JNIOR
Software
Description

JNIOR Software Description	Revision 0.1	May 24, 2001
INTEG process group, inc.		Draft

Table of Contents:

List of Tables:

Description

The Java Network I/O Resource (JNIOR) is a low-cost digital and analog interface that runs the Java software language and communicates over computer networks using TCP/IP. This document describes the basic operating system software (JNIOROS) that will ship loaded and ready to run on the JNIOR product.

JNIOR is a flexible embedded core that is rich in hardware and software features. The Java Virtual Machine (JVM) and TCP/IP connectivity are expected to be cornerstone features of JNIOR, which allow it to be easily ported into a myriad of web-enabled device and system applications. The object-oriented Java programming language is ideal for developing modular classes that can be reused to minimize the cost of leveraging code into custom applications.

One of Java's most powerful features is portability. Java Virtual Machines are available for most popular CPU platforms and operating systems. JNIOR's core software and hardware can be ported into any platform that supports a JVM, from the smallest embedded systems to servers running multiple processors. Electronic hardware features are a major cost driver in any embedded system and the added cost of porting software drivers can prohibit entry into low volume applications. The greatest benefit in JNIOR is the ability to port and extend the hardware and software features into different footprints without the need to rewrite low drivers and interfaces. The end result is a lowered development cost and the potential to open new markets. There are currently several microprocessor solutions available that have integral JVM support including the Tiny Internet Interface (TINI) from Dallas Semiconductor.

JNIOR is an entry point product for INTEG process group, inc. that allows the Company to market its goods and services into the web-enabled device and system application markets. JNIOR leverages the Dallas Semiconductor TINI in a simple low cost device that server socket-connects I/O points across TCP/IP networks.

The JNIOROS software is a combination of Java applications that run on JNIOR's embedded JVM. JNIOROS is meant to provide basic network connectivity and configuration of JNIOR's I/O points. JNIOROS is a starting point operating system that can be extended in the future to provide solutions to industry specific applications.

The Internet has caused growth and interest in web-enabled devices and factory floor Ethernet. Remote connectivity solutions used to be limited to expensive leased-lines, proprietary wireless links, and dedicated equipment that all contributed to a high entry and maintenance cost. Enterprise wide networks, Local Area Networks (LAN), and broadband Internet access has all shrunk the technology gap once associated with remote access. The widespread acceptance of Java for application software distributed over TCP/IP and low cost Ethernet hardware has further shrunk the cost of developing web-enabled solutions. It is anticipated that INTEG process group, inc. can capitalize on the ability to produce JNIOR as a cost effective stand-alone product, develop products that act as turn-key application specific solutions, and create a flexible embedded core that can be leveraged into custom applications on a contract basis.

Scope

The purpose of the JNIOR Software Description is to document the software functionality and architecture of the JNIOR product. The JNIOR feature set, functionality, and ultimate implementation described herein are subject to debate and change.

The JNIOR development project is on a fast-track schedule and there is little or no time available for a committee style design or a risky implementation. The proposed software architecture attempts to leverage as much of the open source code available for the TINI product and makes liberal use of techniques known to INTEG process group, inc. With these goals and in mind, the author respectfully asks that any proposed diversions be for the good of the schedule or to correct a flaw that will affect the overall performance of JNIOR.

Goals

JNIOR was developed at the INTEG process group, inc. to extend their product offering into web enabled devices, OEM product development, and turnkey system development.
JNIOR is an entry point product for INTEG process group, inc. that will be market over its transport medium, the Internet. JNIOR is being marketed as a general-purpose I/O box that can be configured by the end user for simple remote control and monitoring applications and leveraged by INTEG into custom end-to-end application solutions.

The objective of the JNIOR Software Description is to capture the requirements, features, and implementation of what will be distributed as the JNIOR operating system (JNIOROS). The goal is to develop a quick turn application that demonstrates the usability and allows INTEG to market the JNIOR box to customers. It is expected that some portion of the classes developed for the JNIOROS will be portable into custom applications and reused in future upgrades or extensions. The overall goals of the JNIOROS development are as follows:

- Demonstrate the remote I/O connectivity of JNIOR over TCP/IP networks
    - Integral TCP/IP servers
    - MMI based on Java applets embedded in html web pages
- User configurable system and I/O points parameters
    - System network parameters
    - I/O point name
    - Range, scale, and unit translation of analog I/O points
    - Logic inversion of digital I/O points
- I/O point control
    - Manual web-browser “point and click” control,
    - Flat file over ftp or socket
    - Remote TCP socket for streaming Applets and dedicated applications
    - Hook(s) for conditional based outputs
- Useful but limited functionality
    - Maximize device speed and apparent data throughput
    - Need to leave functional upgrade path for custom application development
- Modular software design
    - Object oriented design methodology
    - Multi-threaded OS
    - Reusable GUIs for control/monitor and configuration Applets
    - Configurable I/O populations

    - Design for manufacturability, testability, and remote software upgrades
- Develop on an ASAP schedule
    - Required for sale of JNIOR box to generate revenue stream
    - JNIOROS useable for live on-line web demonstration systems

