the new UINFO table, and specify them in the *:config:terminals* file and load your new version of the *h550drv* driver. You could use this same method to rebuild the *comdrv* driver to set up COM1 or COM2 for a modem.

See also: Loading and unlocking terminal devices, Chapter 3,
comdrv, Chapter 4,
UINFO table structure for a terminal driver, making a device driver loadable, *Driver Programming Concepts*

Setting Up the Modem

Set up a Hayes-compatible modem as follows:

Switch	Setting
DTR (Data Terminal Ready)	Normal
DSR (Data Set Ready)	Normal
Carrier Detect	Normal
Auto Answer	Enabled
Command Response	Suppress
Command Mode	Disabled

Once these have been set, and both sides of the modem link are properly cabled, the modem link is ready. Use the modem on the terminal end of the link to dial up the iRMX system. When the modem answers, the dynamic logon procedure asserts the DTR signal. The word `CONNECT` appears on your terminal. If the serial port on the iRMX end of the link has an auto-baud search, enter four capital `U`s (at a rate of about two per second) and a <CR>. If your terminal is not configured for auto-baud search enter <CR>. The terminal should now respond as if it were connected to the iRMX system by a dedicated line. The response from a modem link operates at the baud rate of the modem and therefore may be noticeably slower than a true dedicated line.

Configuring a Multiuser Environment

On a multiuser system, you may want to control access to a system's files and devices. This discussion only applies if you install the OS on an iRMX hard disk partition. In this case, the named file driver provides file access control according to user IDs. If you install DOSRMX or iRMX for PCs on a DOS file system, this discussion does not apply. You can protect your system in these ways:

- Limit access to the configuration files.
- Allow only the system administrator to detach system devices by ensuring that only the Super user has all access rights to the *:sd:* root directory.
- Allow users to be owners of their default directories and restrict access to other files and directories.
- Allow the World user read access to files and list access to directories containing commands and utilities, but disallow users to delete or modify files in those directories.
- Allow the World user full access to the *:work:* directory; this is required to use the compilers and development tools.

Changing Access Rights

Use the **permit** command to change access rights for files and directories. This command adds or removes users from a file's access list and set specific access rights for these users. You can give only three users access rights to any file or directory. Display the access rights for files and directories with the **dir** command.

See also: **dir, permit**, *Command Reference*

File Structure of a Multiuser Environment

The *:sd:* directory is the root directory for the system device. The Super user owns this directory. Other users can have list access to allow them to view the files in the root directory. However, they should not have change-entry or delete access to the root directory. Add-entry access is optional.

Four first-level directories store commands and utilities. They are *:lang:*, *:utils:*, *:util286:*, and *:system:*. The Super user owns these directories and the files they contain. To protect the commands and utilities from damage or deletion, other users should have only list access to the directories (to be able to see what is available) and read access to the data files (to be able to run the commands).

Configuring Loadable Jobs and Drivers 3

You configure loadable jobs and drivers when they are loaded by the **sysload** command. The loaded job or driver becomes part of the OS, as a child job of the HI. When loaded, the job or driver becomes resident in memory. It remains part of the OS until it explicitly exits or until the system is rebooted.

Run **sysload** from the command line or from a submit file. You can use **sysload** while testing drivers and jobs, before configuring them as part of a bootable system with the ICU.

In iRMX for PCs and DOSRMX, you use the *:config:loadinfo* file, containing a series of **sysload** commands, to load drivers and jobs automatically during system initialization. The default *loadinfo* file has command lines for each loadable job or driver, with some commented out.

Note

Typically the *r?init* file contains a line to submit the *loadinfo* file. If you need more than one *loadinfo* file (for example, on a system that supports other remotely-booted systems), you may want to use different versions of the *r?init* file to submit different versions of the *loadinfo* file. You can force the HI to use a different file than *r?init* for its initialization. Specify the initialization file in the SCF parameter of the *rmx.ini* file or in the `rq_hscf` parameter of a BPS file on a Multibus II system.

See also: Chapter 5 for *rmx.ini* parameters
BPS parameters, *MSA for the iRMX Operating System*

Loading some jobs may also require that you change the *rmx.ini* load-time configuration file or DOS configuration files.

See also: *rmx.ini* file, Chapter 5,
ICU-configurable jobs, Chapter 6,
Writing loadable drivers, *Driver Programming Concepts*

Using the Sysload Command

Invoking *`sysload -l`* lists the currently loaded jobs and drivers. When loading jobs and drivers, the command syntax is:

```
sysload [switch] [(min,max)] pathname [driver_params]
```

Where:

`switch` Optional command line switches:

Value	Meaning
-i *name*	Specifies a file, logical name, or logical device as the *:ci:* (standard input) for the loaded job.
-l	List the names of all jobs and drivers currently loaded through this facility.
-o *name*	Specifies a file, logical name, or logical device as the *:co:* (standard output) for the loaded job.
-r	Load this job over an existing instance. The existing job is deleted only after successful loading.
-u *name*	Unload a job by deleting it; specify either the job name or token. **Note:** Not all jobs can be successfully unloaded
-w	Wait for job initialization to complete using a configurable timeout value. See also: Job synchronization timeout (JST), Chapter 5

`(min,max)` Minimum and maximum sizes in Kbytes of the memory pool required. If not specified, default values built into the job or driver are used. If you specify `min`, you must also specify `max`. Enclose the values with parentheses and separate them with a comma.

`pathname` The full pathname of the job or driver, for example, */rmx386/drivers/ramdrv*. This pathname must begin at the root directory and can include logical names.

`driver_params`
Parameters specific to the job or driver being loaded (if any). The syntax of parameters is defined by each job and driver; **sysload** passes them directly. Some drivers provided with the OS require parameters to be enclosed in parentheses; parentheses are optional otherwise.

See also: **sysload**, *Command Reference*

How Jobs and Drivers Handle Sysload Switches

To handle the command-line switches -w, -u, and -r, the job or driver itself may need to take some action. Loadable jobs or drivers that you create should follow the guidelines discussed here.

The -w (Wait) Switch

When you specify the -w switch, the **sysload** command attempts to synchronize with the new job's initialization by blocking until the initialization is complete. First **sysload** creates the new job. Then, if the -w switch is specified, **sysload** looks up the object R?END_INIT in the new job's object directory and waits for the object to be cataloged. The wait period is controlled by the JST timeout parameter. Set JST either in the *:config:rmx.ini* file or on the HI screen of the ICU. The default wait period is 60 seconds.

It is expected that the job will catalog an R?END_INIT object (a null token is OK) when its initialization is complete. If the job fails to catalog R?END_INIT, the lookup will timeout after the configured time or the default time. Thus, if you do not set JST shorter, any **sysload** -w invocation of a job that does not catalog an R?END_INIT object will always take 60 seconds before the command terminates.

The -u (Unload) Switch

When you specify the -u switch, the **sysload** command attempts to delete a loaded job, using one of these mechanisms:

- First **sysload** looks up a data mailbox object with the name R?EXIT_MBX in the specified job. If this object exists, the **sysload** command does not delete the job directly. Instead, **sysload** sends to the mailbox a 2-byte data message with the exception code 0FFDDH, meaning E_DELETE_YOURSELF. Then **sysload** waits until the job deletes itself before returning. The job is expected to have a task waiting at this mailbox that will receive the message, perform any cleanup, and delete the job. The job must create and catalog this mailbox during its initialization to allow itself to control its own deletion.
- If the R?EXIT_MBOX object does not exist, or if any error occurs during the lookup, **sysload** attempts to delete the job normally by calling **rq_delete_job**.

This mechanism lets a job control its own deletion in an orderly fashion and free any resources under its control.

> **Note**
> Many of the loadable jobs and drivers provided with the OS cannot be successfully removed with the -u switch. See the descriptions in Chapter 4 for jobs that support this switch.

The -r (Reload) Switch

When you specify the -r switch, the **sysload** command first loads and starts a new instance of the specified job or driver, then deletes the previous instance. The command uses the same mechanism described for the -u switch to delete the previous instance of the job or driver.

> **Note**
> Only jobs and drivers that support the -u switch can successfully be reloaded with the -r switch.

Log Files

Loadable jobs and drivers create a log file in the directory for that job or driver. The log file has the name of the job or driver with a `.log` extension. The sign-on message from the loaded driver indicates that the operation is successful. Otherwise, an appropriate error message appears in the log file.

Using a Loaded Device Driver

After loading a device driver, invoke the **attachdevice** command using one of the DUIB names installed by the driver. Device drivers take effect immediately after loading. If you re-load the driver (with the -r switch) the new DUIB is used the next time you invoke **attachdevice**.

See also: **attachdevice**, *Command Reference*

Testing the Configuration

Test or debug your system by adding loadable drivers and jobs while the system is running. This lets you experiment with the configuration while observing system performance. To do this, invoke **sysload** from the command line, rather than from the *loadinfo* file. The changes disappear when you reset or reboot the system.

For example, to load the ram disk driver with a 1024 Kbyte disk, enter this at the iRMX prompt:

```
sysload /rmx386/drivers/ramdrv (1024)
```

A log file named *ramdrv.log* is created. Access the ram disk like any other device.

See also: *ramdrv*, Chapter 4

Example Loadinfo File

Figure 3-1 is an example *:config:loadinfo* file. It contains **sysload** commands for each loadable job and device driver available to the system. Not all of these are loaded by default. Some of the lines are commented out (disabled) by a preceding semicolon (;). Some of the jobs are mutually exclusive, others are not selected during installation. You may also add your own **sysload** commands for terminals, application jobs, or custom drivers. Changes made to the *:config:loadinfo* file take effect on reboot.

See also: Your *:config:loadinfo* file,
Descriptions of loadable jobs and drivers, Chapter 4

```
; iRMX System Jobs
•

; Shared C Library Job
;sysload /rmx386/jobs/clib.job


; System Debugger (SDB) Job

;sysload /rmx386/jobs/sdb.job


; Soft-Scope III Kernel Job

;sysload /rmx386/jobs/ssk.job(local)
```

Figure 3-1. Loadinfo File Examplee

```
; iNA 960 Network Jobs for PC Systems
;
; COMMputer Job (EtherExpress Pro) - ES-IS Network Layer
;sysload -w /rmx386/jobs/iethproe.job
; COMMputer Job (EtherExpress Pro) - NULL2 Network Layer
;sysload -w /rmx386/jobs/iethpron.job
; COMMputer Job (DEC 21X4X NIC) - ES-IS Network Layer
;sysload -w /rmx386/jobs/idec43e.job
; COMMputer Job (DEC 21X4X NIC) - NULL2 Network Layer
;sysload -w /rmx386/jobs/idec43n.job
; COMMengine MIP Job (PCLINK2,PCLINK2A)
;sysload -w /rmx386/jobs/ipcl2.job (360,21,CC000)
;

; iNA 960 Network Jobs for Multibus I Systems

; iSBC386SX COMMputer Job (with SBX586) - ES-IS Network Layer
;sysload -w /rmx386/jobs/i386sxe.job
; iSBC386SX COMMputer Job (with SBX586) - NULL2 Network Layer
;sysload -w /rmx386/jobs/i386sxn.job
; iSBCPCP4 COMMputer Job - NULL2 Network Layer
;sysload -w /rmx386/jobs/iethpron.job
; iSBCPCP4 COMMputer Job - ES-IS Network Layer
;sysload -w /rmx386/jobs/iethpron.job
; Multibus I COMMputer Job (Null Subnet) - NULL2 Network Layer

•
```

Figure 3-1. Loadinfo File Example (continued)

```
; iNA 960 Network Jobs for Multibus II Systems
;
; iSBCP5090 COMMputer Job - ES-IS Network Layer
;sysload -w /rmx386/jobs/ieproe2e.job
; iSBCP5090 COMMputer Job - NULL2 Network Layer
;sysload -w /rmx386/jobs/ieproe2n.job
; iSBC486SX25,iSBC486DX33/66 COMMputer Job (EWENET) - ES-IS Network
Layer
;sysload -w /rmx386/jobs/iewexpe.job
; iSBC486SX25,iSBC486DX33/66 COMMputer Job (EWENET) - NULL2 Network
Layer
;sysload -w /rmx386/jobs/iewexpn.job
; Multibus II COMMengine MIP Job (186/530 default)
;sysload -w /rmx386/jobs/icemb2.job
; Multibus II COMMputer Job (Null Subnet) - NULL2 Network Layer
;sysload -w /rmx386/jobs/inlmb2n.job
```

Figure 3-1. Loadinfo File Example (continued)

```
;
•
; For All Systems
•
; iRMX-NET Server Job
;sysload /rmx386/jobs/rnetserv.job


; iRMX-Net Remote File Consumer Job

;sysload /rmx386/jobs/remotefd.job


; Paging Subsystem
;sysload /rmx386/jobs/paging.job
•
; Flat Model Support - requires the paging subsystem
;sysload /rmx386/jobs/flat.job
•
; RAM Disk Driver
;sysload /rmx386/drivers/ramdrv(1024)
```

; Figure 3-1. Loadinfo File Example (continued)

```
;Loading the new RadiSys TCP/IP stack

;      Load the desired NIC drivers
;Loopback driver for the new TCP/IP stack
;sysload /rmx386/jobs/loopback.job ntrans=256 ncbs=256
;
;To use the Intel EtherExpressPro100plus PCI NIC
;sysload /rmx386/jobs/eepro100.job ntrans=256 ncbs=256
;
;Or to use an NE2000 compatible ISA NIC
;sysload /rmx386/jobs/ne.job irq=9 base=0x320 ntrans=256 ncbs=256
;
;Or to use a DEC 21X4X-based PCI NIC
;sysload /rmx386/jobs/tulip.job ntrans=256 ncbs=256
;
;Or to use load the new TCP/IP stack on top of one of the iNA jobs
;sysload /rmx386/jobs/edl.job ifport=0x20 ntrans=256 ncbs=256
;
;Sleep 2 Seconds to let drivers finish NIC driver initialization
;;sleep 2

;Load the TCPIP Stack – Assumes that the TCP/IP stack configuration is in the
;::CONFIG:RMX.INI file
;sysload /rmx386/jobs/ip.job
sysload /rmx386/jobs/rip.job
sysload /rmx386/jobs/udp.job
sysload /rmx386/jobs/tcp.job

•
```

; ; Figure 3-1. Loadinfo File Example (continued)

```
; iRMX Device Drivers – PC Architecture systems;


; Keyboard/Console Job
sysload -w /rmx386/jobs/keybd.job (f,1,ffff)
•

; COMn: Drivers
;sysload /rmx386/drivers/comdrv (1,3f8,48)
;sysload /rmx386/drivers/comdrv (2,2f8,38)
•

; Line Printer Driver
;sysload /rmx386/drivers/lpdrv


; Hostess 550 Serial Board Driver

;sysload /rmx386/drivers/h550drv(10,280,58)

;sysload /rmx386/drivers/h550drv(8,280,58)

;sysload /rmx386/drivers/h550drv(4,280,58)


; DigiBoard DigiCHANNEL PC/X Serial Board Driver
;sysload /rmx386/drivers/pcxdrv(0,10,0,58)
;sysload /rmx386/drivers/pcxdrv(0,8,0,58)
;sysload /rmx386/drivers/pcxdrv(0,4,0,58)

•

; PCI SCSI Server
;sysload /rmx386/jobs/pcisrv.job (PCIAD1)
•

; PCI SCSI Driver

;sysload /rmx386/drivers/pcidrv ('AT',1,0,a)
```

Figure 3-1. Loadinfo File Example (continued)

```
; iRMX Device Drivers – Multibus I systems;

; Multibus I TCC driver
•
;sysload /rmx386/drivers/tccdrv(da0000,88a7,27,f)
•

; iRMX Device Drivers – Multibus II systems;

; Multibus II PCI Driver
;sysload /rmx386/drivers/pcidrv('386/258',1,0,a)
;sysload /rmx386/drivers/pcidrv('386/258D',1,0,a)
;sysload /rmx386/drivers/pcidrv('486/133SE',1,0,a)

; Multibus II ATCS Driver
;sysload /rmx386/drivers/atcsdrv('186/410',1,6,a)
;sysload /rmx386/drivers/atcsdrv('186/450',1,12,b)
;sysload /rmx386/drivers/atcsdrv('MIX386/020',1,36,c)
;sysload /rmx386/drivers/atcsdrv('MIX386020A',1,36,c)
;sysload /rmx386/drivers/atcsdrv('MIX486020A',1,36,c)
;sysload /rmx386/drivers/atcsdrv('386/120',1,12,d)
;sysload /rmx386/drivers/atcsdrv('386/133',1,12,d)
;sysload /rmx386/drivers/atcsdrv('486/125',1,12,d)
;sysload /rmx386/drivers/atcsdrv('386/258',1,5,r)
;sysload /rmx386/drivers/atcsdrv('386/258D',1,5,r)
;sysload /rmx386/drivers/atcsdrv('486/133SE',1,5,r)
```

Figure 3-1. Loadinfo File Example (continued)

Loading and Unlocking Terminal Devices

The HI attempts to initialize all terminals designated by the *:config:terminals* file after completing the execution of all commands present in the *:config:r?init* file. If any of these terminals are supported by device drivers that are not configured into the iRMX boot image or are not sysloaded by commands in the *:config:loadinfo* file, the HI aborts their initialization and marks them as locked.

If you attempt to sysload from the command line a terminal driver that supports terminals listed in the *:config:terminals* file, these terminals are not automatically initialized by the HI. You must also enter an **unlock** command before initialization will take place.

Choosing a Network Job

The native iRMX networking jobs include iNA 960 network jobs with a programmatic interface only, *i*.job*. Each job matches a specific computer's hardware and software configuration. On top of an iNA 960 job you can run iRMX-NET client and/or server jobs to provide transparent file access and support iRMX-NET command-line utilities. You can also run the new TCP/IP stack on top of any of the iNA jobs after loading the interface driver EDL.JOB.

Only one of the iNA 960 jobs can run at a time. Any attempt to load a second job will fail, and may disrupt the operation of the first job.

See also: *Network User's Guide and Reference* for information about the features of iRMX-NET and the programmatic network interface

iNA 960 Jobs

The iNA 960 jobs provide programmatic access to transport services from your application, without transparent file access. These are called collectively *i*.job*. The selection includes:

- COMMputer jobs, which use LAN hardware integrated into the host CPU
- COMMengine jobs, which use a NIC separate from the host CPU (also called MIP jobs)
- Jobs that run without network hardware, to support local ISO transport applications

See also: *i*.job*, Chapter 4

iRMX-NET Jobs

iRMX-NET provides a complete network package, including a command-line interface and transparent file access to remote iRMX-NET systems. Load the iRMX-NET job(s) after loading the underlying *i*.job* that is appropriate for your system. The jobs include:

- iRMX-NET client, which includes the file consumer and remote file driver in a single job, *remotefd.job*
- iRMX-NET server, *rnetserv.job*

You can run one or both of these jobs depending on whether you want the system to be only a client, only a server, or both.

New vs. Old Network Job Names

A wider selection of network jobs has been provided in this release, and the existing file names have changed. If you are familiar with the old job names, Table 3-1 shows the equivalent new names to substitute. The former names include iRMX-NET and iNA 960 in the same job. In the list of current names, if you want iRMX-NET you must add the iRMX-NET client and/or server separately from the *i*.job* names.

Table 3-1. New Network Job Names

Old Name	Release 2.1 Name	Current Name
mipat.job	ipcl2.job	ipcl2.job
mipmb2.job	icemb2.job	icemb2.job
netat.job	rpcl2n.job	ipcl2.job plus rnetserv.job/remotefd.job
netlp486.job	r486hidn.job	i486hidn.job plus rnetserv.job/remotefd.job
netmb1.job	r386sxn.job	i386sxn.job plus rnetserv.job/remotefd.job
netmb2.job	rcemb2n.job	icemb2n.job plus rnetserv.job/remotefd.job
netatn.job	rnlatn.job	inlatn.job plus rnetserv.job/remotefd.job
netmb1n.job	rnlmb1n.job	inlmb1.job plus rnetserv.job/remotefd.job
netmb2n.job	rnlmb2n.job	inlmb2.job plus rnetserv.job/remotefd.job
ntp4at.job	inlatn.job	inlatn.job
ntp4mb1.job	inlmb1n.job	inlmb1.job
ntp4mb2.job	inlmb2n.job	inlmb2.job

The ISO Transport Software

With any of the iNA 960 jobs, your application can make calls to the various networking layers. The software conforms to the International Standards Organization (ISO) Transport Class 4 and the Open Systems Interconnection (OSI) model. Regardless of the job you choose, the interface to your application is the same. The only exception is *inl*n.job*, which does not include a Data Link layer or Network Management Facility, because it runs without network hardware.

Loadable or First-Level Network Jobs

You can either sysload the loadable network jobs or use the ICU to configure them as first-level jobs that boot with the OS. If you need to change the configuration of a loadable job, you can enter configuration parameters for one of the first-level jobs in the ICU and then produce a loadable job from it. Enter the name of the network job, without the *.job* extension, and choose the loadable option.

See also: Network screens, *ICU User's Guide and Quick Reference*

Network Jobs in the Loadinfo File

Once you know which network job you need, check the initial *loadinfo* file to see which job it is set up to load. The initial configuration of the file depends on your answers to the prompts during installation. If you specify that you use a network, an

i.job* and iRMX-NET jobs are enabled for loading in the *loadinfo* file. If you specify that you do not use a network, *inl*.job* and iRMX-NET are enabled for loading.

If necessary, edit the file to load a different job. Be sure to comment out the old jobs when you enable the new ones.

If your computer is not connected to a network and your applications do not communicate by making **cq_** calls to the ISO Transport Software, comment out the network jobs. There is significant overhead to running network software.

Timing Sequence of Loading the Network

When you load an *i*.job* or iRMX-NET jobs, always use the **sysload -w** command, which starts the job and waits until the NIC is loaded before it returns. Otherwise, there are timing problems, if you start the iRMX OS from the *autoexec.bat* file and load from the *loadinfo* file.

Some network jobs load software onto a NIC. This can take some time: up to 30 seconds in a PC loading i*pcl2.job*. The iRMX OS could be running and the *autoexec.bat* file finished executing before the NIC is initialized. Attempts to do any network-related operations before the NIC is initialized will fail.

Modifying the Client Definition File (CDF)

If you load iRMX-NET, there is one other network configuration file: the client definition file, or CDF. Entries in the CDF specify a client (consumer) machine name and password. Under iRMX-NET, if a user attaches to a remote node whose CDF contains the name and password of the user's local node, the user has access to any public directories on the remote system. (All public directories are accessible if installed on the DOS file system, but are subject to iRMX file access permissions if installed on an iRMX file system.) Modify the CDF by invoking the **modcdf** command as the Super user.

See also: Client-based and server-based protection, *Network User's Guide and Reference*

The node name and password are configuration values and, by default, are the same for all current releases of the iRMX OS. This name and password are specified in the default CDF. If you don't change the configuration of the CDF or of a client node's name and password, all iRMX systems connected by a network have access to the other systems' files.

In an ICU-configurable system, use the ICU to change a node's name and password. In DOSRMX and iRMX for PCs, change these values in the load-time configuration.

See also: [UA] block of the *rmx.ini* file, Chapter 5

Using iRMX-NET in a DOS Environment

There are two ways you can use DOS network software together with iRMX-NET software on the DOSRMX OS:

- Use any NIC supported by iNA 960 to allow simultaneous DOS and iRMX networking on an Ethernet network. The iRMX OS manages the NIC but DOS can also communicate on the network.
- Use a null network iNA 960 job to allow network software on the DOS and iRMX OSs to communicate with each other on the same machine. There is no actual network interface, but DOS has access to iRMX-controlled volumes as if they were on a network.

Both methods require that you use MS-NET software. If you use the PCL2(A) NIC, MS-NET is included with the product. MS-NET is also shipped with the iRMX OS. In any case, you must replace the standard MS-NET NetBIOS driver with a driver called PCNET, shipped with the iRMX OS. The executable filename of this driver is *pcnet.exe*; it runs under DOS.

You must also run both a standard DOS network redirector and an iRMX network redirector called *netrdr.job*.

The following sections describe how to load this software to make the two OS's network software work together.

Using PCNET with MS-NET

MS-NET software includes a configuration file named *msnet.ini* that you typically use to automatically start the MS-NET software. When you install DOSRMX, the installation software will append new commands to the *msnet.ini* file if it already exists on your machine. The OS installation also includes the appropriate MS-NET software, with *msnet.ini* configured correctly to use PCNET. This software is in the */dosrmx* directory.

This section describes the differences between the standard DOS-only version of *msnet.*ini and the commands needed to use PCNET with DOSRMX, so that you can set up the *msnet.ini* file yourself if necessary.

The DOS-only version of *msnet.ini* includes statements like the following.

```
start rdr $1
    netbios c:\net\config.nia
    minses
    redir /l:10 /s:10 /P1:4096 /P2:4096 /P3:4096 /Z:1024 /B:10
    setname $1
start srv $1
    netbios
    minses
    setname $1
    share
    psprint
    server /i: /s: /n:-1 /mb:-1 /f:-1 /o:-1 /c:-1 /l:-1
```

These DOS-specific entries allow you to start the MS-NET client software with the command `net start rdr <machine-name>` and to start the server with the command `net start srv <machine-name>`.

To use PCNET with the iRMX OS, add the following commands in the *msnet.ini* file. Notice that these iRMX-specific commands replace the *netbios* driver with *pcnet.exe*.

```
start netrdr $1
    pcnet
    minses
    redir /l:10 /s:10 /P1:4096 /P2:4096 /P3:4096 /Z:1024 /B:10
    setname $1
start netsrv $1
    pcnet
    minses
    setname $1
    psprint
    server /i: /s: /n:-1 /mb:-1 /f:-1 /o:-1 /c:-1 /l:-1
```

These iRMX-specific entries allow you to start the MS-NET client software with the command:

```
net start netrdr <machine-name>
```

Note

The iRMX-specific entries also allow you to start the MS-NET server software with a `net start netsrv <machine-name>` command. However, there is no point in starting the MS-NET server on a system that runs DOSRMX, because the server software runs constantly on the DOS OS, which negates the value of running DOS together with the iRMX OS.

PCNET Configuration

You can add the following switches to the `pcnet` line in *msnet.*ini to change the default configuration of the PCNET driver (the `s` and `c` options may be upper- or lower-case):

`/s:<sessions>`
: Where `<sessions>` is the number of NetBIOS sessions supported. The default value is 6.

`/c:<commands>`
: Where `<commands>` is the number of NetBIOS commands that can be queued to the NetBIOS driver simultaneously The default value is 12.

Network Redirector Software

To use the PCNET driver you must run both an iRMX and a DOS network redirector:

- The iRMX network redirector is *netrdr.job*, which you load under the iRMX OS from the *:config:loadinfo* file, as described in Chapter 4.
- The DOS network redirector is *redir.exe*, which runs under DOS and is invoked by the `redir` line in the *msnet.ini* file.

There are different versions of the DOS *redir* file, for specific versions of DOS. These are typically installed with DOS, but the iRMX installation also provides these files in the */dosrmx* directory. Copy the appropriate file listed below to the filename *redir.exe* and put it in a directory that is on your DOS PATH.

Filename	DOS Version
redir.500	5.0 and 6.x
redir.401	4.01
redir.400	4.0
redir.330	3.3

Note

If you use *redir* on a DOS 6.x system, you must load *setver.exe* in your *C:\autoexec.bat* file before loading *redir.exe*.

See also: Your DOS manual for more information about these files

PCNET and NetBIOS Technical Details

The PCNET driver implements IBM's NetBIOS specification using iNA 960 Transport and Data Link functions. PCNET is a Terminate and Stay Resident (TSR) program that receives NetBIOS interrupt 5CH.

NetBIOS applications produce Network Control Blocks (NCBs) at software interrupt 5Ch with ES:BX pointing to the NCB. PCNET translates NCBs into iNA 960 Request Blocks and issues a software interrupt 5BH with ES:BX pointing to the Request Block.

The iRMX *netrdr.job* takes over interrupt 5BH so it can send and receive Request Blocks between PCNET and the iNA 960 job running under DOSRMX.

The result is that PCNET redirects NetBIOS commands originating from DOS to the transport services provided by the iNA 960 job.

Steps to Load DOS and iRMX Network Software

Follow these steps in order:

1. Set up the *msnet.ini* file and *redir.exe* file as described in previous sections.
2. Enable loading of the appropriate iNA 960 job in the iRMX *:config:loadinfo* file.

See also:*i*.job*, Chapter 4

3. Enable loading of *netrdr*.job and the iRMX-NET Server job, *rnetserv.job*, in the *:config:loadinfo* file. (You can also load the client *remotefd.job*, but it is not necessary.)
4. Enter the iRMX-NET server name in the */net/data* file so it will be established when DOSRMX boots. (This is the name you will use in the `net use` command in Step 11 below.)

See also: */net/data* file, *Network User's Guide and Reference*

5. In one of the iRMX files that runs at boot time, such as *:config:r?init* or *r?init2*, establish public directory names for each iRMX volume you want to access from DOS, using the **offer** command. Don't offer the names until iRMX-NET is running. Example commands are:
```
offer :c_rmx: as rmx_wini
offer :ram: as ramdisk
```
6. Also establish public names for DOS volumes that you want to be available to other systems across the network. For example, if the iRMX system device (*:sd:*) is on a DOS-format drive:
```
offer :sd: as sd
```
7. From the iRMX OS, give World user access to the files and directories on the iRMX volumes so that DOS users can access them. For example:
```
traverse :c_rmx: permit * drau u=world
```

Any file or directory not given World access will not be visible from DOS. Also, files on iRMX volumes that violate the DOS 8.3 filename convention will not be visible from DOS.

8. Remove all iRMX-NET public directories that refer to directories on DOS (EDOS file driver) volumes using the **remove** command. For example:
   `remove :work:`

Note

Do not remove public names for DOS volumes; only the public names for directories on those volumes.

9. Reboot the system to load DOSRMX and its network jobs.

10. Once the iRMX-NET network software has been loaded with the steps above, you can start the MS-NET software that loads PCNET, using this command at the DOS prompt or in a batch file:
    `net start netrdr <machine-name>`
    Where `<machine-name>` is a network name for the DOS network node, and is different from the iRMX-NET server name.

11. From the DOS side of the system you should be able to access files on the iRMX volume through the network. For example, with this command:
    `net use g: \\rmxserver\world`
    all files and directories on drive G: with World access that use the DOS 8.3 naming convention are accessible from DOS. (See also Step 4 above.)

Checking PCNET Programmatically

DOS applications that use the NetBIOS interface provided by the PCNET driver should verify that PCNET has been loaded before trying to use its services. To verify this, issue a software Interrupt 2FH, with the value 0B951H in the AX register and 0H in the BX register. If PCNET is loaded, the BX register will change to a value of 1H. If not, BX remains unchanged.

PCNET Limitations

On Multibus I and II systems, NetBIOS receive datagrams do not work.

On a Multibus II system, if a PC-compatible board running PCNET is individually rebooted with a <Ctrl-Alt-Del>, PCNET will report a system error when it tries to restart. The problem is that the Name Server running on another board (for example, an SBC 486/133SE or MIX 560) has not also been reset. PCNET attempts to establish the PC node name with a **setname** command. The Name Server already has that name cataloged and will not allow it to be cataloged again. If you perform a **deletename** of the PC node name on the iRMX server before rebooting the PC, this problem does not occur. If you do not delete the name and ignore the error, the network redirection may work properly. However, in this case you should reboot the whole system, not just the PC-compatible board.

If you attempt to load the DOS-specific NetBIOS driver instead of PCNET, after starting iRMX network software, you will receive a file server error and the system may encounter a GP fault. This would happen if you entered the command

```
net use rdr
```

instead of the correct command to load PCNET:

```
net use netrdr
```

after modifying the *msnet.ini* file. If you load the DOS-specific NetBIOS driver instead of PCNET and then try to start DOSRMX, the PC may hang. You may need to do a cold reboot (not <Ctrl-Alt-Del>) to properly restart the system.

If you do not remove all public directories on DOS volumes from the iRMX OS, the system will hang or have a GP fault when you attempt to connect to the iRMX-NET server from DOS.

Other iRMX Networking Options

The iRMX OS offers a number of other networking options that work with the basic iRMX network software covered here. These include TCP/IP.

See also: For more on TCP/IP configuration, see *TCP/IP for the iRMX Operating System*

□□□

Reference to Loadable Jobs and Drivers 4

This chapter summarizes the loadable jobs and drivers provided with the OS, and provides an alphabetic reference to each.

Source code for the front-end modules and configuration files for device drivers is located under the *:rmx:demo* directory. Use this source code to modify a driver to suit your needs, or as the basis for writing your own loadable drivers.

See also: *Driver Programming Concepts* for information on modifying supplied driver source modules

Supplied Loadable Jobs and Drivers

The OS provides these kinds of loadable jobs and drivers:

- System jobs, including a variety of network jobs
- File drivers
- Device drivers

The tables that follow are summaries of the jobs and drivers described in this chapter, as well as a summary of interrupt encoding between the iRMX OS and DOS. The rest of the chapter provides reference descriptions for every loadable job and driver mentioned. In the summary tables, these abbreviations are used:

Abbreviation	Meaning
III	iRMX III OS
RPC	iRMX for PCs OS
DRMX	DOSRMX OS
MBI	Multibus I bus
MBII	Multibus II bus
PC	PC bus (ISA or EISA)

System Jobs and File Drivers

Table 4-1 lists the loadable system jobs and file drivers. Some jobs are similar in function to their ICU-configurable versions, except that they operate as child jobs of the HI rather than as first-level jobs.

See also: Configuring System Jobs Using the ICU, Chapter 6

Table 4-1. Loadable System Jobs and File Drivers

Job Name	OS	Bus	Purpose
atcs279.job	all	MBII	ATCS 279/ARC server job for terminal controller boards
bootserv.job	all	MBII	Receives and services requests from clients that boot dependently in a Multibus II system
cdromfd.job	RPC	all	Native CDROM file system driver
clib.job	all	all	C library functions that can be shared by applications rather than being linked to each application
dosfd.job	RPC	all	Native DOS file system driver
edl.job	all	all	NIC interface to iNA conversion job
eepro100.job	all	All	Network interface driver for Intel EtherExpressPro100+ PCI-based NIC
flat.job	all	all	Support code for flat model applications to make system calls and call C Library functions; paging.job is also required
i*.job	all	all	Hardware-specific iNA 960 network jobs; for specific job names see Tables 4-4 and 4-5 on pages 72 and 73
ip.job			Part of TCP/IP
keybd.job	all	all	Supports the DOS console, including PC keyboards
namedfd.job	all	all	Native Named32 and Named48 file iRMX system driver
ne.job			Network interface driver for ISA-based NE2000 compatible NIC cards
ntxproxy.job	all	all	Provides the NTX interfaces needed to make an iRMX III.2.3 system into a Remote INtime client

Table 4-1. Loadable System Jobs and File Drivers (continued)

Job Name	OS	Bus	Purpose
paging.job	all	all	Manages memory with the processor in paging mode and makes available rqv_ system calls for flat model applications
pcisrv.job	all	all	PCI server for support of SCSI devices
rbootsrv.job	all	all	Services requests from clients that boot remotely on the LAN
remotefd.job	all	all	iRMX-NET client/remote file driver for transparent file access
rintmjob.job	all	all	Remote INtime Personality Job
rip.job	all	All	Part of TCP/IP
rnetserv.job	all	All	iRMX-NET server for transparent file access
rtcimcom.job	all	All	Serial Interface between NTXproxy.job and serdrvr.job
rtcimudp.job	all	All	UDP Interface between NTXproxy.job and the new TCP/IP stack
sdb.job	all	All	System Debugger (SDB), which is aware of OS objects
serdrvr.job	all	PC	Ports-based serial driver (COM1 and COM2)
ssk.job	all	All	Soft-Scope Kernel job
tcp.job	all	all	Part of TCP/IP
telnetd.job	all	all	Telnet Server Job (loads PTTY driverr)
tulip.job	all	all	Network interface driver for DEC 21143 PCI-based NIC
udp.job	all	all	Part of TCP/IP

Table 4-1. Loadable System Jobs and File Drivers (continued)

Loadable Device Drivers

Table 4-2 lists the loadable device drivers supplied with the OS. Each driver makes available one or more DUIB names. Use the DUIB name in the **attachdevice** command for access to the device.Loadable Device Drivers

> **Note**
> When you load a driver and attach a device under the iRMX OS, DOS can no longer use that device. For DOS access, you must first detach the device .

Table 4-2. Loadable Device Drivers

Driver	OS	BUS	Purpose
atapidrvv	DRMX/ RPC	all	Driver for ATA and ATAPI compliant devices at the IDE interface(i.e. Harddisk and CDROM devices)
atcsdrv	all	MBII	Asynchronous Terminal Controller Server (ATCS) terminal driver
comdrv	all	all	Driver for PC COM*n* serial ports
drv82530	all	all	Terminal driver for serial ports on an SBX 354 module
h550drv	DRMX/ RPC	PC	Terminal driver for a Comtrol HOSTESS 4- or 16-channel serial controller board in a PC system
lpdrv	DRMX/ RPC	all	Driver for PC parallel ports
pcidrv	all	all	PCI driver for Multibus II and Adaptec SCSI controllers
pcxdrv	DRMX/ RPC	PC	Device driver for the Digiboard PC/*X* terminal controller board
ramdrv	all	all	RAM disk driver
tccdrv	all	MBI	Terminal driver for Multibus I serial controller boards

Interrupt Encoding

For some device drivers you must specify an iRMX encoded interrupt on the invocation line. Devices installed in a PC system are typically configured to a DOS interrupt request (IRQ). To translate the DOS IRQ to an iRMX encoded interrupt, use Table 4-3. The table shows the typical assignment of IRQs in a PC, organized according to the programmable interrupt controller (PIC). This table is only valid for PC-compatible architectures, and may not be correct if you can change the interrupt configuration of your system hardware.

Table 4-3. DOS Interrupt Requests and iRMX Encoded Interrupts

Master PIC			Slave PIC		
DOS IRQ (decimal)	**Encoded Int (hexadecimal)**	**Typical PC Use**	**DOS IRQ (decimal)**	**Encoded Int (hexadecimal)**	**Typical PC Use**
0	8	Interval timer	8	20	Clock
1	18	Keybd out buffer	9	21	Redirect to IRQ2
2	(21)*	Slave PIC	10	22	Available
3	38	COM2	11	23	Available
4	48	COM1	12	24	Auxiliary mouse
5	58	LPT2	13	25	Coprocessor
6	68	Floppy controller	14	26	Hard disk
7	78	LPT1	15	27	Available

* Boards set for IRQ 2 and IRQ 9 have the same iRMX encoded interrupts, because the PC-compatible ROM BIOS redirects IRQ 9 to IRQ 2.

Note

Lower-numbered encoded interrupts have higher iRMX priority.

atapidrv

A loadable device driver that controls ATA and ATAPI devices at the IDE interface.

Syntax

```
sysload /rmx386/jobs/atapidrv
```

Additional Information

In conjunction with the iRMX Named (NAMEDFD.JOB) or DOS (DOSFD.JOB) file drivers, the atapidrv driver can read and write Hard Disks that conform to the ATA and ATAPI specifications.

Standard DUIBs for the Hard Disk devices are as follows:

- HDA = Hard Disk drive on master of first IDE interface
- HDB = Hard Disk drive on slave of first IDE interface
- HDC = Hard Disk drive on master of second IDE interface
- HDD = Hard Disk drive on slave of second IDE interface

Partitioning is also supported. Standard DUIBs for each partition on Hard Disk device master on the first IDE interface are as follows:

- HDA0 = Entire Hard Disk drive on master of first IDE interface
- HDA1 = Partition 1 of the Hard Disk drive on master of first IDE interface
- HDA2 = Partition 2 of the Hard Disk drive on master of first IDE interface
- HDA3 = Partition 3 of the Hard Disk drive on master of first IDE interface
- HDA4 = Partition 4 of the Hard Disk drive on master of first IDE interface

In conjunction with the CD ROM file driver (CDROMFD.JOB), the atapidrv

ATA and ATAPI driver can read CD-ROM devices which conform to the ATAPI CD-ROM specification r2.6. Specifically excluded from support are drives which do not report a sector size of 2048 when queried.

Standard DUIB for the CD ROM devices are as follows:

- CDA = CD-ROM drive on master of first IDE interface
- CDB = CD-ROM drive on slave of first IDE interface
- CDC = CD-ROM drive on master of second IDE interface
- CDD = CD-ROM drive on slave of second IDE interface

escription

atcs279.job

The Asynchronous Terminal Controller Server (ATCS) manages Multibus II terminal controller boards. This is a loadable version of the ATCS/279/ARC server job. Start this job for each terminal controller board, then start an ATCS device driver on all hosts that use the board through the job.

See also: *atcsdrv*, in this chapter
ICU-configurable version of ATCS/279/ARC server job, Chapter 6, and Appendix C for details of how to use the job

Syntax

```
sysload /rmx386/jobs/atcs279.job [dev_name]
```

Parameter

`dev_name`

The device name that the server job will use as a serial device. If you do not specify a name, the default is `t82530_0`.

Additional Information

The ATCS/279/ARC job typically manages the system console. It operates in one of two ways:

- If there is an SBX 279 graphics board on the host where the server is configured, the server manages this graphics terminal, with multiple windows available for other CPUs on the bus.
- If there is not an SBX 279 board, the server multiplexes a character-based terminal on this host between CPUs on the bus; you switch CPUs with a hot key.

Note
If the ATCS/ARC Server is configured for the `t82530_0` device (no SBX 279), tasks at priority 255 do not execute. The workaround is to make sure that all application tasks execute at priority 254 or higher (numerically lower).

atcsdrv

Device driver for Multibus II serial controllers that support Asynchronous Terminal Controller Server (ATCS) protocol. You can use this driver to access an ATCS job started by another OS on another host board in the system. Otherwise, you must download the ATCS software with a **dload** command before loading this driver.

See also: **dload**, Appendix B

Syntax

```
sysload /rmx386/drivers/atcsdrv ('board_name', board_instance,
                                 num_channels, duib_id)
```

Parameters

`'board_name'`

The name of the board as recorded in its interconnect space. Enclose the string with single quotes. Options are:

Board Name	Channels	Comments
'186/410'	6	Must download ATCS SW from this board
'186/450'	12	Must download ATCS SW from this board
'386/258'	5	With SBX 279 module attached
'386/258D'	5	With SBX 279 module attached
'486/133SE'	5	With SBX 279 module attached
'486/166SE'	5	With SBX 279 module attached
'MIX386/020'	12	Per associated MIX450(S) or MPI450(S)
'MIX386020A'	12	Per associated MIX450(S) or MPI450(S)
'MIX486020A'	12	Per associated MIX450(S) or MPI450(S)
'386/120'	12	Specify this or any other CPU board name to access an MPI450(S) controller associated with that board.

`board_instance`

A value of 1 (hexadecimal, no H suffix) specifies the first board of this type (`board_name`) from slot 0, 2 the next, and so on. This is not the slot ID.

`num_channels`

The number of serial channels on this board, in decimal.

`duib_id`

A single letter from A - Z. It is placed in the DUIB name for devices attached using this instance of the driver. For example, if you load the driver for the first 186/410 board with `duib_id` = B, and the next with C, the DUIB names available for attach are `t_atcs_b0` through `t_atcs_b5` and `t_atcs_c0` through `t_atcs_c5`.

DUIB Names

DUIB names are constructed from `duib_id` and `num_channels`:

```
t_atcs_<duib_id><num_channels>
```

For example, if there are three MIX450 terminal controllers attached to a MIX386/020 board, and you specify A for `duib_id` and 36 for `num_channels`, the driver makes available 36 DUIB names: `t_atcs_a0` through `t_atcs_a35`.

Note

When operating from a board that hosts an SBX 279 terminal controller, you cannot use the DUIBs provided by *atcsdrv* to access the 279 terminal. Instead use the local 279 DUIB names.

See also: Table of local 279 DUIB names, Appendix C

Additional Information

An ATCS device is a buffered terminal driver supporting block input and output of data with solicited and unsolicited messages. The driver has these characteristics:

- Supports **read**, **write**, **special**, **attach_device**, **detach_device**, **open**, and **close**.
- Supports getting and setting terminal data, setting signal characters, setting special characters, setting link parameters, and enabling and disabling flow control with **special**.
- Recognizes baud rates 110, 150, 300, 600, 1200, 2400, 4800, 9600, and 19200. If an unsupported baud rate is requested, the driver overrides it with the next higher supported baud rate.
- Supports all TSC features including special-character interrupts, with these exceptions.
  - The buffered-device features not supported are configurable start/stop input characters and high/low-water marks. The controller always uses XON and XOFF for the start and stop characters, and fixed values for high- and low-water marks.
  - Separate input and output baud rates for a single serial line are not supported.

Multiple hosts can share a single ATCS controller or ATCS/279 server, but multiple hosts cannot simultaneously share the same serial line on the controller.

Note

If the ATCS driver is configured to strip the parity bit on input characters, special characters (07FH and above) are not processed properly if they also happen to be signal characters.

bootserv.job

An implementation of the MSA bootserver protocol. When a host requests bootstrapping, the bootserver reads the dependent second stage from disk and sends it to that boot client. The dependent second stage runs on the boot client and issues a request to the bootserver to load the target file containing the OS.

See also: *MSA Bootstrap Specification*

Syntax

```
sysload /rmx386/jobs/bootserv.job (config_file)
```

Parameter

`config_file`

The pathname for the bootstrap loader configuration file containing Bootstrap Parameter String (BPS) entries. If omitted, the default value */msa/config/bps* is used. If */msa/config/bps* or the specified file does not exist, the bootserver job fails to initialize properly.

See also: Bootserver Job, Chapter 6

Additional Information

Bootserv.job is intended for use with the SBC 486DX33 and SBC 486SX25 embedded workstation boards running DOSRMX or iRMX for PCs, so that they may act as the MSA bootserver.

See also: BPS parameters, *MSA for the iRMX Operating System*

Note

When you include client/server jobs in your system (such as ATCS, PCI, etc.), the following problem can occur: Since every Multibus II board uses a Multibus II broadcast to find the status of the boot master and Quasi-Independent master, the message queues can fill on bootserver boards. This can lock up the system if the bootserver does not flush the messages. To ensure that the bootserver runs and can flush the messages, make the priority of your application task lower (numerically higher) than the bootserver task priority.

Other workarounds are to use the Master Test Handler to set offline any Multibus II boards that will not be booted, or to remove them from the chassis.

cdromfd.job

A loadable file driver that allows access to ISO 9660 CD-ROMs. Access to DATA sectors are supported; none of the various extensions to ISO 9660 are currently supported.

Syntax

x

Parameters

x Definition.

Additional Information

Description

clib.job

A library of C functions that can be shared by all applications in the multitasking environment.

Syntax

```
sysload /rmx386/jobs/clib.job
```

This loadable version of the shared C library layer includes functions for both floating-point and non-floating-point applications. It also includes all available I/O functions. In ICU-configurable systems, you can either load *clib.job* or configure the C library with the ICU. The ICU makes available smaller versions of the C library for applications that do not require certain functions.

In DOSRMX and iRMX for PCs, you can configure some aspects of *clib.job* by changing values in the *rmx.ini* file. You cannot do this in the iRMX III OS; in this case use the ICU version of the C library to set configuration options.

See also: C Library [CLIB] block, Chapter 5,
C Library, Chapter 6,
Function descriptions, *C Library Reference*

Configuring Environment Variables

The C library supports environment variables using the **getenv** and **putenv** functions. A single environment applies to all tasks and jobs sharing the C library. The environment context is not maintained on a per-task or per-job basis. The application's environment is set up on the first call to **getenv** by the application.

You can configure environment variables in the *:config:r?env* file, using the form:

```
environment_variable = ascii_value
```

Each entry is on a separate line, terminated by a carriage-return/line-feed character. Spaces are required around the equals sign (=). The default *:config:r?env* file contains a string specifying the Pacific standard and daylight time zones:

```
TZ = PST8PDT
```

The internal environment table can hold up to 40 environment variables. For each entry the maximum-length variable name is 16 characters and the maximum-length value is 80 characters.

See also: **getenv**, **putenv**, *C Library Reference*

comdrv

Terminal drivers for the COM*n* serial ports on a PC. Invoke the **sysload** command once for each driver to be loaded.

Syntax

```
sysload /rmx386/drivers/comdrv (port,io_addr,encoded_int)
```

Parameters

`port` The number of the COM port; for example, 3 specifies COM3.

`io_addr`
Base I/O address (in hex, no H suffix) used by this COM port. Specify the value used in the setup of the hardware.

`encoded_int`
An encoded interrupt, indicating the interrupt level configured for this port. This is a hexadecimal byte value (not ending in H) encoded as shown in Table 4-3 on page 51.

DUIB Names

com1 com2 com3 com4

Additional Information

In DOSRMX and iRMX for PCs, support for the COM1 and COM2 serial ports is built in, with these default values:

Port	I/O Address	DOS IRQ	iRMX Encoded Interrupt
COM1	3F8H	4	48H
COM2	2F8H	3	38H

If you use only these two ports, and they are set to the I/O addresses and interrupts shown above, you do not need to load *comdrv*. If your hardware ports for COM1 and/or COM2 are configured differently, load *comdrv* and specify the correct values. To add support for a COM3 or COM4 port, load *comdrv* for those ports.

PC systems may assign COM3 to the same interrupt as COM1, and assign COM4 to the same interrupt as COM2. *Comdrv* does not support this; each COM port must be set to a different I/O address and interrupt.

In the iRMX III OS, this driver can only be used for the SBC 486SX25 or 486DX33 board. Load the driver to support the COM1 and/or COM2 serial port, overriding the ICU settings for these ports (ENC parameter on the IDEVS screen).

Hardware Information

This driver is for serial ports using an NS16450 or compatible component. The driver communicates directly between the adapter(s) and the iRMX BIOS. The driver is not buffered; characters are transmitted or received one at a time. The driver has these characteristics.

- Modem Control: The default is no modem. If you configure the driver for a modem, the driver signals Data Terminal Ready (DTR) and RTS when a unit attaches. It also passes ring-interrupt or carrier-loss indicators to the TSC from the communications adapter. The DTR signal clears or asserts on an Answer or Hangup request from the Terminal Support Code. When a unit detached, the driver clears DTR and RTS.

  See also: Configuring terminals for a modem, Chapter 2

- Baud: The driver does a baud-rate search when a unit is attached; it recognizes 110, 150, 300, 600, 1200, 2400, 4800, and 9600. It supports 19200 baud rate if the particular communications adapter supports it, but cannot detect the 19200 rate automatically. You must explicitly set a 19200 baud rate in the UINFO table for the device. The output and input baud rates are always the same.

- Parity: The default is 8-bit, no parity. If parity checking is enabled, the driver sets or clears the most significant bit of the received character when an error is detected. The action depends on the binary value of bits 4 and 5 in the `terminal_flags` field of the driver's UINFO table. To change these bits:

  — In ICU-configurable systems, use the ICU to change bit values in the Serial Driver's IPC, OPC, RPC, and WPC parameters.

  — Change settings programmatically with the **special** system call or dynamically using Operating System Command (OSC) sequences.

  — Rebuild the loadable driver with these bits set.

See also: Making a device driver loadable, *Driver Programming Concepts*

Note

If you use this driver for modem support, the following problem can occur. If a user logs in with this driver and the phone connection is dropped for some reason, the HI session is not deleted and the user is not logged off. The next person to dial in at the port is not prompted for any logon information but is instead returned to the earlier session started by the previous user.

dosfd.job

Loadable version of the native DOS file driver. The native DOS file driver allows iRMX for PCs to read and write DOS volumes.

Syntax

```
sysload /rmx386/jobs/dosfd.job
```

DUIB Names

Name	DOS Device(s)
AH	Floppy drive A:, 5.25" high-density
AM	Floppy drive A:, 3.5" low-density
AMH	Floppy drive A:, 3.5" high-density
AMO	Floppy drive A:, 3.5" quad-density
BH	Floppy drive B:, 5.25" high-density
BM	Floppy drive B:, 3.5" low-density
BMH	Floppy drive B:, 3.5" high-density
BMO	Floppy drive B:, 3.5" quad-density
C_DOS thru Z_DOS	Hard drives/partitions C: through Z:

See also: Multibus I and II DUIBs, Appendix E, *Command Reference*

Additional Information

The DOS file driver does not recognize DOS volumes formatted by versions of DOS before 3.3. This file driver overcomes most of the restrictions DOS places on file I/O operations. Files can be renamed to any subdirectory on a volume. Directories have a file size associated with them. Directory time stamps are updated when a file is created or deleted. File I/O speed is increased considerably.

You can use the DTP entry in *rmx.ini* to specify the I/O task priority for *dosfd.job*.

CAUTION
Do not load this file driver in DOSRMX systems. Only use the EDOS file driver in DOSRMX.

The DOS file driver also supports all the Multibus I and II DUIBs such as `wqfo`. To use these DUIBs, after loading *dosfd.job*, specify the `dos` switch in the attachdevice command; for example:

```
ad wqfo as q dos
```

drv82530

A terminal driver that supports the two serial ports on an SBX 354 module. This is a loadable version of the same driver that you can configure on the D2530 screen of the ICU.

Syntax

```
sysload /rmx386/drivers/drv82530 (encoded_int,
      data_io_addr_a, ctrl_io_addr_a, baudrate_a,
      data_io_addr_b, ctrl_io_addr_b, baudrate_b)
```

Parameters

encoded_int

An encoded interrupt, indicating the interrupt level configured for the driver. The same interrupt applies to both channels. This is a hexadecimal byte value (not ending in H) encoded as shown in Table 4-3 on page 51.

data_io_addr_a
data_io_addr_b

The hexadecimal port address of the data ports for Channels A and B, respectively.

ctrl_io_addr_a
ctrl_io_addr_b

The hexadecimal port address of the control/status ports for Channels A and B, respectively.

baudrate_a
baudrate_b

The baud rate for Channels A and B, in hexadecimal.

DUIB Names

t82530_0 t82530_1

Additional Information

Enter all parameters as hexadecimal values. Do not enter an H at the end of the values.

edl.job

NIC job that provides an interface between the new TCP/IP stack and an iNA job. .

Syntax

```
sysload /rmx386/drivers/edl.job ifport=<iNA subnet number>,
            ntrans=<maximum number of outstanding transactions>
            ncbs=<number of control buffers>
```

Parameters

ifport

The iNA subnet ID of the iNA job being whose services the new TCP/IP stack will use. Use the **lanstatus** command to obtain the subnet ID.

ntrans

The number of transaction buffers that can be outstanding at any given time

*ncbs*The

The maximum number of control buffers available to this job.

Additional Information

The EDL NIC interface job can be used with all iNA jobs except the i552A.job.

eepro100.job

Driver for the Intel EtherExpressPro 100 Plus PCI-based NIC.

Syntax

```
sysload /rmx386/drivers/eepro100.job ntrans=<maximum
      number of outstanding transactions>
      ncbs=<number of control buffers>
```

Parameters

ntrans

The number of transaction buffers that can be outstanding at any given time

ncbs

The maximum number of control buffers available to this job.

flat.job

Provides the entire flat model support code. Load this job along with *paging.job* to support flat model applications. .Once **flat.job** is loaded, flat model applications can make iRMX system calls and C library calls.