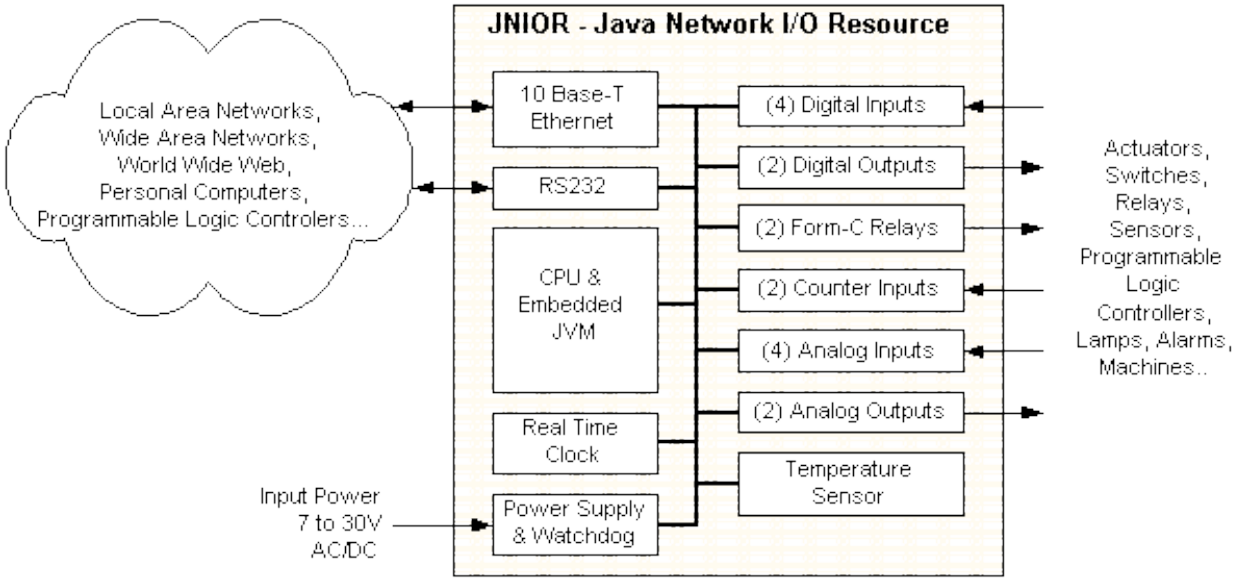
JNIOR Hardware

JNIOR is based on the Dallas Semiconductor DSTINI1 evaluation board, also known as the TINI. The TINI board is a highly integrated assembly that includes a Dallas' DS80C390 microcontroller, FLASH and SRAM memory, various I/O, and a runtime operating system with and an integral JVM. The features inherent in the DSTINI1 assembly are as follows:

- High-Speed DS80C390 microcontroller with embedded JVM
- Compact (31.8 mm x 102.9 mm) 72-pin SIMM board.
- Dual serial ports
- Dual 1-Wire network interfaces
- Dual CAN (Controller Area Network) controllers
- 2-wire synchronous serial bus
- General-purpose digital I/O
- 8051 style high speed processor bus interface
- Real Time clock/calendar for time stamping
- Flash ROM for storage and execution of runtime environment
- 512K/1MByte nonvolatile SRAM
- Level translator provides RS232 voltages on serial port 0
- Wide operating temperature range, -20C to +70C
- Requires +5V power supply

JNIOR is a practical implementation of a TINI board into an embedded system product targeted for general purpose I/O connectivity. JNIOR places the TINI board in a protective case with integral power supply and a variety of general-purpose analog and digital I/O points. External communication is handled via the 10Base-T Ethernet or RS232 serial ports. The following sections detail JNIOR's hardware features and I/O resources. Refer to the JNIOR block diagram in Figure 1.

Figure 1: JNIOR Hardware Block Diagram



CPU & Embedded JVM

JNIOR is based on the Dallas Semiconductor DSTINI1 evaluation circuit board assembly. The JVM core is a Java 1 compliant implementation that runs Java byte-code compiled for the TINI. The CPU core also includes the necessary memory and I/O interfaces required to operate and communicate with the JVM.

Communication Ports

JNIOR has integral 10Base-T Ethernet and RS232 ports used for local and networked TCP/IP communication. The RS232 port is fully programmable via the JVM, which makes it useful for custom interfacing to legacy equipment that lacks TCP/IP connectivity. The RS232 interface is also used to configure the JNIOR system, network parameters, and load the TINI's FLASH memory with application programs.

10Base-T Ethernet

The JNIOR 10Base-T Ethernet port has the following attributes:

- TCP/IP stack network interface
- Provides network connectivity for TCP/IP servers
- JVM configurable interface (inconfig)
    - IP address, netmask, gateway
    - Sendmail client support
    - DNS
    - DHCP and TFTP boot
- RJ45 UTP connector accepts low cost CAT5 wiring

RS232 Serial Port

The JNIOR RS232 Serial port has the following attributes:

- Fully configurable asynchronous protocol

- Fully programmable as a JVM resource
- Used to configure systems that are not network ready
- Used to program binary Java application files into TINI's onboard FLASH memory
- DB9F connector compatible with PC COM ports using a null modem cable
- Ideal for network enabling RS232 style legacy equipment
- Expandable JVM support for two more serial ports via an external 16450 UART device

1-Wire Interface Port

(Note: We need to consider adding this to the current JNIOR design.)
The JNIOR 1-Wire RS232 Serial port has the following attributes:

- *1-Wire device network expansion interface*
- *RJ11 connector compatible with Dallas Semiconductor and 3rd party 1-Wire products*

Analog I/O Points

JNIOR has the following analog I/O points:

- (4) Analog voltage Inputs – software configurable for 8 to 16 bit resolution and 5 or 10V full scale.
- (2) Analog voltage Outputs – 0 to 10V outputs
- (1) Internal temperature sensor – reads the temperature of the JNIOR PCB assembly

Digital I/O Points

JNIOR has the following digital I/O points:

- (4) Optically isolated digital inputs – 5 to 30V AC or DC input
- (2) Optically isolated digital outputs – 5 to 30V DC outputs
- (2) Optically isolated counter inputs – 32 bit counter accepts 5 to 30 V AC or DC input
- (2) Form-C SPDT relay outputs – 1 Amp 32VDC contacts

Real Time Clock

The Real Time Clock (RTC) is integral to the DSTINI1 circuit board assembly. The RTC is used by the JVM as a time and date resource. Java applications can use the RTC resource for event time stamping and automated processing of timed applications.