Syntax

```
sysload /rmx386/jobs/flat.job
```

Additional Information

You can load this job at any time the iRMX OS is running. There are no command line options for the job.

Complete support for flat model applications requires the paging subsystem, which you can load as *paging.job* or configure in the ICU as a first-level job. However, you cannot enable *flat.job* as a first-level job.

Errors and initialization messages from *flat.job* are reported in the *:config:flat.log* file.

See also: Flat Model applications, *Programming Techniques*

h550drv

A device driver for the Comtrol HOSTESS multi-port terminal controller board installed in a PC system.

Syntax

```
sysload /rmx386/drivers/h550drv (num_channels, io_addr, encoded_int)
```

Parameters

`num_channels`
: The hexadecimal number of serial channels on this board: 4, 8, or 10 (for 16-channel support). Do not specify an H suffix.

`io_addr`
: The base I/O address (in hex, no H suffix) configured on the HOSTESS board.

`encoded_int`
: An encoded interrupt (in hex, no H suffix), indicating the interrupt level configured on the HOSTESS board. This is encoded as shown in Table 4-3 on page 51.

DUIB Names

t550_0 t550_1 t550_2 t550_3 t550_4 t550_5 t550_6 t550_7

Additional Information

This driver supports only HOSTESS boards with UARTs that can be programmed to have a 16-byte input FIFO, such as the 16550 component.

All channels are configured as 9600 baud with 8 data bits, no parity, and 1 stop bit. Configure the board by setting switches for the base address and interrupt level.

See also: Hardware configuration, *Installation and Startup*

For example, on an 8-channel board, if you configure base I/O address 280H and IRQ 5, invoke the **sysload** command as shown below. Then to access the first channel of the controller with the logical name *:t:*, attach it as a physical device:

```
sysload /rmx386/drivers/h550drv (8,280,58)
attachdevice t550_0 as t p
```

Note

The HOSTESS board's default jumper setting and **sysload** command in *loadinfo* are for IRQ 5, which might be used by the LPT2 parallel port, or a PCLINK or EtherExpress™ board. If so, reconfigure the board and change the interrupt value in *loadinfo*.

i*.job

There are several iNA 960 jobs that provide a programmatic interface to the ISO transport software. The jobs are specific to a system bus architecture because each bus requires different Network Interface Card (NIC) hardware. The iNA 960 jobs are referred to collectively as *i*.job*.

To add iRMX-NET transparent file access and a user interface, load *remotefd.job* and/or *rnetserv.job* in addition to the appropriate *i*.job*.

See also: *Network User's Guide and Reference* for information about the programmatic interface to these jobs and about using subnet IDs

Syntax

```
sysload /rmx386/jobs/i*n.job
sysload /rmx386/jobs/i*e.job [SNID1=xxxxH] ... [SNID4=yyyyH]
sysload /rmx386/jobs/ipcl2.job (io_addr, encoded_int, mem_addr)
```

Note

The Null2 (non-routable) iNA 960 jobs do not have command-line parameters. The ES-IS (routable) jobs allow you to set the subnet ID(s). For the *ipcl2.job* you set specific parameters to match the PCL2(A) board. Choose the appropriate job name from Table 4-4 on page 72 or Table 4-5 on page 73.

If the system hangs when you load one of these network jobs, you may need to add memory.

Parameters

`SNID1=xxxxH`

For ES-IS jobs (see jobs that end in "e" in Table 4-5), you can specify the subnet ID(s) the job will use. Specify only as many SNID parameters as the job has subnets. For example, *i2mxmpe.job* in Table 4-5 has 3 subnets: the first two apply to the first two MIX 560 boards and the third applies to the Multibus II subnet. The default configuration for the job assigns subnet IDs 1, 2, and 3 to these subnets. To change the subnet IDs, specify three SNID parameters; for example:

```
sysload /rmx386/jobs/i2mxmpe.job SNID1=0004H SNID2=00FEH SNID3=0001H
```

Separate multiple parameters with spaces. The SNID part must be capital letters. Always specify a four-digit value followed with H.

`io_addr`

For *ipcl2.job* only, the I/O port address configured on the board, specified in hexadecimal but not ending in H. The default address for the PCL2(A) boards is 360 (hex).

See also: Table 4-6 on page 74

`encoded_int`

For *ipcl2.job* only, an encoded interrupt, indicating the interrupt level configured on the board. This is a hexadecimal byte value (not ending in H) encoded as shown in Table 4-3 on page 51. The default value for the PCL2(A) boards is 21 (hex), corresponding to IRQ 2.

See also: Table 4-6 on page 74

`mem_addr`

For *ipcl2.job* only, the base memory address in DOS I/O space where the board communicates. This is the beginning physical address of an 8K block (2000H) that cannot be used by any other device (video card, etc.). Enter a five-digit hexadecimal value ending in 0; do not specify the H. A typical value is `CC000`.

Additional Information

Choose a job to match your computer's hardware and software configuration:

- COMMengine or COMMputer environment
- Operating system and bus type
- NIC, or multiple NICs for jobs that act as routers between subnets

Network Jobs for the COMMengine

The iNA 960 MIP jobs run in a COMMengine environment, which means iNA runs on a separate NIC from the board that hosts the OS and other programs. The MIP jobs in Table 4-4 run on the CPU board with the OS. The NIC can be one of these:

- A separate Ethernet controller, in which case one CPU board using that NIC as a COMMengine must load the appropriate iNA 960 download file on the NIC
- A CPU board running one of the iNA 960 COMMputer jobs; CPU boards using that NIC (the COMMputer) as a COMMengine do not download a file to it

To control downloading of an iNA 960 file, specify the NIC and the name of the download file as parameters in the *rmx.ini* file (DOSRMX or PCs) or the BPS file (iRMX III on a Multibus II system). Table 4-4 lists the NICs supported by the MIP jobs and the default download files for particular boards. The question mark (?) in the filenames represents either an N or an E to specify whether network routing is possible.

See also: [MIP] block, Chapter 5,
`rq_mip_xx` BPS parameter, *MSA for the iRMX Operating System*
MIP, Null2 and ES-IS, *Network User's Guide and Reference*

Table 4-4. iNA 960 COMMengine Jobs (NIC is Different Board from the OS)

Job	OS	NIC	Download File	Bus Type/Comments
ipcl2.job	DRMX/ RPCr	PCL2	INAPCL2?.32L	PC
	DRMX/ RPC	PCL2A	INAPL2A?.32L	
i552a.job	III	552A	INA552A?.32L	Multibus I
icemb2.job	all	186/530	INA530?.32L	Multibus II
		MIX386/560	INA560?.32L	No MSA firmware on baseboard
		any Multibus II COMMputer	none	* Separately-loaded iNA 960

? Specify N for Null2 (no routing capability) or E for ES-IS routing
* Do not download an iNA file if the MIP job uses an iNA 960 COMMputer board as a NIC
RPC iRMX for PCs OS
DRMX DOSRMX OS
III iRMX III OS

CAUTION

If you use a MIP job in a Multibus II system and the NIC used as a COMMengine fails, the MIP job does not receive any notification. If this occurs, software that runs on the board running the MIP job will hang when it attempts to make network requests.

Do not use *i552a.job* on a board where the OS or the application runs above 16 Mbytes. The MIP job for the 552A board cannot send iNA 960 request blocks to addresses on the host board of 16 Mbytes or above. For a configuration with more than 16 Mbytes, use an SBX 586 board and the appropriate iNA 960 COMMputer job.

Network Jobs for the COMMputer

A COMMputer environment means the iNA 960 job runs directly on a host CPU board that includes LAN hardware. Table 4-5 lists the NICs supported by these jobs. The question mark (?) in the filenames represents either an N or an E to specify whether network routing is possible. Jobs with more than one subnet act as routers between subnets, using the multiple NICs indicated in Table 4-5

See also: COMMputer jobs, Null2 and ES-IS jobs, and Multibus II subnet, *Network User's Guide and Reference*

Table 4-5. iNA 960 COMMputer Jobs (NIC is Same Board as the OS)

Job	OS	SN	NICs (and Default Subnet IDs) for 1st, 2nd, 3rd, 4th Subnet
			PCs
iethpro?	DRMX/ RPC	1	EtherExpress PRO/10 (10 Mb) **(1)**
Idec43?.	DRMX/ RPC	1	PCI-based Netrwrok Adapter containing the DEC 21143 NIC
inlatn.job	DRMX/ RPC	1	short-circuit inbox network, no NIC **(1)**
			Multibus I Boards
isbx586?	III	1	SBX 586 on any other CPU board **(1)**
inlmb1.job	all	1	short-circuit inbox network, no NIC **(1)**
			Multibus II Boards
i486133?	III	1	SBC 486/1xxSE **(1)**
ihisxe	III	2	SBC 486/1xxSE **(1)**, SBX 586 **(2)**
ihimpe	III	2	SBC 486/1xxSE **(1)**, MB II backplane **(2)**
ihisxmpe	III	3	SBC 486/1xxSE **(1)**, SBX 586 **(2)**, MB II backplane **(3)**
imix560?	III	1	1 MIX560 **(1)**
imxmpe	III	2	1 MIX560 **(1)**, MB II backplane **(2)**
i2mxe	III	2	2 MIX 560s **(1,2)**
i2mxmpe	III	3	2 MIX 560s **(1,2)**, MB II backplane **(3)**
i3mxe	III	3	3 MIX 560s **(1-3)**
i3mxmpe	III	4	3 MIX 560s **(1-3)**, MB II backplane **(4)**
iewexp?	all	1	SBC 486DXxx with EWENET module **(1)**
ie1mpe	all	2	SBC 486DXxx with EWENET **(1)**, MB II backplane **(2)**
ieproe2?	all	1	SBC P5090 **(1)**
ie2mpe	all	2	SBC P5090 **(1)**, MB II backplane **(2)**
imp?	all	1	MB II backplane only **(1)**
inlmb2.job	all	1	short-circuit inbox network, no NIC **(1)**

SN Number of subnets in this job
? Specify N for Null2 (no routing capability) or E for ES-IS routing
RPC iRMX for PCs

DRMX	DOSRMX
III	iRMX III OS only

Network Jobs That Run Without Network Hardware

Table 4-5 lists these network jobs that support short-circuit local network communications:

inlatn.job
inlmb1n.job
inlmb2n.job

Using these jobs, an application on a PC or Multibus I system can communicate with other jobs in the same system. In a Multibus II system, applications can communicate on the same board or between boards in the system. Load these jobs to support DDE and ISO transport applications that operate only on the local computer.

Unlike other network jobs, there is no load-time configuration for these jobs.

The following jobs also do not require network hardware. Unlike the calls listed above, these jobs offer the advantage of using the Multibus II subnet, which means you can run TCP/IP on multiple boards in the system.

impn.job
impe.job

Configuration for a PCL2(A) board

Table 4-6 lists the jumper settings for the PCL2 and PCL2A boards. The first listing for each board is the default setting. Use the appropriate values in the io_addr and encoded_int parameters for *ipcl2.job.*

Table 4-6. Jumper Settings for the PCL2 and PCL2A Boards

	I/O Addresses		Interrupts		
Board	**Jumpers**	**Address**	**Jumpers**	**IRQ**	**Encoded Interrupt**
PCL2 (default)	E2 - E3	360 (hex)	E11 - E12	2	21 (hex)
	E1 - E2	3C0 (hex)	E10 - E11	5	58 (hex)
PCL2A (default)	E2 - E3	360 (hex)	E11 - E12	2	21 (hex)
	E1 - E2	368 (hex)	E10 - E11	5	58 (hex)
			E13 - E14	4	48 (hex)
			E14 - E15	3	38 (hex)

ip.job

The ip.job implements both the Address Resolution Protocol (ARP) and the Internet Protocol (IP).

Syntax

```
sysload /rmx386/jobs/ip.job
```

Additional Information

ARP is used to dynamically map between Internet software addresses and Ethernet hardware addresses.

ARP caches Internet-to-Ethernet address mappings. When the interface requests a mapping for an address not in the cache, ARP queues the message that requires the mapping and broadcasts a message on the associated network, requesting the address mapping. If ARP receives a response, it caches the new mapping and transmits any pending messages to that host. While waiting for a response, ARP will queue only one packet; it keeps only the most recently transmitted packet.

ARP watches passively for hosts impersonating the local host (that is, a host that responds to an ARP mapping request for the local host's address).

IP is the network layer protocol used by the Internet protocol family. It can be accessed through the higher-level Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) as well as directly through the Raw IP interface.

Outgoing packets automatically have an IP header prepended to them. The header is based on the destination address and the protocol number the transport endpoint is created with. Incoming packets are automatically stripped of their IP header before being sent upstream.

keybd.job

This job provides the d_cons device (DOS console) for keyboard input and screen output. It can be used with 84- and 101-key keyboards. You can specify the foreground and background colors, and a screen-saver time limit.

Syntax

```
sysload -w /rmx386/jobs/keybd.job [(foreground, background,
                                    ss_delay)]
```

Parameters

foreground
: Foreground color (in hex, no H suffix) from 0 to F.

background
: Background color from 0 to 7. Do not set foreground and background to the same color.

ss_delay
: Screen-saver delay in 10 ms increments. If no screen activity occurs in this period, the screen contents are saved to a file and the screen blanks. FFFF disables the screen saver.

Color codes are (shades may look different from monitor to monitor):

Value	Color
0	Black
1	Blue
2	Green
3	Cyan
4	Red
5	Magenta/Purple
6	Brown/Orange
7	Light Gray
8	Dark Gray
9	Light Blue
A	Light Green
B	Light Cyan
C	Light Red
D	Light Magenta
E	Yellow
F	White

If no parameters are specified, the default is light gray on black (7,0) with no screen-saver.

DUIB Name

d_cons

Additional Information

The d_cons device is only useful as an HI terminal, so *keybd.job* must be loaded in the *loadinfo* file and d_cons must be specified in the *:config:terminals* file. In the *loadinfo* file, unlock d_cons after loading *keybd.job*.

In DOSRMX, this job provides the console-sharing code that enables you to toggle between DOS and the iRMX OS.

These control codes in the *:config:termcap* file can enable foreground and background colors to be saved, as well as restoration of the previous screen colors upon termination of the **aedit** text editor:

```
AFRV = 0B5F; AFNV = 0B0A; AFST = 15010B0A; AFEN = 15021503;
```

See also: Terminal configuration file, Chapter 2,
Loading and unlocking terminal devices, Chapter 3,
Console Output Codes, Appendix A

lpdrv

Device driver for the LPT1, LPT2, and LPT3 parallel ports.

Syntax

```
sysload /rmx386/jobs/lpdrv
```

DUIB Names

lpt1 lpt2 lpt3

Additional Information

Lpdrv provides an interface between the iRMX BIOS physical file driver and the parallel I/O ports in a PC system. The driver has these characteristics:

- Supports the **write, attach_device,** and **detach_device** system calls
- Accepts the **open** and **close** system calls but does no operations for them

To make the LPT2 port available with the logical name *:lpt2:*, attach the appropriate DUIB name as a physical device:

```
attachdevice lpt2 as lpt2 p
```

namedfd.job

Loadable version of the native iRMX Named file driver. The native iRMX Named file driver allows iRMX systems to read and write iRMX Named32 and Named48 volumes.

Syntax

```
sysload /rmx386/jobs/namedfd.job
```

Additional Information

The Named file driver supports file access of iRMX Named32 and Named48 file systems on a device controlled by the iRMX OS. The DUIBs associated with this file driver are those provided by the iRMX driver controlling the device. For instance, in an iRMX for PCs system where the first partition of the C Drive has been formated as an iRMX Named voulme, the DUIB name utilized will be HDA1 if using the ATAPIDRV driver and c_rmx1 if using the ROMBIOS-based Wini driver.

ne.job

Driver for a number of NE2000 compatible ISA NICs.

Syntax

```
sysload /rmx386/drivers/ne.job irq=<PC IRQ>
      base=<I/O base address>
      ntrans=<maximum number of outstanding transactions>
      ncbs=<number of control buffers>
```

Parameters

irq

The IRQ level associated with this device

base

The base I/O address associated with this device

ntrans

The number of transaction buffers that can be outstanding at any given time

*ncb*s

The maximum number of control buffers available to this job.

Additional Information

Use the DOS setup software that comes with the card to configure the IRQ and Base I/O Address for your NE2000 compatible card as it resides in your PC system.

ntxproxy.job

The NTX Proxy job provides the NTX interface between the Remote INtime Personality job and the loaded Channel Interface Module (CIM). It processes NTX commands from a Windows NT Host system.

Syntax

```
sysload /rmx386/jobs/ntxproxy.job -Name <node_name>
```

Parameters

`Node_name`

The name of the remote node as configured on the Windows NT Host using the iRMX III NTX Link Setup utility

Additional Information

See also:Preparing the NT Host System, Chapter 8

paging.job

Starts the paging subsystem, which manages memory with the processor in paging mode. This job, together with *flat.job*, support flat model applications produced with non-Intel flat model compilers. Also, *paging.job* makes available the **rqv_** system calls used by flat model applications.

Syntax

```
sysload  /rmx386/jobs/paging.job [mem_start, mem_end] [, ...]
```

Parameters

`mem_start, mem_end`

The beginning and end addresses specifying one or more blocks of physical memory that the paging job identity-maps. Specify the addresses in hexadecimal without an H suffix. The starting address is always rounded down to the nearest 4 Kbyte boundary and the ending address is always rounded up to the nearest 4 Kbyte boundary. You can define up to eight such memory blocks. The blocks should not overlap; a memory block that overlaps a previously defined block will be ignored by the paging subsystem.

Note

Any blocks that you specify for identity-mapping must be outside the range of physical memory managed by the Free Space Manager. Specifying memory here does not add it to Free Space memory.

Additional Information

Identity mapping means that the paging subsystem maps virtual memory to the same address as physical memory, so the memory is accessible by applications, device drivers, etc. For example, you might define such memory for use as dual-port memory for I/O cards.

If you don't specify any memory block parameters, the paging subsystem identity-maps only the physical memory known to the Nucleus Free Space Manager.

See also: FSM screen in the ICU for Free Space Manager memory

Any physical memory that is not either managed by the Free Space Manager or identity-mapped by the paging subsystem will not be accessible once paging is enabled.

You can load *paging.job* any time while the system is running, or you can instead configure the paging subsystem as a first-level job in the ICU.

The loadable job reports errors and initialization messages to the *:config:paging.log* file. Initialization messages include the identity memory map created by the paging subsystem. Check the log file to verify that the actual physical memory has been identity-mapped correctly. Entries in the *paging.log* file are similar to this:

```
Paging enabled
Identity mapped physical memory from 00000000 to 007FFFFF
Identity mapped physical memory from FA000000 to FAFFFFF
```

See also: Flat Model applications, *Programming Techniques*

pcidrv

Device driver for a Peripheral Controller Interface (PCI) Server on a SCSI controller board in a Multibus II system or Adaptec SCSI host adapter in a PC system. Load this driver once for each instance of a PCI Server in the system.

Syntax

```
sysload /rmx386/drivers/pcidrv ('board_name',board_instance,
                                server_instance,duib_id)
```

Parameters

`'board_name'`

The name of the board to which the SCSI drive is attached and where a PCI Server job is running. Enclose the name in single quotes; this is the name of the board as stored in its interconnect space. Possible values are `'386/258`, `'386/258D'`, `'486/133SE'`, or `'AT'`. The `'AT'` entry assumes an Adaptec 1542/1742 SCSI host adapter (1742 must be in 1542 compatibility mode for the device driver to function correctly).

`board_instance`

A hexadecimal value (no H suffix) specifying the instance of this board type in the system. This is not the slot ID; 1 specifies the first board of the type indicated in `board_name`, counting up from slot 0; and 2 specifies the second such board, etc. These values apply to MBII systems.

`server_instance`

The instance of the PCI Server running for this board. 0 specifies the first instance and 1 specifies the second. The server instance matches up with the server_instance parameter, part of the rmxini_block_name, in *pcisrv.job*.

`duib_id`

A single letter from A - Z (not case-sensitive) that will uniquely identify the DUIB name. Specify A the first time you invoke the driver, B the second time, etc.

DUIB Names

Tables 4-7 and 4-8 list the DUIB names made available with the PCI driver. In these DUIB names, substitute the `duib_id` value from the **sysload** command for the letter D and substitute the appropriate value of SCSI ID for N.

A set of special DUIB names support partitioned drives; the names have the format `gscw5_DNMx` or `gscw5_DNMxEy`. In these names substitute D and N values as for other DUIB names. The M and E are part of the names. For x and y, substitute:

x The number of the Master Boot Partition record, in the range 1-4.

y The decimal number of the Extended Logical Drive, if this is an Extended partition. Only Master Boot partitions 2-4 can be Extended partitions.

See also: **rdisk** command and Partitioning PCI Drives, *Command Reference*

Table 4-7. Hard Disk DUIB Names for pcidrv Driver

DUIB Name	Device Type	SCSI-ID (N)	Bytes/Sector
scw_DN *	Generic SCSI	2,3,4,5	1024
gscw5_DN	Generic SCSI	2,3,4,5	512
gscw5_DNMx	Generic Partitioned SCSI	2,3	512
gscw5_DNMxEy	Generic Partitioned SCSI	2,3	512
gscw_DN	Generic SCSI	2,3,4,5	1024
m3170_DN	Maxtor XT-3170S	2,3,4,5	1024
m4170_DN	Maxtor XT-4170S	2,3,4,5	1024
m4380_DN	Maxtor XT-4380S	2,3,4,5	1024
m8380_DN	Maxtor XT-8380S	2,3,4,5	1024
m8760_DN	Maxtor XT-8760S	2,3,4,5	1024
hp97536_DN	Hewlett-Packard 97536	2,3,4,5	1024
s2300_DN	Siemens Megafile 2300	2,3,4,5	1024
mp1578_DN	Micropolis 1578	2,3,4,5	1024
q280_DN	Quantum Pro 40S/80S	2,3,4,5	1024
wren5_DN	Wren 5	2,3,4,5	1024

In DUIB names, D = *duib_id* from the **sysload** command and N = SCSI-ID.

* Do not format a drive attached with this DUIB name.

For example, if you load *pcidrv* twice for two instances of a 486/133SE board:

```
sysload /rmx386/drivers/pcidrv ('486/133SE',1,0,A)
sysload /rmx386/drivers/pcidrv ('486/133SE',2,0,B)
```

You could attach SCSI-ID 2 on the first board with DUIB name `scw_a2`, and SCSI-ID 2 on the second board with DUIB name `scw_b2`.

Table 4-8. Other DUIB Names for pcidrv Driver

TAPE DRIVES

DUIB Name	Device Type	SCSI-ID (N)	Bytes/Sector
wtaD0	Archive 2125S	6	N/A
OPTICAL DRIVES			
DUIB Name	**Device Type**	**SCSI-ID (N)**	**Bytes/Sector**
OPT1gigDN	Maxoptix Tahiti II	0,1,2,3,4,5	1024
OPT1gig5DN	Maxoptix Tahiti II	0,1,2,3,4,5	512
OPT1650mDN	Maxoptix Tahiti II	0,1,2,3,4,5	1024
OPT1650m5DN	Maxoptix Tahiti II	0,1,2,3,4,5	512

DISKETTE DRIVES

DUIB	Size	Device Type	SCSI Adapter	SCSI-ID	Density	Bytes/Sector
wqfD0	5.25"	Teac 55GFR	NCR ADP-20	0	high	512
wqfD1	5.25"	Teac 55GFR	NCR ADP-20	1	high	512
wdfD0	5.25"	Teac 55BV	NCR ADP-20	0	double	512
wdfD1	5.25"	Teac 55BV	NCR ADP-20	1	double	512
wmfD0*	5.25"	Teac 55BV	NCR ADP-20	0	double	512
wmfD1*	5.25"	Teac 55BV	NCR ADP-20	1	double	512
t55_D0	5.25"	Teac 55GFR**	N/A	0	high	512
t55_D1	5.25"	Teac 55GFR**	N/A	1	high	512
t55d_D0	5.25"	Teac 55GFR**	N/A	0	double	512
t55d_D1	5.25"	Teac 55GFR**	N/A	1	double	512
t235_D0	3.5"	FD-235HF**	N/A	0	high	512
t235_D1	3.5"	FD-235HF**	N/A	1	high	512

In DUIB names, D = *duib_id* from the **sysload** command and N = SCSI-ID.

* wmf0/1 diskettes are standard-granularity (320 Kbyte with 128-byte sectors on track 0) and can be read only if they are formatted on a Multibus I system with the parameters: format :F:disk ms=0 ext=41 They cannot be written to or formatted on a SCSI device.

** The SCSI adapter is part of this drive. No separate SCSI adapter board is required.

Additional Information

Note

If you are using an Adaptec SCSI host adapter, do not load the ASPI (Advanced SCSI Programming Interface) driver that comes with the board. It is incompatible with the PCI server and will take control of the host adapter, even though the iRMX OS intends to use it.

The *pcidrv* driver has these characteristics:

- Supports **read**, **write**, **seek**, **special**, **attach_device**, and **detach_device**.
- Accepts **open** but does no operation for it.
- Accepts **close** for disk controllers but does no operations for it.

  For tape drives, **close** terminates read or write mode. If terminating the read mode, the tape advances to the next file mark or the end of tape. If terminating the write mode, any pending output completes and writes a file mark before completing the **close** operation.

The driver supports these subfunctions of the **special** calls:

- Format track
- Get device characteristics
- Rewind tape
- Read tape file mark (forward searching only, one or more file marks)
- Write tape file mark
- Get bad track or sector information
- Set bad track or sector information
- Re-tension tape
- Return device specific status

The driver improves hard disk integrity with the seek-on-detach feature, in which the disk heads seek to the innermost cylinder (usually the diagnostic cylinder) in response to the **f_detach_dev** command.

To format a hard disk, either attach it with a device-specific name or with one of the `gscw_` names. Device-specific DUIBs contain all the information needed to format the drive. The `gscw_` generic DUIB names query the SCSI drive to get this information. To access a drive that is already formatted with the iRMX **format** command, attach it with either a `gscw_` or `scw_` DUIB name.

pcisrv.job

Loadable version of the PCI server job that directly manages SCSI peripherals. Each instance of a PCI Server in the system requires that *pcidrv* be loaded. The PCI device driver exchanges commands and status with the PCI server; the PCI server manages all SCSI bus transactions. You can use an ICU-configurable PCI server job instead of the loadable job.

See also: PCI server job, Chapter 6

Syntax

```
sysload /rmx386/jobs/pcisrv.job (rmxini_block_name)
```

Parameters

`rmxini_block_name`

Unique name for a block of entries in *rmx.ini*. The entries contain configuration information for the Adaptec SCSI controller in use: e.g., base address, interrupt level, server instance. The server instance matches up with the `server_instance` parameter in *pcidrv*. 0 specifies the first instance and 1 specifies the second.

If no *rmx.ini* entry is specified, the default [PCIAD1] block is used. A block of this type appears in the *rmx.ini* file for each Adaptec SCSI host adapter installed; if you add a second board, you give that block a unique name. Target mode (when one SCSI host adapter attempts to communicate with another SCSI adapter) is not supported.

See also: [PCIAD1] block, Chapter 5

Additional Information

To get the best performance for large sequential I/O transfers, set the PCI direct threshold level using the **pci** command after attaching the device. The direct threshold is the level at which the PCI server's buffers are bypassed (data transfers directly into your buffers). As a starting point, set the direct threshold to the track size of the device. You can then tune it for a particular application with later **pci** commands. For example:

```
pci direct :scw: 32768
```

Note

If you use an Adaptec 1542 controller, its DMA is limited to memory transfers with addresses less than 16 megabytes. For systems with more than 16 Mbyte of memory, the application should allocate I/O buffer space in the lower 16 Mbyte of memory and use the **pci** command to set the direct threshold level to 0. Setting the threshold to 0 ensures that PCI does not copy the data in its own buffers, which may be above 16 Mbyte. Applications requesting I/O services from the controller should use this low memory buffer.