Power Supply & Watchdog

JNIOR has integral DC power converters and a CPU watchdog circuit. The JNIOR assembly requires 7 to 35 Volts AC or DC input power that is converted to the regulated +5 and +12 VDC voltages necessary to operate the internal analog and digital circuits.

The JNIOR watchdog timer is used to monitor the health of the operating system software and issue a hard CPU reset in the event of a software crash. The watchdog device requires a periodic "kick" from one of three jumper selectable sources to hold off a reboot. The required watchdog kick sources are the 1WIO, A0, and P3.3 pins that come from the TINI board.

TINI Software

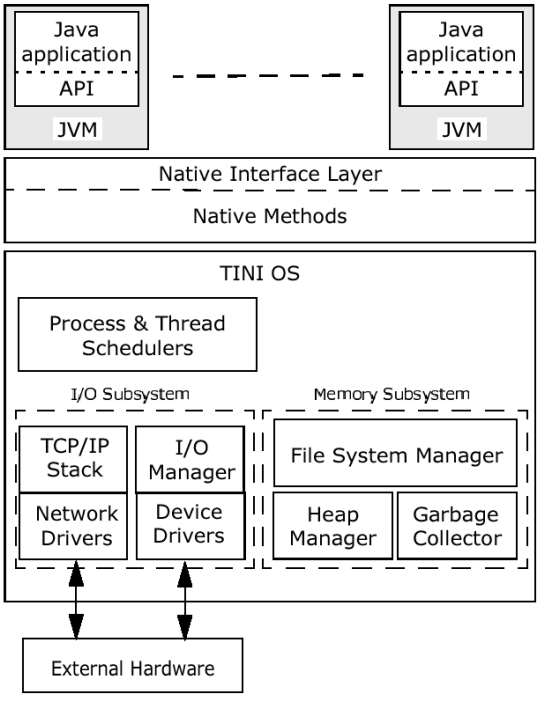
The following sections details the TINI software architecture and operating system provided by Dallas Semiconductor. (Cut-&-pasted directly from the Dallas Semiconductor web site without anyone's permission.)

TINI Runtime Environment

Providing hardware essential for developing embedded network devices is only half of the job. A large amount of software is also required to free application developers from having to worry about the details of creating layers of infrastructure to provide support for executing multiple tasks, network protocol stacks and an application-programming interface. A well-defined runtime environment that provides all of these features allows the developer to focus primarily on the details of the application. For this reason a runtime environment was developed from the beginning as an integral part of the overall platform.

The software that comprises TINI's runtime environment can be divided into two categories: native code executed directly by the microcontroller and an API interpreted as bytecodes by the Java™ Virtual Machine. Application code is written in Java and utilizes the API to exploit the capabilities of the native runtime and the underlying hardware resources. It is also possible to write native libraries that can be loaded from within an application, for the purpose of meeting strict real-time requirements. A graphical representation of the runtime environment is shown in Figure 1.

Figure 2 TINI Runtime Environment



Java VM

The memory footprint of TINI's Java Virtual Machine (JVM) is less than 40 kilobytes. Despite its small size it supports much of the functionality provided by full JVM implementations, including: full support for threads, support for all primitive types, and strings. The JVM provides access to the following core Java packages: java.lang, java.io, java.net and java.util. TINI specific classes are exposed through the com.dalsemi.* packages. These packages allow access to the TINI hardware layer and resources.

TINI OS

TINI OS is very small and provides basic services such as task scheduling, a file system, memory and I/O managers. Unlike most small, embedded-controller operating systems, TINI OS is designed to switch heavyweight tasks. Specifically, it is optimized to switch between multiple executing instances of a Java bytecode interpreter. This provides the foundation required for running multiple Java applications. The task scheduler is a simple round robin scheduler, which

provides constant 8-ms time slices. With the exception of the garbage collector, all tasks driven by the OS are Java applications. Multiple native/kernel processes are managed through cooperative multitasking and are therefore subject to tight execution time requirements.

TINI SDK

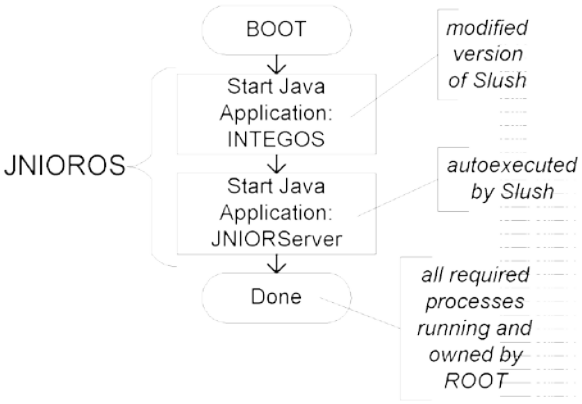
The runtime environment is contained within the TINI Software Developers Kit (SDK). The SDK also provides several additional utilities for developing TINI applications. Download the latest SDK. INTEG process group, inc. is currently keeping downloads of the TINI SDK in the company's INTEG server under the java directory.

JNIOR Software Architecture

JNIOROS is a starting point for INTEG's software development on the JNIOR hardware platform. JNIOROS is a combination of Java applications that function together as JNIOR's run-time operating system. JNIOR is expected to attract attention from a wide range of potential end users. Some will require an easy to configure device, and others will wish to write their own custom Java applications. The ideal JNIOR software architecture will make provisions for both scenarios without compromising performance or reliability.

The JNIOROS software is located in the TINI's binary FLASH file system and needs to be protected from accidental or malicious erasure. The TINI software architecture is flexible and supports a variety of boot and Java execution methods, each with their own caveats. The prime requirement for JNIOROS: ***get to market ASAP with a reliable product***; hence JNIOROS architecture will take the path of least resistance.