On an DOSRMX or iRMX for PCs system, set Nucleus UML parameters in the *rmx.ini* file to block out the memory.

Create a pointer to the buffer with the **rq_create_descriptor** system call.

See also: **pci**, *Command Reference*;
How to Use the Peripheral Controller Interface (PCI) Server manual

The only HI commands supported for tape drives are **attachdevice**, **detachdevice**, **backup**, **retension**, and **restore**. After you issue a **backup** or **restore** command, the software automatically rewinds the tape.

pcxdrv

Device driver for the Digiboard PC/*X* terminal controller board installed in a PC system.

Syntax

```
sysload /rmx386/drivers/pcxdrv (io_addr, num_channels, duib,
                                encoded_int)
```

Parameters

`io_addr`
: The base I/O address, specified in decimal as 0, 1, 2, 3, or 4. This parameter has no relation to the hardware board number established by jumpers or switches.

`num_channels`
: The number (in decimal) of serial ports on the board. The PC/4 board has 4 channels, the PC/8 board has 8 channels, and the PC/16 has 16 channels.

`duib`
: A number (`x`) from 0 to 31 decimal that forms the base of the unique DUIB names for each instance of this driver. Each *pcxdrv* driver establishes DUIB (physical) names beginning with `pcx_tx`, are assigned sequentially from this number. Each board requires the loading of a new instance of the **pcxdrv** driver. Support is only provided for up to 32 ports, regardless of the number of boards in the system. For example, four PC/4 boards provide 16 ports, four PC/8 boards provide 32 ports, and one PC/16 board and one PC/4 board provide 20 ports. Up to two PC/16, four PC/4, and four PC/8 boards are supported.

  For example, if `num_channel` is 4 and `duib` is 8, duib names are `pcx_t8`, `pcx_t9`, `pcx_t10` and `pcx_t11`. The sum of the `duib` and `num_channel` parameters must be less than 32.

`encoded_int`
: An encoded interrupt, indicating the interrupt level on the PC/*X* board. This is a hexadecimal byte value (not ending in H) encoded as shown in Table 4-3 on page 51. The PC/*X* boards can be jumpered for interrupts IRQ2, IRQ3, IRQ4, IRQ5, IRQ6, or IRQ7. Each board must have a unique interrupt level.

  The example invocation below loads the *pcxdrv* driver as board number 4 with 8 serial channels from `pcx_t4` to `pcx_t11` on interrupt 5. Note that the input is both in decimal and hexadecimal.

```
sysload /rmx386/drivers/pcxdrv (0, 8, 0, 58)
```

Additional Information

Note

This driver does not support the following:

- Use of the **sysload** -w switch
- Daisy-chained PC/*X* boards
- User logon sessions initiated by including an entry in the *:config:terminals* file and unlocking the appropriate port after loading **pcxdrv**

There is a relationship between the interrupt level and ACE I/O port jumper settings on the PC/4 board and PC/8 board. Each ACE has a jumper that is used to select an ODD or EVEN interrupt. This allows a single board to generate more than one interrupt, with specific ACEs generating one or the other interrupt. However, the *pcxdrv* driver does not support more that one interrupt per board. If the level is ODD (3, 5, or 7), the I/O ports must have their interrupt jumper set to ODD. If the level is EVEN (2, 4, or 6), the I/O ports must have their interrupt jumper set to EVEN.

There are no jumpers on the PC/16 board, however board 0 must be set for an ODD interrupt level and board 1 for an EVEN interrupt level.

CAUTION

If the correct interrupt is not set, the PC/*X* board may not work properly.

Board I/O Addresses

Each PC/*X* board contains jumpers and/or switches that set the I/O addresses of the board's status register and each asynchronous communication element (ACE). An ACE is one serial channel; if it is a PC/4 board, there will be four serial channels (ACE chips) on it.

In every case except the second PC/16 board, the hardware board number must be set to 0. If you have a second PC/16 board, use the jumpers to set its hardware board number to 1.

Table 4-9. PC/4 I/O Addresses

Board	Status Port	Port 0	Port 1	Port 2	Port 3
0	188	130	138	140	148
1	288	150	158	160	168
2	208	1B0	1B8	1C0	1C8
3	308	1D0	1D8	1E0	1E8

Table 4-10. PC/8 I/O Addresses

		Ports							
Board	**Status Port**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	188	130	138	140	148	150	158	160	168
1	288	1B0	1B8	1C0	1C8	1D0	1D8	1E0	1E8
2	208	230	238	240	248	250	258	260	268
3	308	2B0	2B8	2C0	2C8	2D0	2D8	2E0	2E8

Note

The PC/16 board contains no address switches. PAL chips are used instead to set the addresses. When ordering the PC/16 board, you must specify a board number of 0 or 1 and that you want the PICK OS PAL chips. These are available from Digiboard. Consult your PC/16 hardware reference manual for more information.

See also: Terminal Support Code and terminal devices, *Driver Programming Concepts*;
The PC/X Configuration Manual

ramdrv

Creates a RAM disk in memory, optionally loading a data image file into the RAM disk.

Syntax

```
sysload /rmx386/drivers/ramdrv (size [, load_file])
```

Parameters

`size` A decimal number for the RAM disk size in Kbytes. The minimum size is 64 Kbytes and the maximum is 9999 Kbytes, subject to free-space memory available in the system.

`load_file`
The full pathname of a file containing a previously-created data image to load into the RAM disk. The size of the data image file must exactly match the size of the RAM disk or the load operation will fail. To create the data image file, follow these steps:

1. Load the RAM disk driver and specify the disk size for the image you will create.
2. Attach the RAM disk and format it with the desired number of files.
3. Create a directory structure on the drive and copy the desired programs and data files onto the drive.
4. Detach the RAM disk and reattach it with the `physical` option.
5. Copy the physical RAM disk image into a file on a hard disk.
6. Detach the RAM disk, then reload the driver, specifying the same disk size and the complete pathname of the data image file.

DUIB Name

ram0

Additional Information

To load a 1024 Kbyte *ramdrv* disk and make it available with logical name *:r:*, enter:

```
sysload /rmx386/drivers/ramdrv (1024)
attachdevice ram0 as r
```

Since no data image file is specified in the **sysload** command, a warning indicates that the RAM disk is not a named volume. To make the RAM disk a named volume, enter:

```
format :r:disk_name files=num
```

Where:

`disk_name` An optional 6-character name given to the device

`files=num` An optional maximum number of files that can be created on the drive; the default is 200 if no file information is specified.

You can load multiple instances of the driver, each with a separate logical name. This only works if you perform the following steps while loading each instance:

1. Use **sysload** to load the *ramdrv* driver.
2. Use the **attachdevice** command to create a logical device name: for example, *:r0:*.
3. Format the disk.
4. Access the newly-formatted device: for example, copy files to it or use the **attachfile** command to create a second logical name.
5. Begin again at step 1, assigning a different logical name in step 2 than the first time: for example, *:r1:*.

Access the first disk with logical name *:r0:* and the second disk with logical name *:r1:* If you do not perform step D before reloading and reattaching the driver, you cannot access the *:r0:* drive.

In ICU-configurable systems, you can configure the RAM disk in the ICU (with the RAM parameter on the IDEVS screen) or load it with this driver. *Ramdrv* loads where memory is available, while the ICU version requires a pre-assigned address (base address is the BMA option on the URAM screen, size is the DSZ option on the IRAM screen).

Note

You can unload this job after it is loaded by invoking the **sysload** command with the `-u` switch.

rbootsrv.job

The remote boot server, which allows you to boot diskless workstations across the network. The board that runs the boot server must also run an iNA 960 job (*i*.job*). It need not run iRMX-NET.

Syntax

```
sysload /rmx386/jobs/rbootsrv.job SUBSYS_ID=iiH [DEBUG=x]
      [MAX_BOOT=y] [CC_FILE_SIZE=z] [MAX_DELAY=w]
```

Parameters

> **Note**
> You must use uppercase letters as shown for parameter names.

`ii` The Data Link subsystem ID for this system, as follows:

Value	Data Link for this hardware:
20H	Boards with 82586 component, including first MIX560 board in the system
21H	SBX 586 board, EWENET module, or EtherExpress 16
22H	Second MIX560 board in the system
23H	Third MIX560 board in the system
24H	82595TX component, EtherExpress PRO/10, SBC P5090 PC-compatible board
2FH	Multibus II subnet

`x` The level of debug information to display onscreen (or wherever you direct the standard output for this job with the -o option of the **sysload** command). Specify one of the following levels; the default is 0. The levels are cumulative. If you specify level 4, the messages from levels 1, 2, and 3 are also displayed and may be interspersed with one another as they occur. Use one of these values:

0 No debug display

1 Display node names as they are deleted from internal tables. This can mean the client successfully booted or it may mean the client did not finish booting because the client requests were not received before the timeout expiration.

2 Display also a message when each client's first boot request is received. This message includes:

- Ethernet address of the client
- Class code of the request
- Which number client this is (Numboot=n)
- Index into an internal boot table that stores client information; typically the index information will not be useful to you

3 Display also each filename as it is received from the client.

4 Display also a message as each packet is received from the client request.

y Maximum number of clients that can boot simultaneously. The default is 10. Note that large numbers of simultaneous boots can also be limited by other factors, such as network traffic.

z Maximum size of the *ccinfo* file in bytes; the default is 1024.

w Maximum amount of time to wait between receiving request packets from the client, in seconds. The default is 5. If a client request is delayed longer than this period, the next request is not honored, because the server has deleted this client from its internal boot tables.

Additional Information

In previous releases of the OS, the remote boot server was built into the iNA 960 NMF (network management facility) software. In the current release it is available only as a separately loadable job. The remote boot server is not available as a first-level job configurable in the ICU.

The boot server network multitask address is 01 AA 00 FF FF FF.

See also: Remote Booting, *Network User's Guide and Reference* for details about remote booting, such as client requests and class codes

remotefd.job

The loadable version of the iRMX-NET client. It consists of the iRMX-NET consumer and remote file driver. Load this job to provide transparent file access to systems that run the iRMX-NET server. Before loading this job you must load the appropriate *i*.job* for your system.

Syntax

```
sysload -w /rmx386/jobs/remotefd
```

Additional Information

The iRMX-NET client and server jobs previously contained iNA 960 software. They are now separate jobs. You must load iNA 960 (*i*.job*) separately or configure it in the ICU, then load or use the ICU to add iRMX-NET jobs.

See also: *rnetserv.job* for the loadable iRMX-NET server,
ICU-configurable versions of iRMX-NET jobs, Chapter 6,
Network User's Guide and Reference about using iRMX-NET,
Writing loadable file drivers, *Driver Programming Concepts*

Note

If you use a loadable version of an iNA 960 job, you must use the loadable versions of the iRMX-NET Client and Server jobs. With iNA 960 configured into the OS as a first-level job, you can use either the loadable or first-level versions of iRMX-NET jobs.

rintmjob.job

The Remote Intime personality job.provides the INtime API, DSM, and NTX Loader jobs that, when loaded, turn an iRMX III.2.3 system into a Remote INtime Client. It requires that the paging and flat jobs as well as the C Library jobs be loaded in the system.

Syntax

```
sysload /rmx386/jobs/rintmjob.job
```

rip.job

The raw ip service provides a direct interface to lower-level IP. It can be used to implement a new protocol above IP..

Syntax

```
sysload /rmx386/jobs/rip.job
```

Additional Information

The **ping** command uses the raw interface. Rip.job only receives packets for the protocol specified.

The IP header and any IP options are left intact by raw on receipt of datagrams.

rnetserv.job

The loadable version of iRMX-NET server, which provides transparent file access to remote systems that run the iRMX-NET client job. Before loading this job you must load the appropriate *i*.job* for your system.

Syntax

```
sysload -w /rmx386/jobs/rnetsrv.job
```

Additional Information

The iRMX-NET client and server jobs previously contained iNA 960 software. They are now separate jobs. You must load iNA 960 (*i*.job*) separately or configure it in the ICU, then load or use the ICU to add iRMX-NET jobs.

This job catalogs the names RNETSRV and NSDONE in the Name Server object table during its initialization. Remote boot (RSD) clients look for these cataloged objects so they can coordinate their boot requests with the file server initialization. RNETSRV is cataloged at the beginning of initialization; it lets the clients know that there is a file server. NSDONE is cataloged after the file server has finished adding all the entries from the */net/data* file to the Name Server object table.

Boot clients need to be able to get */net/data* information about the file server from the Name Server. The RSD client waits for server initialization to complete (indicated by NSDONE) only if it finds RNETSRV cataloged.

See also: *remotefd.job* for the loadable iRMX-NET client,
ICU-configurable versions of iRMX-NET jobs, Chapter 6,
Network User's Guide and Reference for information about iRMX-NET and the RNETSRV and NSDONE objects

Note

If you use a loadable version of an iNA 960 job, you must use the loadable versions of the iRMX-NET Client and Server jobs. With iNA 960 configured into the OS as a first-level job, you can use either the loadable or first-level versions of iRMX-NET jobs.

rtcimcom.job

The Serial Comm Channel Interface Module job provides an interface between the NTX proxy job and the ports-based Serial Driver (serdrvr.job).

Syntax

```
sysload /rmx386/jobs/rtcimcom.job <Comm_Channel> -m <Baud_Rate>
```

Parameters

`Comm_Channel`

The Serial Channel being used to provide the NTX interface to a Windows NT Host. Currently, only COM1 and COM2 in a PC Architecture system are supported.

`Baud_Rate`

The Baud Rate of the Serial Channel being used to provide the NTX interface to a Windows NT Host. This must match the value configured on the Windows NT Host using the iRMX III NTX Link Setup utility

Additional Information

See also:Preparing the NT Host System, Chapter 8

rtcimudp.job

The UDP Channel Interface Module job provides an interface between the NTX proxy job and the new TCP/IP stack.

Syntax

```
sysload /rmx386/jobs/rtcimudp.job <Host_IP_address>
```

Parameters

`Host_IP_Address`

The IP Address of the NT Host that will be using NTX commands to interface with this iRMX system.

Additional Information

See also:Preparing the NT Host System, Chapter 8

sdb.job

The system debugger (SDB) job.

Syntax

```
sysload /rmx386/jobs/sdb.job
```

Additional Information

The SDB is a debugging tool for iRMX applications and system programs. It can display information about OS objects and can interpret iRMX calls and stacks. If you use the Soft-Scope debugger, load *sdb.job* to make SDB commands available from within Soft-Scope.

See also: *System Debugger Reference*

serdrvr.job

The ports-based Serial Driver for PC Architecture system comm channels COM1 and COM2.

Syntax

```
sysload /rmx386/jobs/serdrvr.job <Comm_Channel>
```

Parameters

`Comm_Channel`

The serial channel to be controlled by the driver

Additional Information

ssk.job

The kernel part of the Soft-Scope debugger. You must load *ssk.job* before you can invoke the command line version of Soft-Scope.

Syntax

```
sysload /util386/ssk.job
```

Additional Information

The *ssk.job* job is installed in the */util386* directory with other Soft-Scope files, not in the */rmx386/jobs* directory. Instead of using the **sysload** command, you could start *sskernel* as a background job. However, the job would be removed when you logged off the system. Running the kernel as a loadable job makes Soft-Scope available for all users on the system.

You can configure the Soft-Scope kernel as a first-level job with the ICU instead of loading it.

Note
To use *ssk.job*, you must also load *clib.job*. Otherwise, an E_NOT_CONFIGURED error is reported in the *:utils:ssk.log* file.

See also: Chapter 8 in this manual and
Soft-Scope Local and Remote, *Soft-Scope Debugger User's Guide*

tccdrv

Device driver that supports up to eight serial channels in a Multibus I system using one of these serial controller boards: SBC 188/48, 188/56, 546, 547, 548, or 549.

Syntax

```
sysload /rmx386/drivers/tccdrv (base_addr, io_addr, encoded_int
                                [, mega_page])
```

Parameters

`base_addr`

The base address (in hex, no H suffix) of the TCC board's dual port RAM, as seen from the host CPU.

`io_addr`

The I/O address (in hex, no H suffix) of the TCC board, as seen from the host CPU.

`encoded_int`

An encoded interrupt level used by the TCC board to signal the host CPU when a serial channel needs servicing. This is a byte value (in hex, no H suffix) encoded as shown below.

Bits	Value
7	Reserved bit; set to 0
6-4	First digit of the interrupt level (values 0-7)
3	If one, the interrupt is on the master PIC and bits 6-4 specify the entire number If 0, the level is on a slave PIC and bits 3-0 specify the second digit of the interrupt level
3-0	Second digit of the interrupt level (values 0-7), if bit 3 is 0

For example, if the board is set to interrupt 5 and connected to the master PIC on the CPU host, the `encoded_int` value would be 58, where 5 is the interrupt level and 8 indicates the master PIC.

`mega_page`

This parameter is used only for DOSRMX or iRMX for PCs, when the host CPU is an SBC 386SX or SBC PCP4 board. The parameter is the Mbyte page number (0-0FH) where the TCC board's dual port memory actually resides. For example, if the TCC board's dual port memory is at physical address 0F90000H, specify `F` when loading the driver.

DUIB Names

t548_0 t548_1 t548_2 t548_3 t548_4 t548_5 t548_6 t548_7

Additional Information

Under DOSRMX and iRMX for PCs, this device driver supports an SBC 386SX or SBC PCP4 host CPU board, with one of the supported terminal controller modules. Use the examples below to set up the boards to work with this driver.

SBC PCP4 Host

To use this driver with an SBC PCP4 board running DOSRMX or iRMX for PCs, you must configure the board to operate like the SBC 386SX for its dual-port memory. The following example sets up the SBC PCP4 board to use the same addresses as listed above for the SBC 386SX. The 548 TCC board's dual port memory for this example is at F90000H, its I/O address is at 8A6H, and its default Multibus I interrupt is MINT 3.

Note
If you run the iRMX III OS on an SBC PCP4 board, you must still set up the dual port memory as needed by the TCC. However, with that OS you do not use the `mega_page` parameter when you load this driver.

Most configuration on the PCP4 board is done in BIOS setup routines. However, you must set one jumper on the board for this configuration. Set the interrupt for MINT 3 to IRQ 15, by placing a jumper on pins E46-E36.

Then to configure the BIOS setup routines, follow these steps:

1. Power up the system and when the system first prints to the VGA screen, press the <F1> key to enter BIOS setup.
2. In setup go to the Advanced Screen, then to the Multibus Memory and I/O Configuration Screen, and set:

Protected-Mode Memory Base = D9
Limit = DA
Offset = F9

3. In the same screen go to the I/O window and set:

I/O window base = 80
Limit = FB
Offset = 0

Note

This maps the host I/O range 8000H - 0FBFFH to match the Multibus range 0H - 7BFFH on the Multibus bus. This avoids using the I/O addresses from 0FC00H - 0FFFFH, which can be used by SCSI-2 and PCI devices on this board.

4. Press <ESC> to get you out of the screen you are in and go to the Interrupt Configuration Screen. Move to IRQ 15 on this screen and set it to "Used by ISA/MB".

5. Press the <F10> key to exit the BIOS setup and boot the system.

Now the CPU board will see the TCC board's memory at location 0D90000H and the I/O address jumpered on the TCC board (for example 8A6H) as 88A6H.

The invocation for loading the driver would be:

```
sysload /rmx386/drivers/tccdrv (D90000,88a6,27,F)
```

To access the first channel of the controller with the logical name *:t:*, attach it as a physical device:

```
attachdevice t548_0 as t p
```

tcp.job

The Transmission Control Protocol (TCP) provides reliable, flow-controlled, two-way transmission of data. It is a byte-stream protocol used to support the SOCK_STREAM abstraction.

Syntax

```
sysload /rmx386/jobs/tcp.job
```

Additional Information

TCP uses the standard Internet address format augmented by a host-specific collection of port addresses. Thus, each TCP address is composed of an Internet address specifying the host and network, with a specific TCP port on the host identifying the peer entity.

telnetd.job

The telnetd.job is a server that supports the standard TELNET virtual terminal protocol.. **Syntax**

```
sysload /rmx386/jobs/telnetd.job num_pttys=<number of PTTYs
            (between 1 and 16 inclusive)>
```

Parameters

`num_pttys`

The number of pseudo terminals (PTTYs) that will be associated with the Telnet Server.

Additional Information

The Telnet Server operates by allocating a pseudo-terminal device for a client, which has the slave side of the pseudo-terminal as an iRMX terminal device (*ptty_0* to *ptty_n*). The Telnet Server manipulates the master side of the pseudo-terminal, implementing the TELNET protocol and passing characters between the remote client and an iRMX program, such as the **logon** command or the CLI.

When a TELNET session is started, the Telnet Server sends a TELNET option to the client, indicating it is willing to do remote echo of characters, to suppress go ahead, and to receive terminal type information from the remote client. If the remote client is willing, the remote terminal type is propagated in the environment of the iRMX-controlled terminal state

When the Telnet Server is loaded, it checks to see if the Psuedo TTY driver (PTTYDRV) has been loaded. If not, it loads this driver from the /rmx386/jobs directory. Once the PTTYDRV driver has been found/loaded, the Telnet Server allocates the num_pttys number of Pseudo Terminals (4 by default if num_pttys is not specified) and waits to service Telnet client connect requests.

See also: Configuring the Telnetd Server, *services* file, *TCP/IP for the iRMX Operating System*
Configuring terminals, *System Configuration and Administration*

The Telnet Server supports these TELNET options:

- binary mode
- status of options
- remote echoing
- automatic terminal type recognition

- extended options list (there are no options currently defined on the list)

The Telnet Server also supports transmission of urgent data. It does not support timing mark.

Note
The implementation of the TELNET options follow the TELNET specifications. For a detailed description of the options, refer to RFCs 856-861.

tulip.job

Driver for the DEC 21X4X PCI-based NIC. .

Syntax

```
sysload /rmx386/drivers/tulip.job ntrans=<maximum number
      of outstanding transactions>
      ncbs=<number of control buffers>
```

Parameters

ntrans

The number of transaction buffers that can be outstanding at any given time

*ncb*s

The maximum number of control buffers available to this job.

Additional Information

udp.job

The User Datagram Protocol (UDP) is a simple, unreliable datagram protocol. UDP streams are connectionless.

Syntax

```
sysload /rmx386/jobs/udp.job
```

Additional Information

UDP address formats are identical to those used by TCP; UDP provides a port identifier in addition to the normal Internet address format. Note that the UDP port space is separate from the TCP port space (that is, a UDP port may not be connected to a TCP port). If the underlying network interface supports broadcast, UDP can send broadcast packets by using a reserved broadcast address. The broadcast address is dependent on the network interface.

□□□

Configuring RMX.INI 5

In DOSRMX and iRMX for PCs only, you can improve system performance with values specified in the *rmx.ini* file. On reboot, layers of the OS load these values as configuration parameters from *rmx.ini*.

Load-time Configuration Using the rmx.ini File

Installing the OS creates the default *rmx.ini* file, with blocks of entries for various parts of the installed system. You change configuration parameter values contained in each block, to complete *load-time configuration* of the OS. For example, you can optimize file I/O for your application with *rmx.ini* settings that increase buffer size or I/O task priority. To override preconfigured values:

- The *rmx.ini* file must exist in the *:config:* directory.
- Values must be specified according to the *rmx.ini* file syntax, each within minimum and maximum limits.

See also: **loadrmx** command, **-i** switch, *Command Reference*

For example, if your DOSRMX application uses an Intel Embedded Workstation board (SBC 486SX25, 486DX33, or 486DX66), make these changes in the *rmx.ini* file:

- In the [NUC] block, set `BUS=02H`
- For iNA 960 MIP support, make these changes in the [MIPxx] block:
  - — To use an SBC 186/530, set `DN='SD'`, `LD='LOCAL'`, `FN='/NET/INA530N.32L'`, and `CBN='186/530'`
  - — To use an SBC 486/133SE COMMputer, set `LD='NOLOAD'` and `CBN='486/133SE'`
  - — To use a MIX 386/560 board, set `DN='SD'`, `LD='LOCAL'`, `FN='/NET/INA560N.32L'`, and `CBN='MIX386/560'`
  - — To use a MIX 386/020 COMMputer, set `LD='NOLOAD'` and `CBN='MIX386/020'`

— To use one of these configurations:

  MIX 386/020 COMMputer
  MIX 486/020A COMMputer
  MIX 486SX33 COMMputer
  MIX 486DX33 COMMputer
  MIX 486DX66 COMMputer with a MIX 560 module

  set `LD='NOLOAD'` in the [MIPxx] block.

  See also: MIP jobs, *Network User's Guide and Reference*

Note
The [MIPxx] blocks apply only in a Multibus II system, where there can be more than one such block and the xx specifies which instance of a particular NIC to use. In a Multibus I or PC system you use a [MIP] block in the *rmx.ini* file rather than [MIPxx] blocks.

- For iRMX-NET support, make these changes in the [RNETS] and [RNETC] blocks:

  — To use an SBC 186/530, set `CBN='186/530'` and set `CBI` to the instance of the 186/530 being used as a NIC

  — To use an SBC 486/133SE COMMputer, set `CBN='486/133SE'` and set `CBI` to the instance of the 486/133SE being used as a NIC

  — To use a MIX 386/560 board, set `CBN='MIX386/560'` and set `CBI` to the instance of the MIX 386/560 being used as a NIC

  — To use a MIX 386/020 COMMputer, set `CBN='MIX386/020'` and set `CBI` to the instance of the MIX 386/020 being used as a NIC

rmx.ini File Syntax

Look at your *rmx.ini* file. Configuration parameters are within blocks of entries:

```
[NAME]
entry=value; Comment string
```

Where:

`NAME` Case-sensitive block name. Configuration parameters follow on each new line (no blank lines). A block ends with the beginning of the next block or end of file.

`entry` Case-sensitive name (1 to 16 characters) for a configuration parameter, followed by an equal sign (=).

`value` Hexadecimal (with H suffix) value or string (with single quotes), followed by a semicolon (;).

Comments may appear after the semicolon, extending to the end of the line.