Figure 3 JNIOROS Boot Sequence



The Slush operating system supplied by Dallas Semiconductor is an Unix-like OS that supports multiple user accounts. Slush is a command line interpreted OS that can be entered via the RS232 or Ethernet ports. The super user (or ROOT user) has full administration privileges and can start or kill Java processes, create and remove files anyone's files or directories. Any additional users with Slush login accounts have limited system privileges and no rights to certain commands that could adversely affect the processes running on the JVM. For example, a user cannot kill a running process, remove files owned by ROOT, or add more user accounts. The login account for ROOT and any other user is protected by a hidden password that is encrypted and held in the TINI's file system.

One possible solution to the JNIOROS software architecture is too boot directly into a single Java application that does not permit the spawning of any additional processes. This scenario would closely mimic the behavior of most dedicated embedded systems and require a single boot image for each and every possible application. Although this provides for the most "bozo-proof" implementation it seriously limits the potential for savvy end users, and even INTEG, to hook into the boot process. The boot-one-image solution also would require the added software effort

to integrate the functionality of Slush into the same Java application that runs JNIOR. The added software effort and lack of flexibility make this solution undesirable.

A more attractive implementation splits JNIOROS into two components, one a slightly modified version of Slush, the other a custom Java application that contains JNIOR's functionality. Figure 1 at the right illustrates the JNIOROS boot sequence. The JNIOROS would be executed at boot and owned by the ROOT user. This ensures that any additional users cannot affect the integrity of the system.

Running the JNIOROS Java application "like any old" TINI program requires that the end user be prohibited from ever logging into the system as ROOT, hence we won't tell them the password. A ROOT user could potentially screw up the operation of their JNIOR product or even delete the programs making the box useless. It is expected that INTEG will need to develop a standalone PC application that can be used to load programs and configure an IP'less JNIOR. INTEG can selectively reveal JNIOR's ROOT password to end users that wish to write their own customized Java applications. (It should be noted that the warranty issues related to someone blowing away operating system would need to be addressed before permitting anyone to log into JNIOR with ROOT level permission.)

INTEGOS Software Description:

The Java application INTEGOS is a slightly modified version of the Dallas Semiconductor Slush operating system provided with the TINI. INTEGOS is the Java application initially executed when JNIOR boots. When INTEGOS has completed execution and parsed the ***.startup*** file, it calls the JNIORServer application that contains JNIOR's functionality. The following sections describe Slush and the INTEGOS derivative.

Slush Operating System

Slush is the TINI's operating system that is provided by Dallas Semiconductor as part of the basic software development tool package. Slush is a small system shell intended to provide a Unix like interface with serial, Telnet, and FTP servers. Slush provides users with a way to view and manipulate the TINI's FLASH file system, as well as control system functions like the watchdog timer and network interface configuration.

Slush is a multi-threaded, multi-user system allowing simultaneous login sessions. Slush uses a username/password combination to authenticate a login request. Users can be added or removed from the system by a user with super-user privileges like ROOT, or any user who is a member of the group "admin".

If a user has a home directory then they will be placed there at login. If Slush finds a `.login` file located in the user's home directory the commands will be automatically executed as if they were typed on the command line. A "#" at the beginning of a line denotes that the remaining text is a comment. This functionality is very similar to Unix.

The Slush Java application is located in bank 7 of the TINI's FLASH memory. Any Java application located in bank 7 is automatically executed when the TINI warm or cold boots. Slush contains several system files and environment variables used to configure and control the TINI. The environment variable details change with each release of the TINI API.

Slush OS System Files

Slush automatically creates several system files that are located in the `/etc` directory:

```
drwxr-x   1 root    admin     6 Jun 06:34    .
drwxr-x   1 root    admin     6 Jun 06:34    ..
-rwxr--   1 root    admin     6 Jun 06:34    .tininet
-rwxr--   1 root    admin     6 Jun 06:34    passwd
-rwxr--   1 root    admin     6 Jun 06:34    .log
-rwxr--   1 root    admin     6 Jun 06:34    .startup
```

The `.tininet` file contains TINI's network host and domain names. The default hostname is TINI. There is no default domain name.

Slush maintains the login user account information in the `passwd` file. This includes the user's name, hash of their password, and user ID. Slush creates two default user login accounts for "root" and "guest" with the passwords "tini" and "guest" respectively.

The file `.log`, is an automatic system message log file that is enabled or disabled by the `genlog` Slush command. The log file is automatically appended with system messages that report the health and status Slush and any other shell processes.

The `.startup` file is used by Slush to automatically execute commands as if they were coming from a Slush command prompt. The `.startup` file is generally used to initialize system environment variables and resources, setup the Slush software watchdog, and execute additional Java applications.

Slush Serial Server

The serial server provides a means to login into Slush via the RS232 serial port (TTY). Slush defaults to serial0 at 115.2 Kbaud and can only be changed by editing the constants in the Slush source code and creating a new build. There are a number of environment variables that allow the serial server to be customized. The details are located in the Slush readme file.

Note: The Serial0 RS232 level shifted interface is TINI's COM port. The standard TINI product uses Serial0 as the default interface for the Slush server and JavaKit, the FLASH boot program supplied by Dallas Semiconductor. JavaKit is a Java application that runs on a PC and is used to download program files into TINI's FLASH. JavaKit is generally used to download the TINI Java API and Slush.

Telnet Server

The Telnet server provides a means to login into Slush over the Ethernet port. This makes a network connected TINI accessible for maintenance via a desktop Telnet client. The functionality is very similar to a remote UNIX account login. The Telnet server defaults to port 23. There are a number of environment variables that allow the serial server to be customized. The details are located in the Slush readme file.