An Example rmx.ini File

Figure 5-1 shows all of the defined blocks, and default values for entries. The exact version of *rmx.ini* on your system depends on your installed configuration. The rest of this chapter describes each entry, including possible values and discussion of usage.

```
[NUC]
UML=0FFFFFFH; Upper Memory Limit
OSX=14H;      Number of User OS Extensions
RRP=8CH;      Round Robin Threshold Level - Real Time Fence
RRT=05H;      Round Robin Time Period
KTR=01H;      Kernel Tick Ratio
BUS=03H;      Bus Type, 01H = Multibus I, 02H = Multibus II, 03H = PC
DIB=010000H;  MB II message passing DMA input alignment buffer size
DOB=010000H;  MB II message passing DMA output alignment buffer size
DEH=00H;      Default system exception handler is JOB?
       (0H=false,0FFH=true)
XLLM=??H;     Low address, Exclude from memory reclamation, RPC only
XLHM=??H;     High address, Exclude from memory reclamation, RPC only
CAF=??H;      Clock Adjust Factor (signed number, range 0H - 0FFFFH)
NAR=01H;      Number of Ranges Added to Free Space Memory (1 - 5H)
LM1=????H     Low memory (base of first additional memory range)
HM1=????H     High memory (top of first additional memory range)
[DISPJ]
VIE=00H;      Enable interrupt virtualization
EPR=00H;      ESDI drive present (0H=false, 0FFH=true)
CMS=0FFH;     Switch character mode on hardware traps (0H=false,
       0FFH=true)
PMS=0FFH;     Video Page switch SHOULD SET SAME AS CMS
       (0H=false,0FFH=true)
[BIOS]
GC=03H;       Global Clock Type
ADP=81H;      Attach Device Task Priority
CON=82H;      Connection Deletion Task Priority
TTP=81H;      Timer Task Priority
NTP=83H;      Named File Driver I/O Task Priority
RTP=83H;      Remote File Driver I/O Task Priority
ETP=83H;      EDOS File Driver I/O Task Priority
DTP=83H;      DOS File Driver I/O Task Priority
ENE=0FFH;     Enable Named File Driver Extensions (0H=false, 0FFH=true)
EFLC=0FFH;    Force DOS Filenames to Lower Case (0H=false, 0FFH=true)
[EIOS]
SBS=0400H;    Stream File Driver Buffer Size
PBS=0400H;    Physical File Driver Buffer Size
NBS=0400H;    Named File Driver Buffer Size
RBS=0400H;    Remote File Driver Buffer Size
EBS=0400H;    EDOS File Driver Buffer Size
DBS=0400H;    Native DOS File Driver Buffer Size
DDS=032H;     Default I/O Job Directory Size
ADV='device_name';    Default system device name
AFD=05H;      File driver for system device
```

Figure 5-1. Example rmx.ini File

```
 [HI]
SCF='r?init';    Default system configuration file
TCF='terminals'; Default terminal configuration file
JST=1770H;       Job synchronization timeout value
[CLIB]
NEB=0002H;       Number of EIOS buffers per connection
MBS=2000H;       Malloc block size
[KEYBD]
TE=0FFH;         Enable Alt-SysRq keys (0H=false, 0FFH=true)
DOC=0FFH;        DOS owns console on initialization (0H=false, 0FFH=true)
RDA=00H;         Disable Ctrl-Alt-Del (0H=false, 0FFH=true)
MBD=00H;         Disable Ctrl-Alt-Brk (0H=false, 0FFH=true)
[PCIAD1]
LUN0ONLY=0FFH;   Scan SCSI bus for logical unit 0 only (0H=false,
0FFH=true)
SBT=03H;         System Bus (01H = Multibus I, 02H = Multibus II, 03H = PC)
BT=040H;         Board Type (PC=040H, SBC PCP4=030H,
;                   SBC P5090=08H, SBC 486DX66=07H)
SCSICONTYPE= 022H  SCSI Controller (1542/1742=022H, 6360 =023H, 7850 =024H)
HABASE=0330H;    Host Adapter Base Port Address for Adaptec 1542/1742 board
BASEADDR=00H;    Host Adapter Base Port Address for Adaptec 6360/7850 chips
DMA=05H;         DMA Channel used
CCBS=8;          Number of CCBs
STO=000FAH;      Select Time Out
BONT=4;          Bus On Time
BOFFT=4;         Bus Off Time
BXS=0;           SCSI Bus Xfer Speed
INTL=23H;        Host Adapter Interrupt Level
SIN=0;           Server Instance
RESSCSI=0H;      Reset SCSI Bus (0H=false, 0FFH=true)
SCSIID=7;        SCSI ID is programmable in the 6360/7850 chips
[UA]
CNN='rmx';       Consumer Name
CNP='1234567';   Consumer Password
[MIP];                      For PC or Multibus I
DN='SD';         Device Name
LD='LOCAL';      iNA Load
FN='pathname';   iNA File Name
CBN='boardname'; Communication Board Name
CBI=1H;          Communication Board Instance
[MIP00];                    For Multibus II
DN='SD';         Device Name
LD='LOCAL';      iNA Load
FN='pathname';   iNA File Name
CBN='boardname'; Communication Board Name
```

Figure 5-1. Example rmx.ini file (continued)

```
[RNETS];                      iRMX-NET Server
CBN='boardname'; Communication Board Name
CBI=1H;          Communication Board Instance
[RNETC];                      iRMX-NET Client
CBN='boardname'; Communication Board Name
CBI=1H;          Communication Board Instance
```


Note

Optionally, the RMX.INI file can be used to configure the new TCP/IP stack. By default, these configuration parameters are located in the file :CONFIG:tcp.ini in the same format as used here in the RMX.INI file. These configuration parameters can also be placed in the RMX.INI file so that they will be available if the new TCP/IP stack is loaded from the :CONFIG:loadinfo file. For completeness, the TCP/IP stack configuration parameters are listed here as wsell as in Chapter 8 of the *TCP/IP for the iRMX Operating System* manual.

```
[TCP];                      TCP Layer
DEFMSS=200H;          Default max segment size
RCVSPACE=4000H;       Max receive space per socket
SNDSPACE=4000H;       Max sendspace per socket
CTLBUFS=40H;          Max total control buffers
TRANSBUFS=40H;        Max total transaction buffers
MAXTRANS=10H;         Max transactions per socket
MAXPORTS=1388H;       Max port ids
LOWFIXPID=1H;         Well-known port id range
HIFIXPID=3FFH;
LOWAUTOPID=400H;      Ephemeral port id range
HIAUTOPID=1387H;
[UDP];                      UDP Layer
CHECKSUM=1H;          Checksum data?
RCVSPACE=0A000H;      Max receive space per socket
CTLBUFS=40H;          Max total control buffers
TRANSBUFS=40H;        Max total transactions buffers
MAXTRANS=10H;         Max transactions per socket
MAXPORTS=1388H;       Max port ids
LOWFIXPID=1H;         Well-known port id range
HIFIXPID=3FFH;
LOWAUTOPID=400H;      Ephemeral port id range
HIAUTOPID=1387H;
```

Figure 5-1. Example rmx.ini file (continued)

```
[RIP];                      RIP Layer
CTLBUFS=20H;          Max total control buffers
TRANSBUFS=20H;        Max total transaction buffers
MAXTRANS=8H;          Max transactions per socket
MAXPORTS=80H;         Max port ids
[IP];                       IP Layer
IFNAMES='ETH0, LO0'; Interface names
BUFHEAPSIZE=140H;     Total receive buffer size in Kbytes
FORWARDING=0H;        Enable IP forwarding
LOCALSUBNETS=1H;      Enable local subnets
TTL=8H;               Default segment time to live
TOS=0H;               Default type of service
ARPTIMEOUT=20H;       ARP cache flush timeout in minutes
CTLBUFS=80H;          Max total control buffers
TRANSBUFS=80H;        Max total transaction buffers
[ETH0];                     Interface Layer
HOST='206.103.53.119';     Interface IP address
NETMASK='255.255.255.0';   Net mask
DEFROUTE='206.103.53.250'; Default route
RCVBUFS=3FH;               Max receive buffers
MAXTRANS=6FH;              Max simultaneous transactions
[lO0];                     Loopback Interface Layer
HOST='127.0.0.1';          Interface IP address
NETMASK='255.255.0.0';     Net mask
RCVBUFS=3FH;               Max receive buffers
MAXTRANS=6FH;              Max simultaneous transactions
[DNS];                      DNS Layer
DOMAIN='radisys.com';      Local domain name
SERVER1='206.103.52.253';  Primary Domain Name server, total
                            of 3 allowed
```

Figure 5-1. Example rmx.ini file (continued)

Nucleus Block [NUC]

UML Upper Memory Limit

The iRMX OS can use all available contiguous memory beginning at physical address 110000H (1 MByte + 64 Kbytes), up through the address specified here.

Value	Meaning
0, FFFFFFFFH	DOSRMX controls all extended memory found by the ROM BIOS at boot time, up to 64 Mbytes
Any other value	Overrides the amount found by the ROM BIOS memory tests. If your system has more than 64 Mbytes of memory, you must set UML, typically to the total amount of physical memory present in the system.

Use this parameter to reserve memory in the high address range by setting UML to a value lower than the top of memory. For example, memory for an I/O device might be mapped to start at C00000H (12 Mbytes). To prevent the OS from interfering with this memory, you would set `UML=BFFFFFH`. (See also the NAR and LM1, HM1 through LM5, HM5 parameters.)

OSX OS Extensions

Specifies the number of OS Extensions available for applications. OS Extensions used by loadable jobs are not taken from this number.

The OS reserves a slot in the Global Descriptor Table (GDT) for each OS Extension specified here, beginning with GDT slot 440. This reduces the number of objects created by application jobs (each object also uses a GDT slot).

See also: OS Extension example, *Programming Techniques*; **rqe_set_os_extension**, *System Call Reference*

RRP Round-robin Priority (real-time fence)

Value	Meaning
0-0FEH	Tasks below the real-time fence (numerically above priority RRP + 1) are scheduled in round-robin fashion, rather than by highest priority; 0 works well with priorities of HI, EIOS, and BIOS tasks
0FFH	Disables round-robin scheduling

RRT Round-robin Time period

The number of iRMX clock ticks (10 ms) a task runs before being preempted by another task. Only valid for tasks whose priority is lower than the real-time fence.

KTR Kernel Tick Ratio

The ratio of the Nucleus tick interval (10 ms) to the iRMK tick interval. Possible values are 1 (default), 2, 5, 10, or greater than 10.

A higher KTR value shortens the iRMK clock tick value relative to the 10 ms iRMX clock tick as follows:

KTR Value	iRMK Tick
1	10 ms
2	5 ms
5	2 ms
10 (0AH)	1 ms
20 (14H)	500 µs

Never assume that a Nucleus tick is equivalent to an iRMK Kernel tick. You should write code that adapts to the KTR values. The iRMK clock tick interval affects the **create_alarm**, **get_time**, **receive_data**, **receive_unit**, **set_time**, and **sleep** system calls.

See also: KTR, *ICU User's Guide and Quick Reference*;
RQSYSINFO structure, Chapter 1, *System Call Reference*

`BUS` Bus type

Set this value according to the bus functions you intend to use. For example, if you have a PC-compatible board in a Multibus II system, but only use the PC functions with the DOSRMX OS, set BUS=03H. On the other hand, if you use the same board to perform Multibus II message passing, you must enable the Multibus II bus functionality by setting BUS=02H. This sets up low-level functionality such as the type of DMA transfers, etc.

Value	Meaning
01H	Multibus I
02H	Multibus II
03H	PCs

`DIB, DOB`

Multibus II message passing DMA alignment buffer size

Solicited message buffers must be aligned on 4-byte boundaries (low 2 bits of buffer address are 0). DIB is for input and DOB for output. Adjust buffer size to be at least that of the largest unaligned solicited message.

If you use a token for an iRMX buffer pool as a message buffer, it is automatically aligned. The Free Space Manager creates all segments on 16-byte boundaries (low 4 bits of the address are 0).

An attempt to send or receive a message in an unaligned buffer larger than the configured alignment buffer size results in an E_NUC_BAD_BUF exception.

`DEH` Default Exception Handler

Specifies one of two default system exception handlers, which also act as hardware trap handlers:

Value	Meaning
00H	System Debugger and System Debug Monitor (SDB/SDM)
0FFH	Nucleus job deletion handler; only recommended for fully debugged system because it disables the Soft-Scope debugger

`XLLM, XLHM`

In the iRMX for PCs OS only, these parameters work together to exclude a range of memory from being reclaimed by the OS. . The normal operation of this OS is to reclaim lower memory during initialization for use as OS system memory. The beginning of this memory to be reclaimed can vary; it is determined by the address of communications buffers set up during the boot process, but typically will be about 164K. The end of the memory to be reclaimed is 640K. On certain boards with specific I/O restrictions in this area, you may need to exclude a range of memory from being reclaimed. For example, on an SBC PCP4 board, you may need to set up dual-port memory with Multibus in this range.

To exclude memory in this range from being reclaimed, set XLLM to the low physical address and XLHM to the high address for the range you want to exclude. If you set the exclusion range within 16K of the boundary that would normally be reclaimed, the memory is excluded all the way to that boundary. For example, if you set XLHM to 625K, memory up to the 640K boundary is not reclaimed.

`CAF` Clock Adjust Factor

This parameter lets you compensate for inconsistencies in particular time-of-day clocks. Use this only on a particular machine whose clock consistently gains or loses time. This factor adjusts the Nucleus time (clock ticks) relative to the time-of-day clock. It is a signed number, so you can set it to a positive or negative amount (range 0h to 0FFFFH). Determine the correct value for your machine empirically. For example, if your machine loses 10 seconds in a day, set CAF to a small positive number, then set the OS time. Exactly 24 hours later, check the OS time and adjust CAF accordingly.

`NAR` Number of Ranges

This parameter lets you specify up to 5 ranges of memory to be included as Free Space Memory, and managed by the Free Space Manager. Define the memory ranges with parameters LM1, HM1 through LM5, HM5.

`LM1, HM1` through `LM5, HM5`

Each pair of LM*x*, HM*x* parameters specifies a memory range to be included as Free Space, managed by the OS. Set LM*x* to the beginning address of the range and HM*x* to the end (top) address of the memory range. Use as many pairs of these parameters as you specify in the NAR parameter, up to 5. If NAR = 2, you would set an LM1, HM1 pair and an LM2, HM2 pair.

When you use these parameters, you must set UML to 0. Free Space memory is then defined by the LM/HM pair blocks, none of which should overlap:

- The LM1, HM1 parameters define the first memory block.
- The LM2, HM2 parameters define the next higher block; then LM3, HM3, etc.
- The LM5, HM5 parameters, if used, define the highest block.

Defining Free Space memory in this way allows you to block out ranges in physical memory that are not part of Free Space memory and therefore not managed by the OS. You might need such ranges for use by device drivers, for example, to perform I/O in an area of physical memory.

Dispatcher Job Block [DISPJ]

`VIE` Virtualize Interrupts Enabled

Specifies the amount of CPU control that the iRMX OS gives to DOS.

Value	Meaning
00H (false)	Interrupt virtualization disabled. The iRMX OS traps all DOS access of the system timer and interrupt controller. DOS can use all other real-mode instructions supported by the CPU, including interrupts. This degrades interrupt latency and real-time performance, but DOS disk and ROM BIOS I/O perform near their native levels. The bounds of interrupt latency depend on the ROM BIOS and vary on different platforms. This setting is best for applications that use the ROM BIOS for performance-critical disk I/O without needing tight interrupt latency.
0FFH (true)	Virtualizes interrupts seen by DOS, while disabling CPU interrupts for a very short time. Virtualized interrupts have the optimum interrupt response time and latency, but DOS disk and ROM BIOS I/O performance is degraded (transfers of less than 4 Kbytes are slowed dramatically). With this setting, the iRMX OS traps all DOS access of privileged instructions (e.g., CLI, STI, INT, POPF) and accesses to the system timer and interrupt controller. This setting is best for applications that do not rely on disk I/O performance (e.g., Nucleus level applications) or that use a native iRMX disk driver (e.g., Adaptec SCSI).

`EPR` ESDI drive Present

Value	Meaning
0H	No ESDI drive (if there is an ESDI drive, DOS interrupts will be missed due to a race condition in the ESDI ROM BIOS)
0FFH	DOS disk drive is ESDI; only recognized if VIE=0FFH (this setting can degrade interrupt latency in iRMX for PCs, defeating the purpose of setting VIE=0FFH)

If you are using an LP486/33E Professional Workstation with an IDE drive and VIE=0FFH, you must also set EPR=0FFH.

`CMS` Character Mode Switch

Value	Meaning
0H	Do not switch console display to text mode upon entry to SDM.
0FFH	Switch console display to text mode upon entry to SDM, allowing full control of the system.

PMS Page Mode Switch
To ensure that the video page mode works correctly when you adjust CMS, always set PMS to the same value as CMS.

Basic I/O System Block [BIOS]

GC Global Clock type
Specifies the hardware time-of-day clock maintained by battery (global clock).

Value	Meaning
00H	No clock (OS clock starts at the time of the last access to *:system:* directory, unsynchronized with the global clock)
01H	SBC 546 or 549 board in a Multibus I system
02H	CSM/001 or CSM/002 clock in a Multibus II system
03H	PC system clock

ADP Attach Device task Priority
Specifies the priority of the BIOS attach_device task. The default priority of 81H is the highest priority allowed for a non-interrupt task.

See also: Writing loadable file drivers, *Driver Programming Concepts*

CON Connection deletion task priority
Specifies the priority of the BIOS task that deletes file and device connections. This can affect the performance of job deletion. Before the Nucleus can delete a job, all objects contained in the job must be deleted. Because connections are composite objects, the Nucleus cannot delete them directly. Rather, it sends them to a deletion mailbox where the deletion task waits. The default value of 82H is one greater than the default priority of the device service tasks, and should be sufficient for most applications.

TTP Timer Task Priority
Specifies the priority of the task that maintains the local (OS) time-of-day clock. If you adjust this value, change it in increments of 1.

NTP Named file driver Task Priority
RTP Remote file driver Task Priority
ETP EDOS file driver Task Priority
DTP DOS file driver Task Priority
Specifies the priority of the task associated with a device when you attach it for use with the named, remote, EDOS, or native DOS file driver.

In DOSRMX, the EDOS file driver uses the ROM BIOS, which spends time polling the device. You may want to set the priority of the EDOS file driver lower (numerically higher). Although this can degrade the performance of tasks using the EDOS file driver, it could speed up applications that do not depend on DOS file I/O.

ENE Enable Named File Driver Extensions

This parameter enables Named file driver features for compatibility with DOS:

Value	Meaning
0H	Disable extensions
0FFH	Enable named file driver extensions, including these features:

Feature	Description
Timestamps	When a file is renamed or the file permissions changed, the last access time is updated, but the last modified time is not (no actual data in the file has changed).
Rename in-place	When a file is renamed within a subdirectory, the filename is simply changed (not moved within the directory).

EFLC Force EDOS and DOS Files to Lower-Case

Value	Meaning
0H	Filenames on EDOS-managed devices are always shown in all upper-case.
0FFH	The **a_get_directory_entry** system call and iRMX **dir** command display filenames in lower-case.

Extended I/O System Block [EIOS]

SBS Stream file driver Buffer Size

PBS Physical file driver Buffer Size

NBS Named file driver Buffer Size

RBS Remote file driver Buffer Size

EBS EDOS file driver Buffer Size

DBS Native DOS file driver Buffer Size

The size (0-0FFFFH bytes) of EIOS buffers for file drivers. When a task opens a connection to a file, the EIOS creates buffers equal to the largest multiple of device granularity that does not exceed the value specified here. Larger buffers increase I/O performance, at the expense of system memory. The memory tradeoff must be considered, since buffers are allocated on a per-connection basis system-wide. Increasing the default size can provide an increase in the sequential I/O performance, but will not enhance random I/O performance. OS utilities use two buffers for each open connection. Your application can specify the number of buffers used each time a task opens a connection.

See also: **s_open**, *System Call Reference*;
EIOS buffering, *System Concepts*

DDS Default Directory Size

Specifies the maximum number of entries (05H-0F00H) in the object directories for all I/O jobs created by the EIOS.

You normally use the default value. However, I/O jobs can communicate only through the object directory of a common ancestor job, so your system might require a higher value. DOSRMX and iRMX for PCs require at least 25 entries because they include the HI layer.

ADV Automatic boot Device name
Specifies the physical name (network name if remote) for the boot device. Typically, this entry will not appear in *rmx.ini* except in remote boot applications.

AFD Automatic boot device File Driver
Specifies the file driver used by the boot device in the ADV parameter:

Value	Meaning
0	Not a valid file driver ID
1	Physical
2	Stream
3	Native DOS
4	Named
5	Remote
6	EDOS
7-16	Available for loadable file drivers, including NFS

See also: Writing loadable file drivers, *Driver Programming Concepts*

Human Interface Block [HI]

SCF System Configuration File
Specifies the filename for the system initialization file; the default is *r?init*, but you can use this parameter to specify another file. For example, if you set SCF to *initrsd*, the initialization executes commands from *:config:initrsd* instead of *r?init*, and then attempts to execute commands from *:config:initrsd2* instead of from *r?init2*.

See also: HI initialization and logon, Chapter 1

TCF Terminal Configuration File
Specifies the filename for the terminal initialization file; the default is *:config:terminals*, but you can use this parameter to specify another file.

See also: Terminal Configuration File, Chapter 2

JST Job Synchronization Timeout
When the `-w` switch is used with **sysload**, it waits until this timeout value expires or until the job catalogs *R?END_INIT* , whichever comes first. Cataloging R?END_INIT(it can have a null token) indicates the end of initialization. The default value of JST (1770) corresponds to 1 minute.

Shared C Library Block [CLIB]

NEB Number of EIOS Buffers

Specifies the number of EIOS buffers to be associated with file connections created by C library calls. Each task can also configure its own number of EIOS buffers with the **_set_info** function.

See also: CINFO_STRUCT, *C Library Reference*

MBS Malloc Block Size

The size in bytes of the iRMX memory segment used to satisfy calls to **malloc**. A segment of this size is created for each job using the C library when the first call to **malloc** is made from that job. When all the memory in a given segment has been allocated by **malloc** calls, another segment of MBS size is created (up to the job's memory limit).

Keyboard Block [KEYBD]

TE Console Toggle Enable

Value	Meaning
0H	Disables toggling console ownership between iRMX and DOS through <Alt-SysRq>
0FFH	Enables <Alt-SysRq> toggle

DOC DOS Owns Console

Value	Meaning
0H	iRMX OS owns console
0FFH	DOS owns console on boot (default)

RDA Reset Disable

Value	Meaning
0H	Enable <Ctrl-Alt-Del>
0FFH	Disable <Ctrl-Alt-Del> (system reset)

MBD Monitor Break Disable

Value	Meaning
0H	Enable <Ctrl-Alt-Break> as means to actuate the monitor
0FFH	Disable <Ctrl-Alt-Break> keys so you can not break to the monitor

Peripheral Controller Interface Adaptec Driver Block [PCIAD1]

LUN0ONLY

Logical Unit 0 Scan

This specifies the PCI server to scan for logical units other than 0 at initialization. Some SCSI devices do not operate correctly when an attempt is made to access logical units greater than 0.

Value	Meaning
0	Scan for all logical units
0FFH	Scan for logical unit 0 only

`SBT` System Bus Type
Same as the BUS parameter in the [NUC] block.

Value	Meaning
01H	Multibus I
02H	Multibus II
03H	PCs

`BT` Board Type
Set to one of the following:

Value	Meaning
07H	SBC 486SX25, 486DX33, 486DX66
08H	SBC P5090 and P5120, all versions
30H	SBC PCP4DX or SX versions
40H	PC systems

`SCSICONTYPE` SCSI Controller Type
Set to one of the following:

Value	Meaning
22H	Adaptec 1542/1742
23H	Adaptec 6360 chip (on SBC 486DXxx boards)
24H	Adaptec 7850 chip (on SBC P5090, P5120 and PCP4 boards)
25H	Symbios 53C8xx PCI SCSI adapter (also on SBCP500 board)

`HABASE` Host Adapter Base port address
Use this parameter to specify the base port address of an Adaptec 1542 or 1742 SCSI host adapter as configured by jumpers on board.

`BASEADDR` Host Adapter Base port address
Use this parameter to specify the base port address of a board that uses an Adaptec 6360 or 7850 SCSI chip. This includes the SBC 486DXxx, P5090, P5120 and PCP4 boards.

`DMA` DMA Channel
DMA channel as configured by jumpers on board.

See also: Adaptec hardware manual for address and DMA jumper information

`CCBS` Command Control Blocks
Number of CCBs for this board, from 8 to 64. The more you request, the more memory is used. There is no benefit to having more than 8 CCBs if there are not more than 8 devices on the same SCSI bus.

`STO` Select Time Out
The amount of time (1-0FFFFH ms) to wait for a device to respond before returning an error.

`BONT` Bus On Time
Specifies the time in microseconds that the host adapter spends on the bus when transferring data. For the Adaptec 6360 chip the valid range of values is 0H-0FH; for 1542/1742 boards the valid range is 02H-0FH.

`BOFFT` Bus Off Time
Specifies the time in microseconds that the host adapter spends off the bus during a data transfer. For the Adaptec 6360 chip the valid range of values is 0H-0FH; for 1542/1742 boards the valid range is 04H-040H.

`BXS` SCSI Bus Transfer Speed
These values set a transfer rate in megabytes/sec. (For the Adaptec 1542/1742 this actually sets the DMA transfer rate instead of the SCSI bus transfer rate; see the Adaptec hardware manual for values 80H-0FFH.)

Value	1542/1742	Value	6360 Chip	Value	7850 Chip
0H	5.0 MB/sec	0H or 19H	10 MB/sec	0H	10 MB/sec
1H	6.7 MB/sec	25H	6.67 MB/sec	08H	8 MB/sec
2H	8.0 MB/sec	32H	5.0 MB/sec	10H	6.7 MB/sec
3H	10 MB/sec	3EH	4.0 MB/sec	18H	5.7 MB/sec
4H	5.7 MB/sec	4BH	3.33 MB/sec	20H	5.0 MB/sec
80-FF	see manual	57H	2.86 MB/sec	28H	4.4 MB/sec
		64H	2.5 MB/sec	30H	4.0 MB/sec
		70H	2.22 MB/sec	38H	3.6 MB/sec

`INTL` Host Adapter Interrupt Level
The iRMX encoded interrupt level; on an adapter where you set a DOS interrupt level you must correlate between the iRMX and DOS interrupt values. On the SBC P5090 boards this parameter is ignored and the value read from CMOS memory.

See also: Table 4-3, *DOS Interrupt Requests and iRMX Encoded Interrupts*, Chapter 4

`SIN` Server Instance
The instance of the PCI Server running for this board; 0 specifies the first instance and 1 specifies the second.

`RESSCSI` Reset SCSI Bus

Value	Meaning
0H	Don't reset the SCSI bus at initialization
0FFH	Reset the bus at initialization

`SCSIID` SCSI ID
You can set the SCSI ID for the device on boards that use the Adaptec 7850 chip.

Network User Administration Block [UA]

`CNN` Client Node Name
Specifies the client (consumer) name of the system in a string of 3 to 8 characters.

`CNP` Client Node Password
Dynamic logon password for the client system in the CNN parameter.

> **Note**
> The password you specify here is the real password (not encrypted).

MIP Block [MIP]

`DN` Device Name
Specifies the device where the iNA 960 load file resides. Only valid if `LD` is set to `'LOCAL'`.

`LD` iNA Load
Specifies how the NIC is loaded with iNA 960 transport software. Use one of these values:

Value	Meaning
'LOCAL'	Download a file from a local hard disk.
'NOLOAD'	Do not load iNA software from this CPU board.

`FN` iNA File Name
Pathname of the iNA 960 load file. This parameter is only valid if `LD` is set to `'LOCAL'`.

See also: *i*.job*, Chapter 4

`CBN` Communication Board Name
Specifies the NIC for a Multibus II system. The name must match the one encoded in interconnect space on the board.

`CBI` Communication Board Instance
Specifies the instance of the NIC in Multibus II systems. This is not the slot in which the board resides. If there is more than one board of this type, this specifies which board to load the iNA 960 software onto. 1 specifies the first such board found in the system, counting up from slot 0, up to 14H.

See also: MIP jobs, *Network User's Guide and Reference*

[MIPxx] Blocks for Multibus II

The *icemb2.job* MIP job ignores the [MIP] block in the *rmx.ini* file. Instead, it uses one or more [MIP*xx*] blocks, where *xx* is a decimal number ranging from 00 to 19. Each block must be in sequence in the file, beginning with [MIP00]. The purpose of

these blocks is to provide access from a single CPU board to multiple NICs. The number *xx* specifies the instance of the NIC, counting up from slot 0. For example, the block [MIP00] refers to the first network controller, found in the lowest slot number of the system. Block [MIP01] refers to the second instance, etc.

In one block you could configure downloading of an iNA 960 load file to a separate NIC. In another block you could set `LD='NOLOAD'` to use a CPU board running an iNA 960 COMMputer job as the NIC.