FTP Server

The FTP server provides a means to transfer files to and from the TINI's FLASH file system. The functionality is very similar to any UNIX style remote login session. Users can connect to TINI FTP server using desktop PC applications like WSFTP, CuteFTP, or "ftp" from the

command line of most operating systems. Users are automatically placed in their respective home directory (/user) if one exists. Account permissions can be set for any of the files or directories in the TINI. The FTP server has numerous environment variables and has gone through many changes as the TINI JVM has been developed. The FTP server can be activated by any Java application using the TINI.System classes. Consult the TINI API JavaDocs for details.

INTEGOS Operating System

INTEGOS is the INTEG process group's implementation of Slush. It is responsible for the warm and cold boot process. INTEGOS is a version of Slush that requires some additional commands to handle I/O functions specific to JNIOR, but the structure can remain essentially unchanged. The boot-up text banners that scroll out the standard device (RS232 or Telnet) need to be changed to a JNIOR flavor. INTEGOS will be the lowest level of access allowed for base (non-programming) customers; therefore care must be taken to ensure that a customer cannot use a command to "kill" their JNIOR and cost INTEG revenue or reputation. The changes required to turn Slush into INTEGOS are as follows:

- Change text message banners to reflect JNIOR and INTEG process group, inc.
  - INTEG process group, inc. contact information
  - Visibility to JNIOR I/O device and configuration information
    - (See for ***Table 8 INTEGOS I/O Specific Command*** Set command details)
  - Ability to enable or disable the banner messages
  - Software revision level
  - Date and time
- FLASH file system configuration
  - Create user and root accounts
  - Create file structure and set permissions
- Change command interface
  - Create commands for polling and setting I/O points
  - Remove unnecessary commands
  - Lock out users from sensitive commands (if not already inherent)

The present Slush OS is loaded into the TINI FLASH memory with a PC based utility called JavaKit. The FLASH memory is segmented into 8 blocks. At boot time the TINI's resident firmware looks for a valid Java application in block 7 and passes execution if it finds a hook. INTEGOS will be loaded in the same manner.

The INTEGOS needs to have access to the same system configuration and real-time I/O point data utilized by the JNIORServer application to execute I/O specific commands. We need to develop a method to share data between these two applications. It may be possible to use the environment variable utilized in Slush.

The INTEGOS ultimately provides a mechanism to operate a JNIOR via automated Telnet sessions that run on a system host computer. In this mode of operation, a control program resident on a network host logs in, polls I/O point data, and sets the outputs accordingly. The JNIOR operates without a web-browser, although it could be used to complement such a system. The JNIOR acts as a "dumb" data acquisition device providing nothing more than remote connectivity. For end-users it provides a means to build up a network of devices all controlled from a central program. (This diverges from the intelligent node concept but the upside is that there is only program to develop and maintain.)

JNIORServer Software Description

The JNIORServer Java application is a stand-alone program that provides the JNIOR product's I/O points with a configuration method and connectivity. JNIORServer auto-executes at the end of the INTEGOS startup sequence. This can be a hard-coded change to INTEGOS or accomplished using the scripted command "*Java JNIORServer &*" within the systems .startup file. The handoff is seamless from the end-user's perspective.

JNIORServer is a miniature version of the typical web server found on the Internet. When a customer pulls their JNIOR out of the box it is critical that they get up and running quick and painless. JNIORServer uses an integral HTTP server to communicate with a standard web browser. Customer set-up should be limited to the following:

- Physically connect the JNIOR's Ethernet port to their network or PC.
- Set the JNIOR IP address as desired via the INTEGOS command using RS232 and a dumb program terminal. INTEG should consider providing a PC utility that automates the IP address configuration from a GUI'd exe.
    - (Default the IP address to 192.168.1.99, popular but out of the way)
- Enter the JNIOR box IP address into their favorite web browser's URL location
- Point, click, and fill in text boxes to configure system and I/O functions

The customer is greeted with the JNIORServer HTTP home page when they connect via a web browser. From there they have the option to configure, monitor, or view online documentation. JNIOR will look like a simple web site that has the ability to view dynamic data or change I/O output from afar via a private LAN or the Internet.

(The more technically savy customer can use JNIOR's INTEGOS I/O commands to remotely login and do the same via automated script files from a host computer.)

JNIORServer HTML Web Pages

The JNIOR product's man machine interface is a PC web browser that receives HTML pages from JNIOR's HTTP server. When an end-user "hits" a JNIOR device they are greeted with a home page that allows them to navigate the products functionality. Users need to configure the I/O and system, monitor data, and access simple help documentation. The following tables itemize the operation of the JNIOR web pages:

Table 1 JNIORServer Web-Pages

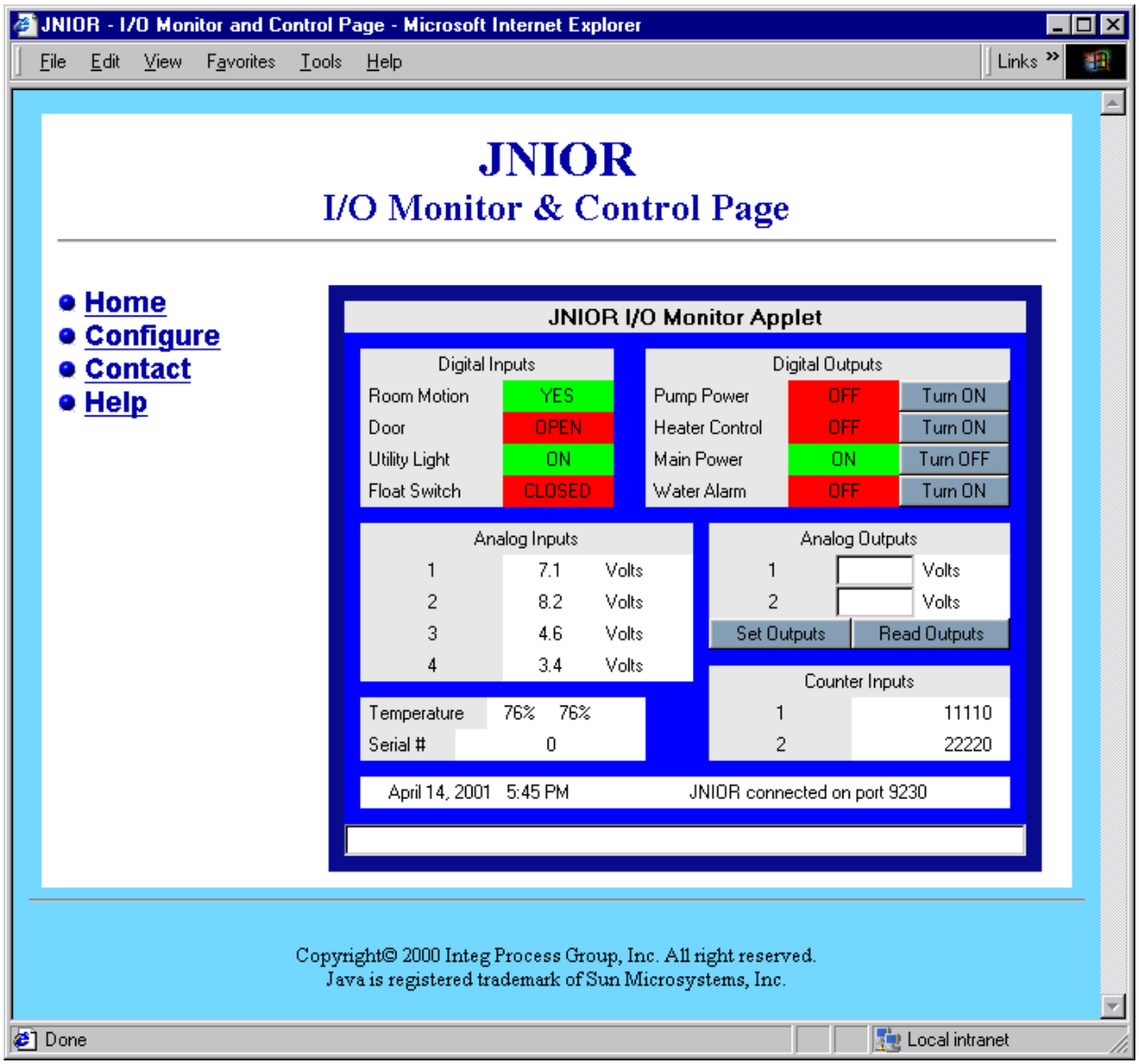
Home	index.html – Default home page. Show JNIOR name and INTEG contact information. Display time, date, temperature and uptime statistics.
Configuration	config.html – Web page that contains the configuration applet. Used to setup the I/O and system operating parameters.
Contact	Contact.html – INTEG process group inc. contact information and a plug for our custom design and development services.
Monitor	monitor.html – Web page that contains the monitor applet. Used to monitor and control JNIOR's I/O points.
Docs	docs.html – Readme, help, and FAQ files in HTML form.

Home Web Page

The JNIOR home page is the default HTML page sent by the web server when it receives a hit from a browser via the Ethernet network connection. The home page contains dynamic information to show operating status and the current time and date. The JNIOR home page could be completely static HTML or contain Applets. The home page needs to load quickly and not replicate the configuration, monitor, and control functionality found elsewhere, hence Applets may not be necessary, will slow the load time, and cause confusion to an end-user who does not have Java enabled. A static page, built by the JNIORServer can serve the purpose of displaying the most critical system parameters and show end users that their JNIOR product is functioning. The following is an example of the JNIOR's home page:

Monitor Web Page

The monitor.html web page is used to view and control JNIOR's I/O points from a web browser. The web page contains a Java applet that contains the GUI and TCP/IP socket functionality to allow end users to make real-time connections to a network reachable JNIOR. The following is an example of the JNIOR monitor.html web page:



Configuration Web Page

The configure.html web page is used to setup JNIOR's system and I/O parameters from a web browser. It contains the configuration applet. There is no example available

JNIORServer HTML Applets:

Java applets are used as an interface between a JNIOR device and a web browser. The applets are embedded in standard HTML web pages and make TCP/IP socket connection with the JNIORServer application program running on JNIOR. The configuration applet is used to interface an end users GUI to JNIOR's configuration file information. The monitor applet is used to do the same with JNIOR's I/O points in real-time.

I/O Configuration Applet

The configuration applet is used to setup the operation of JNIOR's system and I/O devices. The applet has the following requirements:

- Graphical User Interface
  - User defined text names for I/O points
  - I/O point specific configuration (See the I/O Functionality section for details)
- JNIOR TCP/IP data socket connection
  - Connect for read/write of configuration information
  - Flexible port number, configurable with IPCONFIG if possible
- JNIOR Configuration information
  - Read existing JNIOR system and I/O setup parameters and display to GUI
  - Update JNIOR resident configuration information file
  - Password security, use existing TINI structure if possible

The configuration applet updates the JNIOR I/O and system parameters located in the ***.ioconfig*** file in the ***/etc*** directory. The ***.ioconfig*** configuration file is a text script that contains information specific to setup and I/O point. For example an analog output would have a channel number, text name, A/D resolution, A/D range, scaling equation, and real world unit of measure. The information located in the ***.ioconfig*** file complements the lower level ***.iomap*** file used to map 1-Wire devices to JNIOR I/O. These two files define the JNIOR product configuration from 1-Wire device population to the name and units displayed on the web pages and Applet GUIs.