Each [MIP*xx*] block can have the same parameters as the [MIP] block listed above, with the exception of CBI.

iRMX-NET Server and Client Blocks [RNETS] and [RNETC]

`CBN` Communication Board Name
If you use a COMMengine environment (running a MIP job on the local board), this parameter specifies the board name of the NIC from which the iRMX-NET Server and Client jobs take their services. The name must match the one encoded in interconnect space on the board.

`CBI` Communication Board Instance
If you use a COMMengine environment (running a MIP job on the local board), this parameter specifies the instance of the board type named in the CBN parameter. This is not the slot in which the board resides. If there is more than one board with this name, specifying 1 for CBI indicates the first such board found in the system, counting up from slot 0; 2 indicates the second such board, etc.

TCP/IP Stack Configuration

Please refer to Chapter 8 of the *TCP/IP for the iRMX Operating System* manual for detailed information on configuring the optional TCP/IP stack sections of the RMX.INI file.

❑❑❑

Configuring System Jobs Using the ICU 6

This chapter describes system jobs that you can configure into the OS using the ICU. These first-level jobs run at the same level in the job tree as OS jobs. Use the ICU screens listed below to configure these system jobs:

Job	ICU Screen(s)	Description
ATCS	SYSJ, ATCJ, ATC50	The Asynchronous Terminal Controller Server (ATCS) manages Multibus II terminal controller boards. To use the ATCS you must start this job for each terminal controller board, then an ATCS device driver on all hosts that use the board through the job.
Bootserver	SYSJ, BSJ	The Multibus II Systems Architecture (MSA) Bootserver job receives and services requests from boot clients that want to boot dependently. It is an implementation of the MSA bootserver protocol.
C Library	SUB, CLIB	Version of the loadable *clib.job* that runs as a layer of the OS when configured by the ICU.
Downloader	SYSJ, DLJ	The Multibus II downloader job loads object files for the OS and application programs onto boards in a Multibus II system. This job is available both as an ICU-configurable job on the DLJ screen and as the **dload** command.
FPI server	SYSJ, FPIJ	The Front Panel Interrupt (FPI) server job sends an interrupt to the Multibus II host board on which the CSM/002 clock module and front panel switch is mounted. Other hosts on the system bus do not know an interrupt has occurred.
iNA 960	NET, ICMPJ, MIPJ, NS, NSDOM	The iNA 960 network jobs provide programmatic access to OSI network protocols. You must load an iNA 960 job to run iRMX-NET or TCP/IP. See also: *i*.job*, Chapter 4

Job	ICU Screen(s)	Description
iRMX-NET	NET, RCJ, FC, REM, RSJ, FS	The iRMX-NET client and server jobs provide transparent file access on top of an iNA 960 job.
Paging subsystem	SUB, PIMM, APPL	The paging subsystem manages memory with the processor in paging mode, to support flat model applications.
PCI server	SYSJ, PCIJ, PCISC	The Peripheral Controller Interface (PCI) server manages SCSI-based peripheral devices. On each host that uses a SCSI device, configure a PCI driver to communicate with the PCI server, using the DPCI screen.
Soft-Scope debugger	SYSJ, SSKJ	Version of the loadable *sskernel* job that runs as a layer of the OS when configured by the ICU.

On a Multibus II board running the iRMX III OS, you can override some ICU configuration parameters with entries in a Bootstrap Parameter String (BPS) file. This enables you to make minor changes and reboot the system, rather than generating a new boot image.

See also: *ICU User's Guide and Quick Reference* for ICU screen details; BPS parameters, *MSA for the iRMX Operating System*

ATCS Job

ATCS jobs are specific to the type of terminal controller being managed. There are three ways to start an ATCS job, depending on the terminal controller:

- Configure an ATCS job in the ICU
- Load an ATCS job with a **sysload** command
- Download the job with the Multibus II downloader (either the ICU-configured version or the **dload** command)

See also: **dload**, Appendix B

Choose the ATCS job and configuration method from this list:

Controller Board	Server Name	Method
186/410	*atcs.410* file	Downloader
186/450	*atcs.450* file	Downloader
MPI 450	*atcs.450* file or	Downloader
	ATCS/450 Server	ICU-configured job
MIX 450	ATCS/450 Server	ICU-configured job
SBX 279 graphics	ATCS/279/ARC Server	ICU-configured or loadable job

ATCS/279/ARC Job

The ATCS/279/ARC job typically manages the system console. It operates in one of two ways:

- If there is an SBX 279 graphics board on the host where the server is configured, the server manages this graphics terminal, with multiple windows available for other CPUs on the bus.
- If there is not an SBX 279 board, the server multiplexes a character-based terminal on this host between CPUs on the bus; you switch CPUs with a hot key.

See also: *atcs279.job*, Chapter 4

Note
If the ATCS/ARC Server is configured for the `t82530_0` device (no SBX 279), tasks at priority 255 do not execute. The workaround is to make sure that all application tasks execute at priority 254 or higher (numerically lower).

ATCS/450 Job

The MPI 450 board is a non-intelligent controller board. It has identical hardware to the 186/450 board.

For MIX 450 modules, the ATCS/450 server job must be configured on the MIX baseboard containing the 450 modules. If there are not three 450 modules on the baseboard, the ATCS job can also manage MPI 450 boards (to a maximum of 36 serial ports) located in higher slot numbers on the bus.

See also: ATCS/279/ARC/450 system jobs, Appendix C

Configuring ATCS Drivers

The configuration of the ATCS driver is identical in most of the standard Multibus II ICU definition files. When a definition file includes both the ATCS driver and the SBX 279A driver, as is the case for the SBC 386/258 and 486/133SE boards, use the 279A driver DUIBs. The ATCS driver does not provide them.

See also: ATCS/279/ARC job, Appendix C for 279 DUIB names.

Table 6-1 summarizes the ATCS configuration parameters for the first instance of each board; you may configure more instances. You can specify the configuration in the ICU or use System Parameter Strings (SPS) in the MSA Bootstrap Parameter String (BPS) file. SPS parameters supersede values set with the ICU.

See also: SPS parameters, *MSA for the iRMX Operating System*

Table 6-1. Standard ATCS Device Driver Configurations

Board ID	DUIB Names	Line/ Unit	DEV Parameter	SPS Parameter
186/410	T_ATCS_A0 ... T_ATCS_A11	0 ... 11	ATCS_A	RQ_ATCS_A
186/450	T_ATCS_B0 ... T_ATCS_B11	0 ... 11	ATCS_B	RQ_ATCS_B
MIX baseboard, 450 modules	T_ATCS_C0 ... T_ATCS_C35	0 ... 35	ATCS_C	RQ_ATCS_C
486/125 *	T_ATCS_D0 ... T_ATCS_D35	0 ... 35	ATCS_D	RQ_ATCS_D
386/258 **	ATCS_CON_0 ... ATCS_CON_4	0 ... 4	ATCS_CON	RQ_ATCS_CON
	T279_0 ... T279_4	0 ... 4	ATCS_CON	RQ_ATCS_CON

* and all other CPU boards

** and all other I/O Server boards

Note

If the ATCS driver is configured to strip the parity bit on input characters, special characters (07FH and above) are not processed properly if they also happen to be signal characters.

Example Configurations

These examples illustrate ways you can change the configuration to use multiple boards of one type. You can use the BPS file or ICU. In the BPS file, the board ID is the `bnam` option and the board instance is the `bin` option. In the ICU these are the BID and IN options.

SBC 186/410

Six serial channels, first one is `t_atcs_a0`. If adding a second board, change the board ID of the second set (currently assigned to the 186/450) to 186/410 and the board instance to 2. In the BPS file, modify the SPS parameter `rq_atcs_b`. You would then use the first six of both the `t_atcs_a?` and `t_atcs_b?` physical device names.

Use the ICU to create DINFO and UINFO tables, and DUIBs for each additional board, assigning unique physical device names.

SBC 186/450

Twelve serial channels, first one is `t_atcs_b0`. If adding a second board, change the board ID of the first set (currently assigned to the 186/410) to 186/450 and the board instance to 2. In the BPS file, modify the SPS parameter `rq_atcs_a`. You would then use both the `t_atcs_a?` and `t_atcs_b?` physical device names.

Use the ICU to create DINFO tables, UINFO tables, and DUIBs for each additional board, assigning unique physical device names.

MIX x86/020

Baseboard with one to three MIX 450 modules mounted on it, for up to a total of 36 serial channels, first one is `t_atcs_c0`. If adding a second board, change the board ID of the fourth set (currently assigned to a CPU board) to the board ID for the MIX baseboard and change the board instance to 2. In the BPS file, modify the SPS parameter `rq_atcs_d`. You would then use both the `t_atcs_c?` and `t_atcs_d?` physical device names.

Use the ICU to create DINFO tables, UINFO tables, and DUIBs for each additional board. Assigning the same MIX board ID, but unique instances and physical device names.

CPU board (SBC 486/125 and ATCS/450 job)

A CPU board can control one to three MPI 450 boards for up to 36 channels, first one is `t_atcs_d0`. Each can control no more than three MPI 450 boards. If you use multiple CPU boards running this server, the board ID and board instance parameters must be unique in each iRMX boot file. In the BPS file, modify the SPS parameter `rq_atcs_d`.

I/O board (SBC 386/258 and ATCS/279/ARC job)

The ATCS/279/ARC server job can provide terminal I/O for not only the I/O Server board, but for all other CPU boards in the system. This job manages I/O either in graphic windows (on the SBX 279A board using the 279 portion of the server) or on a serial terminal (using the ARC portion of the server). There are 5 device names that support windows on the SBX 279A board. The first of these device names is `t279_0`. When using a serial terminal, the ARC server supports one connection per CPU board, named `atcs_con_0` on each board. The DUIB names `atcs_con_1` through `atcs_con_4` are not available with the ARC server.

Use the ICU to create DINFO tables, UINFO tables, and DUIBs for each additional board. Assign the same I/O Server board ID, but unique instances and physical device names. You can modify the board ID and board instance parameters in the BPS file with the SPS parameter `rq_atcs_con`.

See also: Multibus II standard definition files, *ICU User's Guide and Quick Reference*, to see which default configurations include the ATCS/450 and ATCS/279/ARC servers

Bootserver Job

The Bootserver is an I/O job created during OS initialization. When a host requests bootstrapping, the Bootserver reads the dependent second stage from disk and sends it to that boot client. The dependent second stage runs on the boot client and issues a request to the Bootserver to load the target file containing the OS. The Bootserver remains an active job in the system as long as the system runs. You can access the Bootserver through the Nucleus message-passing system calls; bootstrap requests use these calls. The Bootserver uses EIOS calls to access configuration files, bootloader second stage files, and OS target files.

See also: *MSA for the iRMX Operating System* and
Multibus II System Architecture Bootstrap Specification

The Bootserver does not attach devices; therefore, pathnames passed to it must contain a logical device name. The device referenced by the logical name must already be logically attached. You can attach the device with an EIOS **logical_attach_device** system call or, if the Bootserver is running on a host that includes the HI, you can invoke the **attachdevice** command at the command line. Do this attachment before any boot client attempts to reference that logical device. If you specify a null logical device name, the system uses the default logical device specified in the ICU.

The Bootserver conforms to the Local Boot Service compliance level of the MSA architecture. It provides the message requests shown in Table 6-2.

Table 6-2. Bootserver Functions

Bootserver Function	Request Value	Response Value
Locate Bootserver	01H	8001H
Locate Config Server	02H	8002H
Connect Bootserver	03H	8003H
Open Second Stage	04H	8004H
Get Host BPS	05H	8005H
Open File	06H	8006H
Read File	07H	8007H
Seek File	08H	8008H
Close File	09H	8009H
Disconnect Bootserver	0AH	800AH

Required Bootstrap Parameter String (BPS) Parameters

Unlike the loadable *bootserv.job* (which uses a default BPS filename or a BPS filename in the **sysload** invocation), the linkable Bootserver job requires a `BL_config_file` parameter in the Bootserver's host BPS Save Area. The Bootserver uses this value as the pathname to locate the BPS file. A fatal error occurs and the Bootserver job is deleted if this parameter is not set when the Bootserver initializes.

The `BL_config_file` parameter is not normally defined in the BPS file. However, for a host that boots independently or quasi-independently, where this parameter is not already defined, the second stage bootstrap loader sets `BL_config_file` to a default value: */msa/config/bps*. For such a host, if */msa/config/bps* is the proper pathname, you need not take any special action.

In systems where a board that boots dependently hosts the Bootserver, define the `BL_config_file` parameter in the BPS file that controls the board's dependent boot. Although it may seem like the parameter would not be found when defined in a BPS file (since it is used to point to a BPS file), the parameter is not used to boot the dependent host. The parameter is used by the Bootserver after the host has booted.

See also: BPS parameters, *MSA for the iRMX Operating System*

Initialization Errors

If you enable reporting of initialization errors on the ICU Nucleus screen (RIE parameter on the NUC screen), errors that occur while the Bootserver is initializing will be displayed on the system console. Such errors have the form:

```
MSA ERROR: xxxxH WHILE INITIALIZING BOOTSERVER - JOB DELETED
```

or

```
iRMX ERROR: xxxxH WHILE INITIALIZING BOOTSERVER - JOB DELETED
```

Where `xxxxH` is either an iRMX condition code or an MSA Firmware error code.

For the Bootserver to function, configure the I/O User's screen (IOUS) to a default I/O user named SUPER, with ID 0.

C Library

The shared C library is available as a loadable job (*clib.job*) or as an OS layer configurable with the ICU. The main advantage of using the ICU-configurable version is that you can configure a minimal C library for systems that do not use upper layers of the OS.

See also: *clib.job*, Chapter 4, for the loadable job description

You link each application to a small interface library, which provides access to the shared C library.

See also: Interface Libraries, *System Call Reference*

Configuring the C Library for OS Layers

The full C library requires these layers of the OS: Nucleus, BIOS, EIOS, Application Loader, and HI. These are all the OS layers except the UDI. If you configure out any of these layers with the ICU, you cannot use certain C functions. For example, the HI is required to parse command-line arguments (*argc*, *argv*). The minimal configuration for an application that makes C library calls is one that includes only the Nucleus and the C library.

Using the ICU, you can limit these functions available in the C library:

- Math support: The math coprocessor support library and related functions are included by default.
- HI functions: If you configure the HI out of the system, several C library functions that use the HI are configured out automatically. This includes parsing of command line arguments; when the HI is not present, *argc* is always zero and *argv* is always null.

  See also: **_get_arguments**, *C Library Reference*
- I/O functions: If you configure either the BIOS or EIOS out of the system, some I/O-related functions are automatically configured out of the C library.
- Console I/O: On the CLIB screen of the ICU, you can specify that console I/O is supported by the EIOS, by the SDM Monitor, by a user routine (that you supply), or that there is no support (Nucleus-only configuration). Console I/O supports the *stdin*, *stdout*, and *stderr* streams.

See also: *ci* and *co* interfaces, *System Debugger Reference*;
Function descriptions, *C Library Reference*

- TCP/IP Socket I/O. On the CLIB screen of the ICU, you can specify whether or not to include the TCP/IP sockets library as part of the C-Library. If you are planning to use the new TCP/IP stack in your system environment, you must include the TCP/IP sockets library as part of the C-Library.

See also: Function descriptions, *TCP/IP for the iRMX Operating System*

Multibus II Downloader Job

The Multibus II downloader job loads object files for the OS and application programs onto boards in a Multibus II system. This job is available both as an ICU-configurable job (on the DLJ screen) and as the **dload** command. If you configure this job with the ICU, it executes after the BIOS is initialized and before the EIOS and I/O jobs. The downloader job deletes itself when it completes. Figure 6-1 shows how the downloader loads files onto a Multibus II controller board.

See also: **dload**, Appendix B

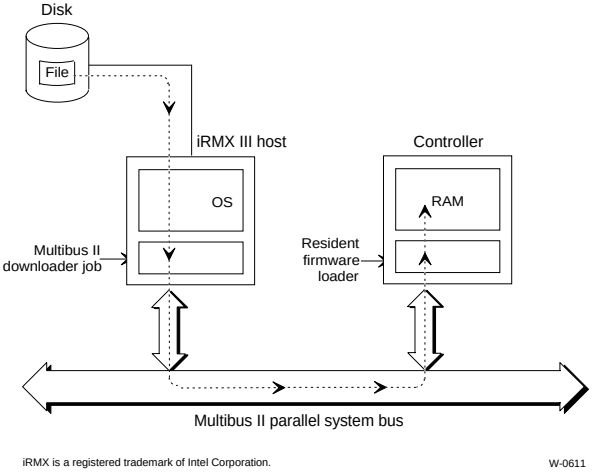


Figure 6-1. Downloading a File with dload

Front Panel Interrupt Server Job

The Front Panel Interrupt (FPI) Server Job simulates the CSM/001 board interrupt behavior for systems that use the CSM/002 module. A front panel interrupt is generated by the keyswitch on the front of the system. In the event of a front panel interrupt, the CSM/002 module only sends an interrupt to the host board to which it is mounted. Other hosts on the Multibus II bus do not know an interrupt has occurred, unless you configure the FPI server to run on the host for the CSM/002. You may have to increase the memory for the system (on the MEMS screen of the ICU) to accommodate the FPI Server job.

The FPI server runs on the host in slot 0 and waits for a front panel interrupt. The front panel interrupt signal must be connected to master PIC level 1. Any host on the Multibus II PSB, including the host in slot 0, can request the FPI Server to send notification of a front panel interrupt. The request for notification can be subsequently canceled. You make or cancel the request with one of these commands:

```
ic -c fpi arm
ic -c fpi disarm
```

See also: **ic** command, *Command Reference*

In the event of a front panel interrupt, each host requesting notification is signaled by a non-maskable interrupt (NMI) written into interconnect space. The FPI server does this by setting Bit 2 of the general control register (IC register 19h). Interconnect records for the FPI arming and disarming functions are only written to the host from which the **ic** command is invoked.

Limitations

The NMI is not cleared immediately. The SDM monitor may ignore several commands before beginning to respond normally.

The FPI Server does not exactly mimic the CSM/001 board. The CSM/001 has an interconnect register in which a single agent can set its own slot ID. When an interrupt occurs, a special four-byte message is sent to the specified agent. The FPI Server owns no registers and does not send such a message.

iNA 960 Network Jobs

The iNA 960 software provides a programmatic interface to the protocol layers of an ISO OSI network.

To configure an iNA 960 job in the ICU, on the NET screen choose either MIP (for a COMMengine MIP job) or CMP (for a COMMputer job). Then set parameters for a MIP job on the MIPJ screen or for a COMMputer job on the ICMPJ screen. With a COMMputer job you can also set Name Server parameters on the NS and NSDOM screen.

See also: *i*.job*, Chapter 4, for a list of specific iNA 960 jobs and their characteristics
MIP and COMMputer jobs, *Network User's Guide and Reference*

You must run an iNA 960 job to be able to run any of these:

- iRMX-NET
- TCP/IP
- NFS

iRMX-NET Jobs

The iRMX-NET network jobs provide transparent file access on an ISO OSI network based on iNA 960 software. There are two jobs; you can configure either one or both:

- iRMX-NET Server, configured on the NET, RSJ, and FS screens of the ICU
- iRMX-NET Client, which includes the File Consumer and Remote File Driver (RFD), configured on the NET, REM, and FC screens of the ICU

See also: *Network User's Guide and Reference* for how to use iRMX-NET

Note

If you use a loadable version of an iNA 960 job, you must use the loadable versions of the iRMX-NET Client and Server jobs. With iNA 960 configured into the OS as a first-level job, you can use either the loadable or first-level versions of iRMX-NET jobs.

Paging Subsystem Job

The paging subsystem manages memory with the processor in paging mode. This job, together with the *flat.job*, support flat model applications produced with non-Intel flat model compilers. The paging subsystem job makes available the **rqv_** system calls used by flat model applications. Enable the paging job with the PGS parameter on the SUB screen of the ICU. In the VSG parameter of the APPL screen, specify the size of virtual segment the paging subsystem will create when you load a flat model application. The size is rounded up to the nearest 4 Mbytes.

Configure memory for identity mapping on the PIMM screen. The start and end addresses that you enter are the 32-bit physical memory locations that the paging subsystem will identity-map into the linear address space. Do not enter any memory areas that have reserved for OS or Free Space Manager (MEMS and MEMF screens).

The memory blocks that you identity-map are locations not managed by the OS that your flat model application needs to access. For example, you might write a custom device driver that performs I/O in memory space outside the OS. With a flat model application you would need to identity-map that memory. However, the general memory used by a flat model application should not be identity mapped. Memory that the application allocates is automatically handled by the paging subsystem.

You can use either the first-level paging job or the loadable version of paging.job. When you configure the paging job into a system image, the ICU also includes the flat job in the same image.

See also: *paging.job* and *flat.job*, Chapter 4

PCI Server Job

You must run a PCI server job for each instance of a SCSI host adapter board installed in the system. The PCI server runs on these SBC boards:

Multibus I	**Multibus II**
386/12S	386/258 and 386/258D
486/12S	486/133SE and 486/166SE
PCP4DX and PCP4SX	486SX25, 486DX33, and 486DX66
	P5090, P5090ISE, P5120ISE, and P5200

In a Multibus II system where the PCI server and driver are on different host boards, they exchange commands, status, and data across the bus through message passing. The PCI server enables a host driver's view of the I/O server board and its SCSI peripheral resources to conform to the standard client-server model defined by the Multibus II Transport Protocol.

In a Multibus I system or a Multibus II system where the PCI server and driver are on the same host board, the server and driver exchange commands, status, and data using short-circuit message passing. iRMX default configurations for the SBC 386/12S and 486/12S boards include the Communication Service layer for message passing and use the same PCI driver and server as Multibus II systems.

The PCI server manages up to 64 outstanding commands and permits multi-threaded I/O operations with up to 56 SCSI peripheral devices.

PCI Server SCSI Pass-through Capability

The PCI protocol gives host device drivers a generic view of random and sequential access to peripheral devices that adhere to the SCSI protocol. Host device drivers can benefit from, but are not limited by, this vendor-independent view. Many peripheral vendors offer unique capabilities that are accessed using vendor-specified commands.

Some of the PCI server jobs for particular boards give access to this functionality by providing a SCSI pass-through option. The PCI server still manages the transaction. However, the host provides the actual command that is passed over the SCSI bus to the device.

See also: *How to Use the PCI Server* manual

Soft-Scope Kernel

You can use the ICU to configure the kernel part of the Soft-Scope debugger, rather than loading the loadable version of *sskernel(ssk.job)*. Add the job on the SYSJ screen and configure parameters for it on the SSKJ screen. You must load the kernel before you can invoke Soft-Scope at the command line.

If you want to use System Debugger (SDB) commands from Soft-Scope, configure SDM and SDB on the SUB screen of the ICU.

Note

To use Soft-Scope, you must also configure the C Library, either as a first-level job or a loadable job.

See also: C Library Job, this chapter
clib.job, Chapter 4

See also: Chapter 8

❑❑❑

Basic Concepts 7

Writing a Loadable Job

In any iRMX OS, you can write an application as a loadable job. If you choose, you can create a loadable version of your application job for debugging, then a non-loadable version for later incorporation with the OS using the ICU.

Write Your Application as an HI Command

As an HI command, your application can be invoked just like any iRMX OS command, and run either from the command prompt or configuration file. The main difference from an OS command, is that the command prompt never returns. It continues to run until explicit termination, system shutdown, or reboot.

See also: Creating commands, *System Concepts*

If your job requires access to the *:ci:* and *:co:* standard input and output streams, use the -i and -o options in the **sysload** command. If a loadable job cannot access the input and output streams:

- It cannot accept input from the keyboard.
- It cannot display error or exit messages.
- If there is an unrecoverable error, it should delete itself or take some action other than calling the **exit_io_job** system call.
- It should create a log file using the same name as the job with *.log* appended, in the directory where the job is located.

Establish Priorities for Tasks

Your application may need to create interrupt handlers and tasks. If so, it must establish a higher priority than the job that invokes it (the **sysload** command). To do this, the initial task in the loadable job makes a **set_max_priority** system call, using a priority value of 0. This enables the job to create tasks with sufficient priority for its needs, such as interrupt handlers and tasks. You should give time-critical parts of the job priority over less critical parts.

See also: Writing loadable drivers, *Driver Programming Concepts*;
set_max_priority, *System Call Reference*

Debug the Loadable Job or Driver

Load the job (or driver) with the **debug** command or the Soft-Scope debugger, rather than **sysload**. This removes the code from memory at the end of the debug session. Be sure to detach all devices before the session terminates; otherwise, reboot the system. Then, modify the code to fix errors discovered during the debug session, and reload the driver to continue debugging.

See also: **debug** and **detachdevice**, *Command Reference*;
Soft-Scope Debugger User's Guide

You can test a loadable job or driver by invoking it as a background job rather than loading it with **sysload**. In this case, the job remains in memory only until you terminate it or log off.

You must use an alternate console when testing a loadable device driver. The device driver has an initialization front-end that suspends itself, leaving necessary code (such as the driver code) in memory. The command prompt never returns.

See also: **background** and **kill**, *Command Reference*;
Making a device driver loadable, *Driver Programming Concepts*

Automatically Booting DOSRMX and iRMX for PCs

You can load DOSRMX and iRMX for PCs from the DOS *autoexec.bat* file. This works with any supported version of DOS, 5.0 and later.

See also: *Installation and Startup* manual

You can load DOSRMX and iRMX for PCs from the *config.sys* file, but only on a DOS 5.0 file system.

Modify *config.sys* to begin with these lines (you may use other values for `buffers`, `files`, and `lastdrive` if you choose to do so):

```
buffers=40
files=64
lastdrive=Z
install=C:\dos\himem.sys
dos=high
install=C:\dosrmx\rmxtsr.exe
install=C:\dosrmx\loadrmx.exe -n C:\dosrmx\dosrmx -s c_dos -f d –w

In an iRMX for PCs system being loaded from DOS, omit the lines

install=C:\dos\himem.sys
dos=high
```

⚠ **CAUTION**

Only use the `install` commands listed above for starting the iRMX OS on DOS 5.0. You may not be able to reliably install the iRMX OS from the *config.sys* file using DOS 6.x. If you receive an “Error in config.sys line xx” message when the system boots, try ignoring the error and determine whether the iRMX OS has indeed been loaded.

Loading Your Application

You may want your application to run automatically when you boot the system, without a user ever seeing the iRMX screen. You can configure a bootable OS image containing your application using the ICU. In DOSRMX and iRMX for PCs, you can start an application in the *:config:loadinfo* file. For DOSRMX, you use `d_cons` as the console; in iRMX for PCs, use `con` as the console.

Note

You can also write a custom loadable command interface to replace the iRMX Command Line Interpreter.

You can also run your job (written as an HI command) in these startup files:

:config:user/<username>

Specify your application as the initial program in the user attributes file, rather than the CLI. Define a static user for the console device. The application need not have a user interface if you do not use the console. This initial program should not exit; it would leave the console user job in an indeterminate state.

:config:r?init

Place your application command after the command that submits *:config:loadinfo*. This method is equivalent to invoking a program (without **sysload**) as the last command in the *loadinfo* file.

These limitations apply only to an application that runs in the context of the console device and *r?init*:

- If the console device is set up as a dynamic terminal, HI initialization does not complete on the console until you press <Alt-SysRq> to switch to the iRMX prompt. If your application runs in the context of the console device, define the console device as a static logon terminal. This avoids having to switch to the iRMX screen.