I/O Monitor Applet

The monitor applet is embedded in JNIOR monitor.html web page and used to display the real time status and allow users to control JNIOR's I/O devices. The monitor applet has the following functionality:

- Graphical User Interface
  - User defined text names for I/O points
  - Dress the I/O point data with configuration information
- JNIOR TCP/IP data socket connection
  - Connect for read/write of I/O data point information
  - Flexible port number, configurable with IPCONFIG if possible
- JNIOR monitoring/control information
  - Read real-time I/O point data and display to GUI
  - Allow end-users to control output devices from the GUI
  - Display alarm conditions and email status
  - Password security, use existing TINI structure if possible

JNIORServer I/O Functionality

A fully populated JNIOR product contains analog and digital I/O points, a serial port, and applet contains analog and digital I/O points, serial, and temperature I/O points. The configuration and monitor applets need to access the I/O hardware populated on any given flavor of JNIOR. The I/O hardware consists mostly of Dallas semiconductor 1-Wire devices. The specific population is determined by the JNIOR product make or the requirements necessary to deliver a custom solution.

1-Wire I/O Devices

The 1-Wire devices operate on a shared bus and use embedded serial numbers to identify themselves. The TINI Java software API has high-level containers that are used to access 1-Wire device functions. Table 2 identifies the 1-Wire I/O devices that could be populated on a JNIOR PCB assembly, Table 3 details the configuration parameters for JNIOR's I/O points.

Table 2 JNIOR 1-Wire I/O Devices

Description	Hardware Device
(4) Analog Inputs	(1) DS2450 quad A/D converter.
(2) Analog Voltage Outputs	(2) DS2890 Digital Potentiometer.
(2) Counter Inputs	(1) DS2423 Dual Counter
(4) Digital Voltage Inputs	(2) DS2406 dual switch
(2) Digital Voltage Outputs	(1) DS2406 dual switch
(2) Form-C Relay Outputs	(1) DS2406 dual switch
Temperature Sensor	(1) DS1820 temperature sensor

* The Dallas i-Button container information is located in Appendix D.

Table 3 JNIOR I/O Point Configuration Details

I/O Point	Parameter	Description
Analog Input:	Name label	Text field
	A/D bit resolution	8,10,12, or 16 bit
	A/D Full-scale voltage	5 or 10VDC
	Units	Text field
	Equation slope	m of y=mx+b scaling equation
	Equation offset	b of y=mx+b scaling equation
	Alarm enable	True or false
	Alarm low limit	Scaled low alarm value
	Alarm high limit	Scaled high alarm value
Analog Output:	Name label	Text field
	Units	Text field
	Equation slope	m of y=mx+b scaling equation
	Equation offset	b of y=mx+b scaling equation
Counter Input:	Name label	Text field
	Units	Text field
	Equation slope	m of y=mx+b scaling equation
	Equation offset	b of y=mx+b scaling equation
	Sampling period	Sample period for read count and do equation
	Alarm enable	True or false
	Alarm low	Scaled low alarm value

	Alarm high	Scaled high alarm value
Digital Input:	Name label	Text field
	True label	Text field for input ON
	False label	Text field for input OFF
	Invert input logic	True or false
	Alarm enable	True or false
	Alarm state	Alarm when ON or OFF
Digital Output:	Name label	Text field
	True label	Text field for output ON
	False label	Text field for output OFF
	Invert output logic	True or false
Relay Output:	Name label	Text field
	True label	Text field for output ON
	False label	Text field for output OFF
	Invert output logic	True or false
Temperature:	Name label	Text field
	Units	Text field
	Equation slope	m of y=mx+b scaling equation
	Equation offset	b of y=mx+b scaling equation
	Alarm enable	True or false
	Alarm low	Scaled low alarm value
	Alarm high	Scaled high alarm value

RS232 Port

JNIOR's RS232 port can be used for a variety of functions. It is used to program the TINI board FLASH memory with *JavaKit*, login to the Slush operating system, and serve as a general-purpose serial port. It is expected that there will be an interest in using JNIOR to connect legacy equipment to LANs via the RS232 and Ethernet ports. This I/O population could consist of a minimal number of 1-Wire devices, if any, resulting in a very low cost device. For this reason, INTEG needs to make the RS232 port accessible and simple to program. There is a myriad of protocols and functionality options that could be considered for the RS232 port and each has their merits. INTEG needs to get the JNIOR product ready for delivery as soon as possible; hence we should limit the functionality at this time and concentrate and simply making the port available with the appropriate hooks.

A simple terminal program that can be remotely accessed via a web page would provide the best tradeoff between functionality and development effort. A terminal program could be an applet that looks like *HyperTerm* and allows end users to remotely access equipment connected to JNIOR's RS232 port. This functionality would be useful to demonstrate JNIOR's capabilities as a network proxy for legacy equipment.

There is an example provided with the TINI software release that demonstrates an applet based remote terminal. Source code is included. The effort required to port this example is considerably less than what it would take to create custom specifications or mimic an existing standard. Additional communication protocols and functionality could be added in the future. The JNIORServer Java application should allow for the following programmability in JNIOR's RS232 port.

JNIOR's RS232 port is physically accessible via a DB-9 female style connector. End-user connect to a standard PC COM port with a null-modem cable. The interface is essentially 3-wire with the exception of the DTR line that is used by *JavaKit* to reset the TINI CPU during FLASH memory upgrades. The DTR is normally isolated from the connector by means of a solder-pad jumper that is only connected when that particular PCB is being used to program a TINI's FLASH memory.