- The amount of screen output generated for the static user on the console can prevent initialization from completing. If enough output is generated to fill the screen buffer, the HI does not complete initialization until you switch to the iRMX screen. To prevent this, in the static user's *:prog:r?logon* file, submit any commands to the null device:

  ```
  submit filename over :bb:
  ```

See also: HI initialization and logon, Chapter 1

❑❑❑

Browsing/Cross Debugging an iRMX III.2.3 System from a Windows NT Host 8

Once the iRMX III.2.3 Development Software has been installed on a Windows NT system, you can use the NT system as a workstation on which you develop, download, cross-debug, and monitor iRMX applications and systems running on a remote target. This chapter describes the steps that need to be taken to establish a link between the NT host and the iRMX target system. This link is referred to as an NTX (NT eXtension) link. The NTX link was developed as part of the INtime Real-Time Extension package for Windows NT and is described and fully featured in that software package. A subset of the INtime NTX features are provided as part of the iRMX III.2.3 Development Software so that the same tools that run under NT on an INtime system can be used to do downloading, cross debugging and system browsing of an iRMX III.2.3 application system from a Windows NT Host.

The NTX Link between the NT Host and an iRMX target system can use either a serial connection (at baud rates between 9600 Baud and 115.2 Kbaud) or an ethernet connection running TCP/IP.

The NT applets that make use of the NTX Link and their function, are as follows:

\INtime Explorer	Browser that displays the state of iRMX jobs, tasks, and other objects on a remote iRMX target system
INtime Application Loader	Loader which can download OMF386 and MSVC Developer Studio-generated iRMX applications onto a remote iRMX target system
NT-Hosted Soft-Scope Debugger	Source-Level Dynamic Debugger that can debug OMF386 and MSVC Developer Studio-generated iRMX applications on a remote iRMX target system. It programmatically invokes the INtime Application Loader to download the application to be debugged.
Find iRMX Applet	Utility that displays all currently known iRMX target systems and the state of their NTX link.
NTX Link Setup Utility	Utility that allows you to configure an NTX link for one or more remote iRMX targets.
Remote Node Connection Manager	NT Service that manages the NTX connection on the NT Host

Using Windows NT as a Cross Development Platform for iRMX III.2.3

You must first use the iRMX III NTX Link Setup utility to set up the NTX Interface channels on the NT system that will be used to communicate with the iRMX target system. Set up the link(s) as follows:

Configuring an NTX Link to an iRMX III R2.3 system

Preparing the NT Host System

iRMX III NTX Link Setup Utility

You must first use the iRMX III NTX Link Setup utility to set up the NTX Interface channels on the NT system that will be used to communicate with the iRMX target system. Set up the link(s) as follows:

1. Start the iRMX III NTX Link Setup utility (Start/Programs/iRMX III Interface Tools/iRMX III NTX Link Setup)
2. Click the Add button in the Remote NTX connections tab. The Remote Client Settings screen displays.
3. Complete the fields on each of these tabs:
   - NTX tab
     Be sure to select the appropriate Remote Client Channel and complete the prompts for that channel as they are displayed on the right side of the screen. Leave the Channel ID button deselected.
   - Setup tab
     Be sure to specify the name of a subdirectory to be created in the \iRMXIII\Remote directory in which will be placed setup and configuration files that can be used on the target iRMX system to establish the NTX Link being setup.
4. Click the OK button. The Remote NTX Connections tab displays.
5. Click the Apply button to update the registry with information about this RT node.

6. Click the Build System button to build the RT subsystem files with the settings you specified. If a message indicates that a setting is not correct, you must fix the error, click the Apply button, then build again.

   When successful, a message tells you where the build files were placed. Build files include:

   ntlink.csd
   tcp.ini (UDP/IP channel only)

   Use the contents of the files created in step 6 to modify the configuration of the iRMX target system so that it can establish an NTX interface to the NT host. The jobs with their necessary input parameters that must be run on the iRMX system to establish this NTX link are found in the file ntlink.csd. If you are using UDP/IP as your NTX channel, the tcp.ini file contains the iRMX TCP Stack configuration parameters needed to establish the link.

   Note
   You can use the files **ntlink.csd** and **tcp.ini** on the iRMX target system to start the iRMX half of the NTX link. If you are using a UDP connection, be sure that a "sleep 5" line is present in the **ntlink.csd** file between the loading of the NIC and Loopback drivers, and the loading of the TCP/IP stack modules

7. Exit the iRMX III NTX Link Setup utility and shutdown and restart Windows NT for your NTX link configurations to take affect.

Activating the Remote Node Connection Manager

In order to establish an NTX connection with an iRMX III R2.3 system, you must first start the Remote Node Connection Manager on the NT Host. Start the Control Panel/Services Applet and use it to start the INtime Remote Connection Manager.

Note
If you start the INtime Remote Connection Manager before a Remote Client (i.e. iRMX target system) has been established using the iRMX III NTX Link Setup Utility, this Remote Client will not be visible until the NT Host has been restarted.

Finding an iRMX Node

Use the **Find iRMX** Applet (Start/Programs/iRMX III Interface Tools/Find iRMX) to display the current set of Remote Clients (i.e. iRMX target systems) that the NT

Host knows about. This Applet requires the INtime Remote Connection Manager to be started. The **Find iRMX** Applet also shows the current state of the NTX Connection. A **?** means the iRMX target has not been found; an **T-**like figure means the connection is in operation; and an **T-**like figure with a **red X** overlaid upon it means the NTX connection was previously established but is now non-functional.

Preparing the Target System

Remote INtime Personality Job

The target system must run the Remote INtime Personality Job, as well as the Paging and Flat jobs. These can jobs can either be configured into the target system via the ICU, or they can be sysloaded. An example of sysloading these jobs follows:

```
- sysload /rmx386/jobs/paging.job
- sysload /rmx386/jobs/flat.job
- sysload /rmx386/jobs/rintmjob.job
```

See also: *rintmjob.job*, Chapter 4
paging.job, Chapter 4
flat.job, Chapter 4

New TCP/IP Stack

If the interface channel between the NT host and the iRMX target is ethernet, the new TCP/IP stack must be running on the target system. You can either start it during system initialization by loading it from the :CONFIG:loadinfo file, or you can use the :CONFIG:tcpstart.csd submit file to load the new TCP/IP stack manually. In either case, the C-Library Job must also be present in the system.

The file ntlink.csd created by the iRMX III NTX Link Setup utility on the NT Host can also be used to start all the jobs needed to establish a remote NTX connection on an iRMX system. Also, the file tcp.ini created by the iRMX III NTX Link Setup utility on the NT Host can be used to configure the TCP/IP stack.

You must also run the UDP Channel Interface Module rtcimudp.job which is loaded as follows:

```
- sysload /rmx386/jobs/rtcimudp.job <NT Host IP address>
```

See also: Chapter 8 of the *TCP/IP for the iRMX Operating System* manual,
rtcimudp.job, Chapter 4

Ports-based Serial Driver

If the interface channel between the NT host and the iRMX target is serial, the new ports-based serial driver must be running on the target system. On a PC Architecture-based target system, you can select either COM1 or COM2 as the serial channel for the link. Assuming you choose COM2, load the ports-based serial driver as follows:

```
- sysload /rmx386/jobs/serdrvr.job(com2)
```

The file ntlink.csd created by the iRMX III NTX Link Setup utility on the NT Host can also be used to start all the jobs needed to establish a remote NTX connection on an iRMX system.

You must also run the Serial Channel Interface Module (rtcimcom.job) which is loaded as follows:

- sysload /rmx386/jobs/rtcimcom.job <COM channel> -m<Baud Rate>

See also: *serdrvr.job*, Chapter 4,
rtcimcom.job, Chapter 4

NTX Interface Job

The target system must also run the NTX proxy (ntxproxy.job). Regardless of the Channel Interface chosen, load the NTX proxy as follows:

- sysload /rmx386/jobs/ntxproxy.job>

See also: ntxproxy.*job*, Chapter 4

Using an NTX Link to Browse/Debug an iRMX III.2.3 system

Debugging an iRMX Application from a Windows NT system using Soft-Scope

The **Soft-Scope Debugger** is an NT application that establishes an NTX link with an iRMX target system, downloads and connects with a version of sskernel, and then communicates with this debug server to debug an iRMX application. This application can be downloaded by Soft-Scope for debugging, or it can be loaded from the iRMX system and have only its symbols loaded on the NT Host for debugging. Use the on-line help file on the NT Host for further information on the use of Soft-Scope. In the iRMX III.2.3 software, INtime

terminology is used, i.e. the name Real-Time Thread is used instead of iRMX Task, and the name Real-Time Process is used instead of iRMX Job.

Use the **Soft-Scope Debugger** Applet (Start/Programs/iRMX III Interface Tools/Soft-Scope Debugger) to debug your iRMX application on a visible iRMX target system. This Applet requires the INtime Remote Connection Manager to be started.

1. Start the Soft-Scope Applet
2. Highlight the desired iRMX target system from the list of available target systems shown by the Remote Node Browser (part of the Soft-Scope Applet) and click on OK
3. Use the File/Load pulldown menu to select the NT resident iRMX application to debug
4. Debug the application as required, following the on-line help for detailed debugging information.
5. Use the File/Load Symbols pulldown menu to load the symbols of a NT resident iRMX application to resume debugging or to begin debugging if the application was loaded from the iRMX system itself.

Browsing the state of an iRMX system from a Windows NT system using the INtime Explorer (INtex.exe.)

The **INtime Explorer** is an NT application that establishes an NTX link with an iRMX target system, downloads an iRMX viewer program, and then communicates with this viewer to display the state of this iRMX system. The displayed information is similar to that shown by the iRMX System Debugger (SDB) except that it is a snapshot of a dynamic system rather that a display of the static (the System Debug Monitor is active) state of the iRMX system. The state of the various jobs, tasks, and other objects in an iRMX target system can be viewed on the NT console. In the iRMX III.2.3 software, INtime terminology is used, i.e. the name Real-Time Thread is used instead of iRMX Task, and the name Real-Time Process is used instead of iRMX Job.

Use the **INtime Explorer** Applet (Start/Programs/iRMX III Interface Tools/INtime Explorer) to display the current state of a visible iRMX target systems. This Applet requires the INtime Remote Connection Manager to be started.

1. Start the INtime Explorer Applet
2. Highlight the desired iRMX target system from the list of available target systems shown by the Remote Node Browser (part of the INtime Explorer Applet) and click on OK
3. Double click on various displayed objects to show their current state

□□□

Keyboard and Console Information A

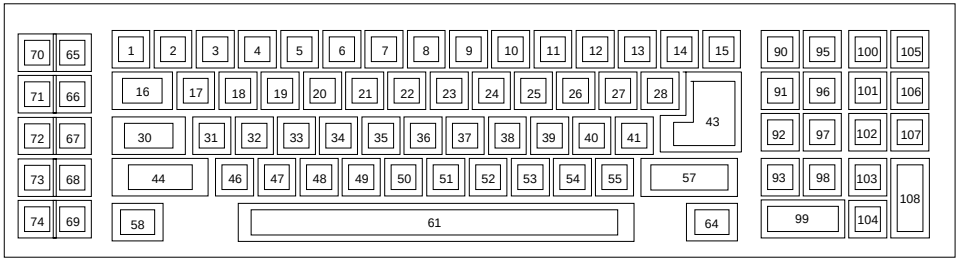
Keyboard Support

In iRMX for PCs and DOSRMX, the Console Driver is the link between the OS and a special terminal called the console. The driver interfaces with two physical devices: an output device (video card) and an input device (keyboard).

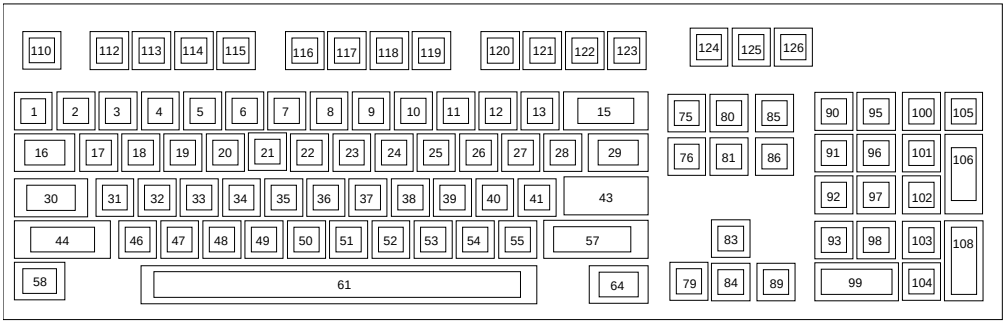
The video card can be an EGA, VGA, SVGA, or any completely compatible alternative. Operate the display adapter only in character mode with the console driver. Graphics mode is not supported. An application can set the EGA device to graphics mode, but should restore it to character mode before performing any character output using the console driver.

Two keyboard types are supported: the 84-key and the 101/102-key keyboards. See Figure A-1 for key diagrams and refer to Table A-1 for key code translations.

IBM 84-Key Keyboard



IBM 101/102-Key Keyboard



W-0496

Figure A-1. Supported Keyboards

Table A-1. Keyboard Codes

IBM 84-Key Keyboard	IBM 101/102-Key Keyboard	ASCII Character	
		No Shift Key	With Shift Key
29H	1BH	ESC	
02H	02H	1	!
03H	03H	2	@
04H	04H	3	#
05H	05H	4	$
06H	06H	5	%
07H	07H	6	^
08H	08H	7	&
09H	09H	8	*
0AH	0AH	9	(
0BH	0BH	0	)
0CH	0CH	-	_
0DH	0DH	=	+
0EH	0EH	Back Space	
0FH	0FH	Tab	
10H	10H	q	Q
11H	11H	w	W
12H	12H	e	E
13H	13H	r	R
14H	14H	t	T
15H	15H	y	Y
16H	16H	u	U
17H	17H	i	I
18H	18H	o	O
19H	19H	p	P
1AH	1AH	[	{
1BH	1BH	]	}
1CH	1CH	Return	
1DH	1DH	Ctrl	
1EH	1EH	a	A
1FH	1FH	s	S
20H	20H	d	D
21H	21H	f	F
22H	22H	g	G

continued

Table A-1. Keyboard Codes (continued)

IBM 84-Key Keyboard	IBM 101/102-Key Keyboard	ASCII Character	
		No Shift Key	With Shift Key
23H	23H	h	H
24H	24H	j	J
25H	25H	k	K
26H	26H	l	L
27H	27H	;	:
28H	28H	'	"
	29H	`	~
2AH	2AH	Left Shift	
2BH	2BH	\	|
2CH	2CH	z	Z
2DH	2DH	x	X
2EH	2EH	c	C
30H	30H	v	V
31H	31H	b	B
32H	32H	n	N
33H	33H	m	M
33H	33H	,	<
34H	34H	.	>
35H	35H	/	?
36H	36H	Right Shift	
	37H	* (num keypad)	
38H	38H	Left Alt	
39H	39H	Space	
3AH	3AH	Cap Lock	
3BH	3BH	F1	
3CH	3CH	F2	
3DH	3DH	F3	
3DH	3DH	F4	
3EH	3EH	F5	
40H	40H	F6	
41H	41H	F7	
42H	42H	F8	
43H	43H	F9	
44H	44H	F10	

continued

Table A-1. Keyboard Codes (continued)

IBM 84-Key Keyboard	IBM 101/102-Key Keyboard	ASCII Character	
		No Shift Key	With Shift Key
45H	45H	Num Lock	
46H	46H	Scroll Lock	
47H	47H	Home	
48H	48H	7 (num keypad)	
49H	49H	8 (num keypad)	
4AH	4AH	9 (num keypad)	
4BH	4BH	4 (num keypad)	
4CH	4CH	5 (num keypad)	
4DH	4DH	6 (num keypad)	
4EH	4EH	+ (num keypad)	
4FH	4FH	1 (num keypad)	
50H	50H	2 (num keypad)	
51H	51H	3 (num keypad)	
52H	52H	Ins	
53H	53H	Del	
	E0 10	Right Ctrl	
	E0 1C	Enter (num keypad)	
	E0 2A 37H	Prt Scr	
	E0 36H	/ (num keypad)	
	E0 38H	Right Alt	
	E1 10 45H	Pause	
	E0 47H	Home	
	E0 49H	Page Up	
	E0 4FH	End	
	E0 51H	Page Down	
	E0 52H	Insert	
	E0 53H	Delete	

Table A-2 lists the values returned by the console driver when various combinations of the function keys (F1 through F12), Alt, Ctrl, as well as the letters are pressed. The console driver returns a NULL code (00) in front of each function code. For example, pressing <Alt-F1> returns the value 0068H. By checking for this NULL your code can determine if a function key combination has been returned.

Table A-2. Function Key Codes

Keystroke	Value Returned	Keystroke	Value Returned
F1	00, 3BH	Shift F1	00, 54H
F2	00, 3CH	Shift F2	00, 55H
F3	00, 3DH	Shift F3	00, 56H
F4	00, 3EH	Shift F4	00, 57H
F5	00, 3FH	Shift F5	00, 58H
F6	00, 40H	Shift F6	00, 59H
F7	00, 41H	Shift F7	00, 5AH
F8	00, 42H	Shift F8	00, 5BH
F9	00, 43H	Shift F9	00, 5CH
F10	00, 44H	Shift F10	00, 5DH
F11	00, 85H	Shift F11	00, 87H
F12	00, 86H	Shift F12	00, 88H
Ctrl F1	00, 5EH	Alt F1	00, 68H
Ctrl F2	00, 5FH	Alt F2	00, 69H
Ctrl F3	00, 60H	Alt F3	00, 6AH
Ctrl F4	00, 61H	Alt F4	00, 6BH
Ctrl F5	00, 62H	Alt F5	00, 6CH
Ctrl F6	00, 63H	Alt F6	00, 6DH
Ctrl F7	00, 64H	Alt F7	00, 6EH
Ctrl F8	00, 65H	Alt F8	00, 6FH
Ctrl F9	00, 66H	Alt F9	00, 70H
Ctrl F10	00, 67H	Alt F10	00, 71H
Ctrl F11	00, 89H	Alt F11	00, 8BH
Ctrl F12	00, 8AH	Alt F12	00, 8CH

continued

Table A-2. Function Key Codes (continued)

Keystroke	Value Returned	Keystroke	Value Returned
Alt 1	7800H	Alt 6	7D00H
Alt 2	7900H	Alt 7	7E00H
Alt 3	7A00H	Alt 8	7F00H
Alt 4	7B00H	Alt 9	8000H
Alt 5	7C00H	Alt 0	8100H
Alt a	1E00H	Alt n	3100H
Alt b	3000H	Alt o	1800H
Alt c	2E00H	Alt p	1900H
Alt d	2000H	Alt q	1000H
Alt e	1200H	Alt r	1300H
Alt f	2100H	Alt s	1F00H
Alt g	2200H	Alt t	1400H
Alt h	2300H	Alt u	1600H
Alt i	1700H	Alt v	2000H
Alt j	2400H	Alt w	1100H
Alt k	2500H	Alt x	2D00H
Alt l	2600H	Alt y	1500H
Alt m	3200H	Alt z	2C00H
Alt Esc	Not Supported	Alt `	Not Supported
Alt -	8200H	Alt =	8300H
Alt Backspace	Not Supported	Alt Tab	Not Supported
Alt [(or {)	Not Supported	Alt] (or })	Not Supported
Alt \	Not Supported	Alt ;	Not Supported
Alt '	Not Supported	Alt <CR>	Not Supported
Alt ,	Not Supported	Alt .	Not Supported
Alt /	Not Supported	Alt <Space>	Not Supported
Alt Insert	Not Supported	Alt Home	Not Supported
Alt Page Up	Not Supported	Alt Delete	Not Supported
Alt End	Not Supported	Alt Page Down	Not Supported
Alt <Up Arrow>	Not Supported	Alt <Left Arrow>	Not Supported
Alt <Down Arrow>	Not Supported	Alt <Right Arrow>	Not Supported
Alt / (keypad)	Not Supported	Alt * (keypad)	3700H
Alt - (keypad)	4A00H	Alt + (keypad)	4E00H
Alt <CR>(keypad)	Not Supported		

These extended keyboard functions are supported:

<Ctrl-Alt-Break>
: Causes the system to break to the SDM debug monitor.

<Alt>
: When input is in raw mode, pressing and releasing the Alt key puts the system back to normal mode.

<Scroll Lock>
: Controls text scrolling on the screen. This key serves the same function as <Ctrl-S> and <Ctrl-Q>.

<Ctrl-Alt-Del>
: Resets the system without performing a **shutdown** command. Because no **shutdown** command is performed when using this key sequence, an error message will be displayed when the system is rebooted.

<Alt-Plus> (using the + key on the numeric keypad)
: Change the foreground color.

<Alt-Minus>
: (using the - key on the numeric keypad)
Change the background color.

<Alt-SysRq>
: Toggle between DOS and iRMX OS.

Console Output Codes

The *keybd.job* console driver interprets console codes (ASCII characters 0 - 1F) as shown in Table A-3.

Each screen character has two bytes associated with it: one is the ASCII character and the other is the attribute byte. Certain console codes affect the attribute byte. The *current attribute* is the attribute in effect when you write to the screen. Set the current attribute with code 0BH and manipulate it with code 15H.

Table A-3. Console Codes

Hex Code	Meaning
01	Erase rest of screen
02	Erase line
03	Erase rest of line
04 rr cc	Set cursor absolute position; rr = row (0-18H)+20H, cc = column (0-4FH)+20H
05	Line delete (following lines are shifted up)
06	Line insert (following lines are shifted down)
07	Bell
08	Backspace
09	Ignored (on other consoles interpreted as a tab)
0A	Linefeed
0B cc	Set current attribute for the color of all characters to be output, where cc is the attribute byte to be stored in video memory (see also code 15H) **Bit** **Meaning** 7 bl: foreground blink 6 br: background red 5 bg: background green 4 bb: background blue 3 fi: foreground intensity 2 fr: foreground red 1 fg: foreground green 0 fb: foreground blue Note: some video controllers interpret fi-fr-fg-fb as a 4 bit color code; foreground blinking is not always supported
0C	Erase screen
0D	Carriage return

continued

Table A-3. Console Codes (continued)

Hex Code	Meaning
0E	Next character is output literally; if it is in the range 00-1FH, it is not interpreted as a special character. This allows output of special characters like card symbols. For example, code 01 ordinarily erases the rest of the screen, but code 0E 01 outputs the smiling face symbol.
0F	Set raw mode: keyboard input will come in not as ASCII codes, but as scan-codes. Pressing a key returns its scan-code+20H, releasing a key returns scan-code+A0H. Scan-codes are defined in the keyboard documentation, and do not depend on country information. Note: releasing the <Alt> key always ends raw mode, so to kill an application that sets raw mode, press and release <Alt>, then <Ctrl-C>.
10	Set cooked (processed) mode: this is the normal mode where pressing a key causes an ASCII code to be returned.
11	X-ON
12	Ignored
13	X-OFF
14 ff	Screen saver: ff=31H means ON, else off
15 cc	Attribute functions (codes 0BH and code 15H manipulate the same attribute, but in different ways) **cc (hex) Meaning** 01 Save current attribute (there is one save area) 02 Restore current attribute from save area 03 Set entire screen to current attribute 04 Invert entire screen; swap foreground and background colors 05 Set current attribute to default (established as a parameter when you load *keybd.job*) 06 Leave attribute as it is (opposite of code 0BH; when you write characters to the screen you do not write a current attribute) 07 Set blink bit in current attribute 08 Clear blink bit in current attribute 30 Use alternate character set (i.e. add 80H to characters <80H) 31 Stop using alternate character set
16	Delete character at the cursor
17 cd	Background: change upper four bits of the current attribute field to c (see code 0BH)
18 cd	Foreground: change lower four bits of the current attribute field to d

continued

Table A-3. Console Codes (continued)

Hex Code	Meaning
19	Cursor right (wraps)
1A	Ignored
1B	Ignored
1C	Cursor down
1D	Cursor home
1E	Cursor up
1F	Cursor left (wraps)

Note

There is no way to return the current or default attribute value from the driver to your application.

❑❑❑

Multibus II Downloader B

The Multibus II downloader loads object files onto boards in a Multibus II system. You can either invoke **dload** as an HI command (a job that deletes itself when it completes) or configure the downloader job with the ICU. A configuration file specifies which files are downloaded to which boards.

Downloader Configuration File

The downloader uses the *:config:dload.mb2* configuration file to determine what files to download and what boards to load. A default copy of this file is installed in the *:config:default* directory. Use this copy to create your own version in the *:config:* directory. The file has this format:

```
number_of_entries
board_name, instance, [slot_ID], filename, [omf_type],
[reset|noreset|resetoverride], [loadngo|loadonly],
[program_table_index]
```

Where `board_name` through `[program_table_index]` are all on one line, and there may be multiple such lines. Brackets ([ ]) indicate optional parameters. Optional parameters separated by a bar (|) indicate that you choose one value. Unspecified optional parameters require a comma as a place holder. The parameter descriptions are:

`number_of_entries`
: Number of valid entries in the file, from 1 - 255. The downloader operates only on valid entries, and skips lines that have invalid parameters.

`board_name`
: Name of the target board for the downloader, up to 10 characters long. With *atscdrv*, the only appropriate names are 186/410 and 186/450. You can use the **agents** alias or the `ic -c agents` command to display the names of boards in the system.

`instance`
: Specifies which of multiple identical boards to load. 1 specifies the first board with `board_name` in the system, counting up from slot 0; 2 specifies the second such board, etc. If `instance` is not specified or is 0, it is ignored and the `slot_id` is used instead. If `instance` is not 0, `slot_id` is ignored.

`slot_ID`
: The Multibus II host ID (slot number) of this board.

`filename`
: Pathname of the file to download. This name can be up to 45 characters long.

`omf_type`
: Object module format (OMF) type of the file image to download. If you do not specify an OMF type, the default is OMF86. The supported OMF types are:

  OMF86 — Absolute output generated by LOC86 or LIB86.

  OMF286 — Absolute output from BLD286 that starts in real mode.

  OMF386 — Absolute output from BLD386 that starts in real mode.

  COFF386 — Files of this OMF type are *a.out* images that load only text and data. The file header must have a magic number of 14CH to indicate that the target board uses an Intel386, Intel486, or Pentium microprocessors. The *a.out* header must have a magic number of 10BH to specify that text and data are aligned so they can be directly paged.

  Because there can be no start address in a downloaded OMF286 or OMF386 image, the file must be loaded with the `loadonly` option. After loading such an image, download an OMF86 stub that transfers control to the OMF286 or OMF386 image. This OMF86 stub must be loaded with the `loadngo` option and must contain a far jump to the start address of the OMF286 or OMF386 image.

`reset|noreset|resetoverride`
: Specifies whether to reset the target board before downloading. The default is `noreset`. The `reset` option resets the target board. Use the `resetoverride` option along with a `program_table_index` to invoke a downloader that coexists with other programs in the target board firmware.

`loadngo|loadonly`
: Specifies whether the downloaded image is started immediately after downloading. The default is `loadngo`. Use the `loadonly` parameter to download an OMF286 or OMF386 image.

`program_table_index`
: This is an offset in a program table that specifies the location of the downloader in firmware on the target board. Use this with the `resetoverride` parameter.

  See also: Target board's hardware reference manual for more details

Download Image Files

For example, use **dload** to download ATCS software for *atcsdrv*. The download image files for the SBC 186/410 and 186/450 terminal controller boards are in the */rmx386/ios* directory. Set up the downloader configuration file to load the

appropriate file(s) and invoke **dload** before you load the *atcsdrv* driver for these boards:

atcs.410 A downloadable OMF86 binary image consisting of the iRMX I Nucleus and a server for the 186/410 board.

atcs.450 A downloadable OMF86 binary image consisting of the iRMX I Nucleus and a server for the 186/450 board.

The default configuration file specifies that the first two instances of a 186/410 board be loaded with the file *:rmx:ios/atcs.410*, and the first two instances of a 186/450 board be loaded with the file *:rmx:ios/atcs.450*. Because the `instance` parameter is used, the Multibus II slot numbers are not specified. The object module format is also not specified, so OMF86 (the default) is indicated. All boards are reset before the file is downloaded. Any existing boards are initialized with this configuration.

```
4
186/410, 1,, :rmx:ios/atcs.410,, reset,
186/410, 2,, :rmx:ios/atcs.410,, reset,
186/450, 1,, :rmx:ios/atcs.450,, reset,
186/450, 2,, :rmx:ios/atcs.450,, reset,
```

If two 186/410 boards are in slots 5 and 6, you could set up the configuration file as follows, specifying a slot number rather than a board instance:

```
2
186/410, 0, 5, :rmx:ios/atcs.410,, reset,
186/410, 0, 6, :rmx:ios/atcs.410,, reset,
```

Receiving Board Requirements

The Multibus II board receiving the downloaded file must have a firmware communication record in interconnect space with global read/write access. The resident firmware loader on the board receiving the file must be executing and ready to accept commands, as indicated by the READY bit in the first byte of the firmware communication record. Currently, the 186/410 and 186/450 boards are shipped with a resident firmware loader.

See also: Information on the resident firmware loader, in the hardware manual for the board

Example Code

This example shows the ASM86 code needed to transfer control to an OMF286 or OMF386 image. To specify the start address, replace `xxxx` and `yyyy` with the starting CS and IP register values in the map file of the OMF286 or OMF386 image.

```
START_CS  EQU  xxxxH  ; Start executing downloaded code here
START_IP  EQU  yyyyH

CODE SEGMENT  'CODE'
ASSUME  CS:CODE
   PUBLIC start

start:
   DB  0EAH
   DW  START_IP, START_CS  ; FAR JMP to downloaded code

CODE      ENDS
END start
```

Error Messages

Dload writes downloading status to the *:config:dload.log* file. When a fatal error occurs, the configuration line processed in *:config:dload.mb2* is aborted and the rest of the lines in the file continue to be processed.

The downloader job **(dload** command) expects the target board to be in a known state for downloading, which can only be guaranteed on reset. Otherwise, the downloader job erroneously reports Target Not Ready.

The downloader reports two types of errors:

- General errors that occur before or after object module processing
- OMF errors that occur during object module processing

Most download errors are considered fatal. When a fatal error occurs, the configuration line being processed in the *:config:dload.mb2* configuration file is aborted, but the rest of the lines in the file are processed.

General Error Messages

This section lists the errors that can occur before or after object module processing.

`Attempting To Load Own Slot`
: The instance or the slot number is the same PSB slot ID as the host's slot ID. This is a fatal error.

`Bad Command Line`
: There is a syntax error on the command line or the command line does not exist. This is a fatal error.

`Cannot Open Load File`
: The load file specified in the command line cannot be accessed. Check the pathname and file permissions. This is a fatal error.

`Download Error`
: An error occurred while downloading the data. This is a fatal error.

`Firmware Communication Record Not Found`
: The firmware communication record on the target cannot be found in interconnect space. This is a fatal error.

`Instance Of Board Not Found`
: The instance of the specified board does not exist in the Multibus II backplane. This is a fatal error.

`Invalid Object Module Format`
: The object module format specified on the command line is not supported by the downloader. This is a fatal error.

`Invalid Slot ID`
: The target slot ID specified in the command line is out of range. Valid slot IDs range from 1 to 19. This is a fatal error.

`No Memory Available`
: The system cannot allocate the buffer needed for OMF processing. This is a fatal error.

`Specified Board Does Not Exist In Slot`
: The board name specified does not match the board in the slot. This is a fatal error.

`Target Not Ready`
: The loader on the target board does not indicate that it is ready. In other words, the first byte in the firmware communication record is not 80H. This is a fatal error.

Object Module Format Error Messages

This section lists the errors that can occur during object module processing.

`Bad Checksum`
: The downloader encountered an error while calculating the checksum for a record. This fatal error occurs during OMF86 processing.

`Bad Header`
: The file specified in the command line is not the correct file format. This fatal error occurs during OMF86 processing.

`Bad Magic Number In a.out Header`
: The *a.out* header has an incorrect magic number. The downloader supports a magic number of 10BH in the *a.out* header. This value indicates that the text and data segments are aligned within *a.out* so that they can be directly paged. This fatal error occurs during COFF386 processing.

`Bad Magic Number In File Header`

The file header has an incorrect magic number. The downloader supports a magic number of 14CH in the file header. The number indicates that the target machine contains an Intel386- or Intel486 microprocessor. This fatal error occurs during COFF386 processing.

`Bad PEDATA Record`

A PEDATA record format is incorrect. This fatal error occurs during OMF86 processing.

See also: PEDATA record format, *iAPX 86, 88 Family Utilities User's Guide*

`Bad PIDATA Record`

A PIDATA record format is incorrect. This fatal error occurs during OMF86 processing.

See also: PIDATA record format, *iAPX 86, 88 Family Utilities User's Guide*

`File Contains Fixup Records`

The file specified in the command line contains load-time locatable code. The file is not in absolute OMF86 format and the resulting loaded image may be incorrect. This non-fatal error occurs during OMF86 processing.

`File Contains Unresolved Externals`

The file specified in the command line contains unresolved external variables. The file is not in absolute OMF86 format and the resulting loaded image may be incorrect. This non-fatal error occurs during OMF86 processing.

`Invalid Module Attribute`

The file specified in the command line contains a module with an invalid attribute in the MODEND record. This non-fatal error occurs during OMF86 processing.

See also: MODEND record, *iAPX 86, 88 Family Utilities User's Guide*

`Load File Is Empty`

The file specified in the command line contains no data. This fatal error occurs during OMF86 processing.

`Load File Is Not Bootloadable`

The file specified in the command line is of the wrong type. This fatal error occurs during OMF286 and OMF386 processing.

`More Than One Start Address In File`

The file specified in the command line contains more than one main module with a start address. Only the last start address is used and the previous addresses are overwritten. This non-fatal error occurs during OMF86 processing.

`No Valid Start Address In File`

The file specified in the command line does not contain a start address. The file has been loaded. This fatal error occurs during OMF86, OMF286, and OMF386 processing.

`Non-Main Module Has Start Address`

The file specified in the command line contains a non-main module with a start address. This non-fatal error occurs during OMF86 processing.

`PIDATA Recursion Level Too High`

There are greater than seventeen nested levels within a PIDATA record. This fatal error occurs during OMF86 processing.

See also: PIDATA record format, *iAPX 86, 88 Family Utilities User's Guide*

`Premature EOF Encountered`

The file specified in the command line is incomplete. This fatal error occurs during OMF86, OMF286, OMF386, and COFF386 processing.

`Record Too Large`

The default buffer size is too small for the file to be downloaded. This fatal error occurs during OMF86, OMF286, and OMF386 processing.

❑❑❑

ATCS/279/ARC/450 System Jobs C

ATCS/279/ARC Server Job

The ATCS/279/ARC job provides a Multibus II system with either a windowed terminal (ATCS/279), or a multiplexed output from hosts and monitors onto a single terminal (ARC). The DUIB entries in the *:config:terminals* file and the system hardware determine the display environment.

You can display the MSA firmware diagnostics on either a serial console (ARC) or graphics console (279), depending on the configuration of the MSA firmware. If the SBX scan bit is not set in the MSA firmware, the serial console is used.

See also: SBX scan bit, *MSA Firmware User's Guide*

The ATCS/279/ARC Server communicates with the ATCS driver running on other CPU boards using the ATCS protocol and to monitor/debuggers using the Remote Console Interface (RCI) protocol.

When you have multiple CPUs in a Multibus II system, there are several options for providing HI terminals to each of the CPU boards:

- Provide a separate SBX 279A window to each of the CPU boards by using the SBX 279A and the ATCS/279/ARC Server.
- Provide one physical terminal to the I/O server board in slot 0 and use the ATCS/279/ARC Server.
- Provide a separate physical terminal to each CPU board. This requires using the serial ports on a terminal controller board and separate terminals for each CPU board.

Note

If the ATCS/ARC Server is configured for the `t82530_0` device (no SBX 279), tasks at priority 255 do not execute. The workaround is to make sure that all application tasks execute at priority 254 or higher (numerically lower).

Separate SBX 279A Window

The first option requires only a single graphics monitor, keyboard, mouse, and cables. It multiplexes the HI terminals from multiple CPU boards to a single display device. Using separate windows, you can access each of the CPU boards by moving from one window to the next using a mouse.

See also: SBX 279A windows, *Command Reference*

The 279 portion of the Server provides three types of windows: one for clients, a second for debug monitors that require single character input and output, and a third for graphics from remote clients. The windows that support clients and debug monitors emulate a CRT terminal. The portion of the ATCS/279 server job that implements debug windows is called the Remote Console Interface (RCI) Server.

A windowed environment for debug monitors enables multiple windows on a single graphic monitor display. Each window shows debug information from a different processor.

One Physical Terminal to I/O Server Board

The second option requires only a single terminal. The ARC portion of the Server does not provide windowing, but instead buffers information from each host OS (ATCS) and host monitor (RCI) line connected to it. Only the output from a single OS (ATCS) or monitor (RCI) line is displayed on the terminal at one time. This information is left on the screen until it scrolls off or is specifically cleared. You can choose which host is connected so that its output can be displayed. Monitor information takes precedence over OS information and is displayed from any host when it is available.

Separate Physical Terminal to CPU Board

The fourth option is the most flexible because it provides a simultaneous full screen display and keyboard input for all CPU boards. However, it also requires multiple cables and terminals.

Configuring the ACTS/279/ARC Server Job

You can load the *atcs279.job* with one configurable parameter, the name of the default terminal device.

See also: *atcs279.job*, Chapter 4

If you add the job as a first-level job in the ICU, configure it on the ATCJ screen of the ICU. The Maximum System Windows (MSW) and Maximum Debug Windows (MDW) options are not used by the ARC server. The ARC server allocates one line per host for OS console I/O and one line per host for monitor console I/O. When queried by the CCI GET SERVER INFO command, the ARC Server replies that it has 21 lines available. Only one line can be used by each client host.

See also: ATCJ screen, *ICU User's Guide and Quick Reference*

You may configure the number of system windows (MSW) for the ATCS/279 Server. Each system window requires a unique DUIB in the SBX 279A driver that you must configure if you include the ATCS/279 server job. These DUIBs must be named `h279_0`, `h279_1` and so on. The ending digit of each name corresponds to the ATCS line number. Other host processors can access the system windows by using the ATCS driver. When `0`, `1`, etc. are used to access system windows, the numbers represent ATCS lines. For the Maximum Debug Windows (MDW) option, the RCI server supports one debug window per host. The typical client of the RCI server is the debug monitor.

Choosing the ATCS/279 or ARC Console

The DUIBs you include in your system, along with the boards or ports actually connected, determine which mode of the ATCS/279/ARC job is used. The mode is chosen according to the flowchart in Figure C-1.

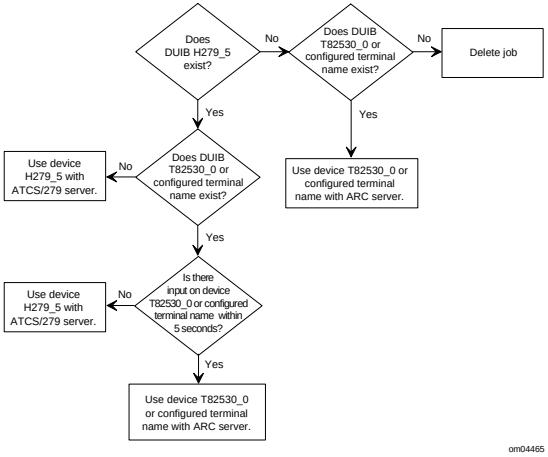


Figure C-1. Choosing ATCS/279/ARC Mode

The terminal DUIB must be configured to point to a locally controlled serial port, as on an 82530 component. By default it points to Unit 0 of the on-board 82530. The device must be available when the BIOS is being initialized.

Select the appropriate Console Controller driver for your console on the SDM screen of the ICU.

The ATCS/279/ARC Server supports short-circuit operations when using the ARC portion of the Server. The 279 portion of the Server does not support short-circuit operation when connected to the SBX 279A board. This means that a board cannot host both the ATCS Driver and ATCS/279/ARC Server and use ATCS Driver physical device names to access SBX 279A windows controlled by the Server.

Mapping SBX 279A Windows to Device Names

Table C-1 outlines the mapping of SBX 279A windows to device names as implemented in the standard Multibus II definition files.

Table C-1. Mapping of the SBX 279A DUIB Names

SBX 279 Unit	Local 279 DUIB Name on I/O Server	Local 279 DUIB Alias for Debug Window	Local 279 DUIB Alias for System Window	ATCS Driver DUIB name on Remote Client CPU
0	t279_0*			
1	t279_1			
2	t279_2		h279_0	t279_0*
3	t279_3		h279_1	t279_1
4	t279_4		h279_2	t279_2
5	t279_5		h279_3	t279_3
6	t279_6		h279_4	t279_4
7	t279_7	m279_0		
8	t279_8	m279_1		
9	t279_9	m279_2		
10	t279_10	m279_3		
11	t279_11	m279_4		
12	g279_0		h279_5**	g279_0
13	g279_1		h279_6	g279_1
14	g279_2		h279_7	g279_2
15	g279_3			

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

* Note the correlation between the ATCS driver configuration on the ATCS client CPUs and the SBX 279A configuration on the ATCS/279 I/O Server.

** Required for server initialization.

In the definition file for the I/O Server (such as an SBC 386/258 or 486/133SE board) you can delete any physical device names that your application does not require.

Since the ATCS/279/ARC Job does not support short-circuit windowed (279) operations, the terminal names you specify in the *:config:terminals* file must match your Multibus II system configuration. In the *:config:terminals* file for a board which hosts the ATCS/279/ARC server, an SBX 279A driver, and the ATCS driver (I/O Server definition files do this) and where you want to use a 279 window, specify the device name `t279_0` for the SBX 279A driver. The *:config:default/terminals.279* file is an example of this use. In the *:config:terminals* file for a board which only hosts the ATCS driver or uses the ARC portion of the server, specify the device name `atcs_con_0`. The *:config:default/terminals.arc* file is an example of this use.

In Table C-1, the column labeled **Local 279 DUIB Name on I/O Server** lists the physical device name used by the 279 driver to refer to its windows when the SBX 279A module is mounted on the I/O Server board. Similar names may be used if the SBX 279A module is mounted on a different CPU board. There are 12 terminal emulation windows (they start with `t`) and four graphics windows (they start with `g`) in the I/O Server definition files. Some of these windows are shared with remote hosts which can access the windows using the ATCS/279 server. Either a remote host or the local host can use a window, but only one host at a time can access a given window. Only unit numbers 2 through 14 are accessible from the remote hosts.

The column labeled **Local 279 DUIB Alias for Debug Windows** lists the DUIB names used locally by the RCI Server running on the I/O Server board. The RCI Server provides Debug Windows for the SDM monitor running on CPU boards. If there are more CPU boards than the configured number of Debug Windows, the SDM console on the remaining boards is the on-board serial port of the CPU board.

The column labeled **Local 279 Alias for System Windows** lists the DUIB names used locally by the ATCS job running on the I/O Server board. These are the local names of System Windows for CPU boards in the system. The standard I/O Server board definition files provide eight such windows. Five are for terminal emulation and three are for graphics. The digit following the underscore (h279_*n*) refers to the ATCS line number. The ATCS driver on the CPU board uses the ATCS line number to access one of these windows. Only one CPU board at a time can use an ATCS window, but one board may use multiple windows at the same time.

The column labeled **ATCS Driver DUIB Names on Remote Client CPU** lists the physical device names used by the ATCS drivers on CPU boards to access remote

windows. The ATCS driver communicates with the ATCS/279/ARC Server, which provides the appropriate 279 window based on the ATCS line number. The remote window appears to the ATCS driver as an ATCS serial line. There are eight ATCS DUIBs for remote windows in the standard Multibus II CPU definition files. Five of these are for terminal emulation and start with `t279`; three are for graphics and start with `g279`. By default, the ATCS driver is configured to search for a 386/258 I/O Server. You can change this parameter in the BPS file to point to another board.

Adding Remote Windows

Each ATCS serial line corresponds to a separate SBX 279A terminal or graphics window. As shown in Table C-1, some 279A DUIBs are configured as System Windows called `h279_`*n*, where *n* corresponds to the ATCS line number.

You can use 279A windows for the SDM monitor running on the client slots. These windows are called Debug Windows and are designated `m279_`*n* (assuming *n*+1 Debug Windows). Each client host needs a separate Debug Window. The part of the ATCS/279/ARC Server that provides Debug Window services is called the Remote Console Interface (RCI) Server. It allocates windows on a first-come-first-served basis. The SDM monitor on the client boards communicate with the RCI Server via Multibus II transport and Multibus II interconnect space.

The standard definition file for the ICU provides eight System Windows. Five of these windows are terminal windows and three are graphics windows. Five Debug Windows are also supplied. If you add more hosts to your Multibus II system, the default number of remote windows may not be adequate.

In these examples you use the ICU to modify the standard definition files to provide additional System and Debug windows.

Adding a System Window

This example assumes that the ATCS/279/ARC Server runs on an I/O Server board with the SBX module mounted on that board. It also assumes that the client iRMX boards are CPU boards.

1. In the definition file for the I/O Server board, add a unit (window) to the 279A driver configuration:
   - Add one more UINFO table. This is a UT279 screen for a normal window, or a UG279 screen for a graphics window. Copy one of the existing screens as an example for the parameter values.
   - Add one more DUIB, or I279 screen. For example, copy the I279 screen with the NAM value `h279_7`. Change the NAM parameter to `h279_8` (because this is the ninth window) and set the UNB parameter to an unused unit number for this device. Make sure the new unit does not have the

highest unit number among those configured. There must always be a dummy DUIB with the highest unit number configured as a Dynamic window (in the TYP option of the UT279 or UG279 screen). Typically this is the `g279_3` DUIB name listed in Table C-1. Change the unit number of this dummy DUIB to be the highest.

2. Increment the number of System Windows in the ATCS/279/ARC System Job configuration to 9. This is the MSW option on the ATCJ screen.
3. Add a unit (ATCS line) to the ATCS driver configuration in the ICU definition file for the client CPU board:
   - Add a UINFO table (U410 screen) and a DUIB (I410 screen) for the ATCS_CON device. Copy and modify one each of the existing screens. Specify ATCS line 8: on the U410 screen, change the NAM parameter to UINFO_ATCS_CON_8 and on the I410 screen, change the UN parameter to 08H and the UIN parameter to UINFO_ATCS_CON_8. The ATCS/279/ARC Server maps this to the `h279_8` DUIB that you added in step 1.
   - If you are adding a graphics window (device name `g279_`*n*), set the following ICU options as shown. These appear on the U410 screen:

Configuration Option	Parameter Value
(LEM) Line Edit Mode	trans
(ECH) Echo Mode	NO
(OCC) Output Control in Input	NO
(OSC) OSC Controls	NO

4. Regenerate the systems for both the I/O Server board and the CPU board.

Adding a Debug Window

This example assumes that the ATCS/279/ARC Server runs on an I/O Server board and that the SBX 279A module is mounted on that board.

1. In the definition file for the I/O Server board, add a unit (window) to the 279A driver configuration:
   - Add a UINFO table or UT279 screen. Copy and modify one of the existing screens.
   - Add a DUIB or I279 screen. Set the NAM value to `m279_5`, for the sixth Debug window. Set the UNB parameter to an unused unit number for this device, making sure the new unit does not have the highest unit number among those configured. (See the previous example.)

2. Increment the number of Debug Windows to 6. This is the MDW option on the ATCJ screen.

3. Regenerate the system image using the ICU G(enerate) command.

ARC Server

The ARC Server initializes and displays its query at the system console before the HI initializes. If there is no operator response to the query, the console displays the HI logon prompt.

The first OS or monitor that makes an input request gets control of the console. When connected, all user input goes to that host. The ARC server lets you specify and use a hot key to connect to any host. By default, the hot key is <Break> for a local terminal and <Ctrl-Y> if you use a modem.

If the OS (ATCS client) is active when you switch the console to a new host, any buffered ATCS output from that host is displayed and input is sent to the OS. If the monitor (RCI client) is active, input is sent to the monitor.

When monitor output becomes available from any client host, the monitor output will interrupt output from an OS. If multiple monitors are trying to do output at the same time, all output from the first active monitor is completed. Then output from the next client host's monitor is displayed until it has completed. This is repeated until no monitor output is available. At that time, the interrupted OS's output is resumed.

When switching monitor output, the ARC server displays a header showing which host is displaying information. The header scrolls off the screen as more information is displayed. The SDM monitor prompt always identifies the host it is running on.

The ARC Server provides a menu for switching between clients using the same physical serial I/O device. You invoke the ARC menu with the hot key; the menu is also displayed for five seconds during server initialization (see Figure C-1). This enables you to connect to a specific host or to change the hot key before being required to use it. Where the SBX 279A module is present, the server sign-on menu is:

```
              MULTIBUS II CONSOLE SERVER INITIALIZATION

*****************************************************************
**                                                             **
** ENTER <CR> WITHIN 5 SECONDS for this to be the SYSTEM CONSOLE **
**                                                             **
*****************************************************************

*****************************************************************
```

```
**                                                                  **
** Terminal Setting:  9600 BAUD, 8 BITS, 1 STOP BIT, NO PARITY     **
**                                                                  **
**********************************************************************
```

If no SBX 279A module is present, the first message is slightly different:

```
ENTER <CR> WITHIN 5 SECONDS to change the HOT KEY
```

You cannot use the HI until a client host has made an input request. The first client host which makes a connection to an ATCS line and requests input becomes the active client. This could be either an OS or a monitor client.

When you press the hot key, the HI displays this prompt:

```
Enter <0-20> to Select Client Host OR <CR> to Display Menu
```

The ARC menu enables you to select the client host, select the hot key, and/or to print a separator between RCI output from different client hosts. The menu is displayed if you press <CR> after the initialization prompt or after the host selection prompt. As shown in this example, the connected client host's slot ID is enclosed in parentheses at the top of the menu.

```
MULTIBUS II CONSOLE SERVER MENU
 - Console Client Hosts in Slots:  #xx  (#xx)  #xx
 - Hot Key:  BREAK
 - Output Mode:  SEPARATED
 - Commands:
    H      - Help
    K      - Select New Hot Key
    M      - Set Monitor/Debugger Output Mode
    <0-20> - Select Client Host
    <CR>   - Exit
Enter Choice -
```

If you select the **H** command, a help message is displayed. For the **K** command, this menu and prompt are displayed.

```
Select HOT KEY -
  1    - ~
  2    - CTRL Y
  3    - BREAK (default)
  <CR> - Exit
Enter Choice -
```

For the **M** command, this prompt is displayed.

```
Separate Output from different Monitor/Debugger Client Hosts [Y/N]
?
Enter Choice -
```

If you enter 0 through 20 at the main menu, input is sent to the selected client host, if present. If the client host ID is not valid or the client host is not present, the server displays this error message.

```
*** Warning:  Invalid Client Host, Active Host Not Changed
```

If invalid input is given to the server at any other time, the server displays this error message.

```
*** Warning:  Invalid Entry
```

ATCS/450 Job

The ATCS/450 job provides access to the serial channels on the MPI 450 and the MIX 450 boards. Each of these boards has 12 serial channels. A single ATCS/450 Server can control up to 36 serial channels.

The ATCS/450 job shields clients from the low level hardware interfaces required to control the serial channels on the MIX 450 modules and MPI 450 boards. It provides a high level interface to the client OS's ATCS protocol. This job can support multiple clients and can be configured in multiple ways. Figure C-2 illustrates example configurations.

The MIX 450 module must be mounted on a MIX baseboard, such as the 386/020(A) or 486/020A. Up to three MIX 450 modules can be mounted on a single baseboard.

The MPI 450 is a non-intelligent I/O board residing on the parallel system bus (PSB). The ATCS/450 server communicates with MPI 450 boards using the PSB I/O space. Each MPI 450 board has 12 serial channels. The MPI 450 board can be controlled by an ATCS/450 server running on a MIX 386/020 board, the 186/450 board, or any CPU board hosting the ATCS/450 server.

It is possible for the ATCS driver and the ATCS/450 server to reside on the same host and communicate with each other using short-circuit messages, as shown in Figure C-2. The ATCS/450 server job can take a significant part of CPU bandwidth when configured with other system jobs.

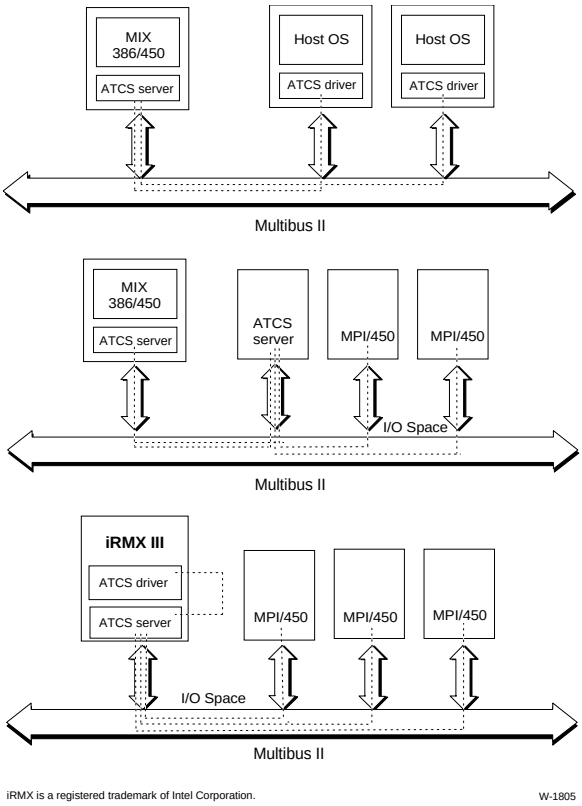


Figure C-2. ATCS Job

ATCS/450 Configurations

A single ATCS/450 server job can control up to a maximum of three MIX 450 modules and MPI 450 boards simultaneously. Use this algorithm to assign ATCS line numbers to serial channels on these boards:

1. Assign line numbers 0 through 11 to serial channels on the first MIX 450 board, numbers 12 through 23 to serial channels on the second MIX 450 board, and numbers 24 through 35 to channels on the third MIX 450 board. The ATCS job searches for MIX 450 modules on the same baseboard.

2. If there are fewer than three MIX 450 modules on the same baseboard, assign the remaining line numbers to serial channels on any MPI 450 boards in the system. The ATCS job scans for MPI 450 boards in slots higher than the slot on which it resides until it finds a non-MPI board or finds a total of 36 serial channels

(including any MIX 450s residing on MPI 450s). The serial channels on MIX 450s are assigned lower ATCS line numbers than channels on MPI 450s.

Using the guidelines above, the following configurations are possible. In each case, the ATCS driver could run on the same board as the ATCS/450 server job and/or on other CPU boards in the system:

- The ATCS/450 server job runs on the MIX baseboard and controls MIX 450 modules mounted on the baseboard.
- The ATCS/450 server job runs on the MIX baseboard and controls MIX 450 modules and MPI 450 boards.
- The ATCS/450 server job runs on a CPU board and controls MPI 450 boards.

Depending on the number of ATCS lines you use, and whether you include an ATCS driver or both the driver and the server, you may need to change some ICU options in the Nucleus Communication Service (NCOM) screen. The affected values are the Maximum Number of Simultaneous Messages (MSM) and Maximum Number of Simultaneous Transactions (MST) parameters. Additional demands may be put on these resources by the PCI server and driver or by your message-passing application. ATCS requirements are shown below.

Parameter	ATCS/450 Server Requirement	ATCS Driver Requirement
MSM	16 * (total number of ATCS lines + 1)	15 * total number of ATCS lines
MST		6 * total number of ATCS lines

❑ ❑ ❑