Push-Button Switch Input
The JNIOR PCB assembly contains circuitry necessary to read a pushbutton switch via the TINI's P5.1 (CRX) general-purpose I/O bit. The pushbutton input signal is available via solder pads located on the PCB assembly. The pushbutton input signal is pulled up to VCC with a resistor and read as a logic low when the pushbutton completes the circuit to common. The pushbutton input could be used to provide some means of input from a local user. There are no requirements for a pushbutton on the present JNIOR product; hence it is loaded on the existing PCB assembly. (The pushbutton I/O pin could be used as a control bit for external circuits scabbed into the JNIOR enclosure.)

Beeper Output
The JNIOR PCB assembly contains circuitry necessary to drive a small audio "beeper" via the TINI's P5.0 (CTX) general-purpose I/O bit and a high-side driver. The beeper output signal is available via solder pads located on the PCB assembly. A beeper could be used to provide some means of feedback to the unconnected user. There are no requirements for this functionality at this time; hence the beeper device is not loaded on the JNIOR product. (The beeper control circuit could also be used to provide a switch +5V source for external circuits scabbed into JNIOR enclosure)

CPU Watch-Dog Timer
Operators of remote equipment installations cannot be burdened with having to reboot equipment out in the field. JNIOR contains a watchdog timer circuit that will reboot the TINI's CPU if it doesn't receive a kick-signal from one of three jumper selectable I/O bits: P3.3, 1-Wire I/O network, or CPU address line A0. The latter, A0, is used to disable the watchdog as it toggles constantly and is not under direct software control.

The specific operation of the JNIOR watchdog is not determined at this time. The timing load off the INTEGOS will need to be understood and a health monitoring mechanism needs to be developed. The 1-Wire I/O network port line could be used to reset the watchdog if the software can ensure that the network is being polled at a minimum of 1 Hz. Ideally, the watchdog is driven by port pin P3.3 every ½ second when JNIOR's health status is normal, meaning no locked-up code. The most robust watchdog would be written in the TINI's native assembly language and available to the Java applications running on JNIOR.

JNIORServer Configuration Files

The shared bus architecture of the 1-Wire network add a layer of complexity in that each JNIOR product needs to be configured to know which device is in a specific JNIOR I/O location. The Java classes that communicate with I/O points need to know how to discriminate between digital input 1 and relay output 2. Each JNIOR product will need to undergo an I/O point search and configuration process prior to shipment or even before doing anything meaningful with JNIORServer.
The JNIOR product configuration process can be done together with the PCB assembly test process at INTEG's contract manufacturer. The test and configure process can be automated

with a host PC or even another JNIOR. The details will need to be developed with INTEG's manufacturer and the method employed will probably depend heavily on production volumes. (For immediate development purposes the files can be created manually and uploaded into the JNIOR using FTP.)

.iomap File

The ***.iomap*** file is used by classes that need to access I/O devices. The ***.iomap*** file is located in the /etc directory and used to map the 1-Wire device serial numbers to their location in the JNIOR I/O population. It is created during the JNIOR product's factory checkout and test process. The JNIOR device identifier on the right side corresponds to the analog, digital, counter, or relay I/O device in a particular PCB location. Table 4 shows the details for the JNIOR I/O device configuration names. The following is an example of a JNIOR ***.iomap*** file:

```
# JNIOR .iomap file
# example created 4/18/01
# 1-Wire device serial number = JNIOR device number
4300000017FD8412 = ai1
D300000017FD8411 = ao1
4F00000017FD8415 = ao2
4600000017FD8412 = ci1
E600000017FA0712 = di1
7D00000017FA0712 = di2
5BD0000017FB0712 = do1
2AC0000017FE0712 = ry1
91D0000017FC0712 = t1
```

Table 4 JNIOR I/O Device Configuration Names

JNIOR Device ID	Description	Suffix
aixs	Analog input	Channel a,b,c, or d of DS2450
aox	Analog output	
cixs	Counter input	Channel a or b of DS2406
dixs	Digital input	Channel a or b of DS2406
doxs	Digital output	Channel a or b of DS2406
ryxs	Relay output	Channel a or b of DS2406
Tx	Temperature sensor	

x = the device number, s = Suffix

.ioconfig File

The ***.ioconfig*** file is used to hold end-user definable configuration parameters for the JNIOR system. The ***.ioconfig*** file is generated by the JNIOR system via the configuration applet and used by the monitor applet to deliver meaningful data to the GUI. An example of an ***.ioconfig*** file is located in Appendix E.

Email Alarm Functions

The JNIOR product is capable of sending email messages in response to changes in I/O state or if an alarm condition has been reached. The analog, digital, and counter inputs are all have configurable low and high alarm values as well as an alarm enable. When an alarm condition is reached, the JNIOR sends a canned email message with the status of the I/O points and a time and date stamp to the email address specified on the JNIOR system configuration.
The email alarm function also has a global enable, a delay time, and a parameter specifying the maximum number of email to be sent in any 24-hour period. The latter will help to provide

peace while commissioning a new installation or preventing unintended conditions from hogging the email servers involved. The email/alarm configuration information is kept in the JNIOR ***.ioconfig*** file. Table 5 details the email/alarm parameters:

Table 5 JNIOR Email Alarm Configuration Details

Parameter	Description	Example
email	Email address of alarm recipient	
email_cc	Email addresses of carbon copy recipients	
email_alarm	Email alarm enable	email_alarm, 1
email_trig	If true, send emails every repeat time. If false, send email only once. Default is send once.	email_trig, 1
email_delay	Delay in seconds between alarm state and email being sent. Condition must persist for this period or the email engine ignores it.	email_delay, 60
email_max	Maximum number of emails sent in a 24 hour period.	email_max, 24
email_repeat	Time in minutes between sending a repeat email.	email_repeat, 30

Appendix A: Glossary of Terms and Abbreviations

Table 6 Glossay of Terms

A/D	Analog to digital converter, outputs a digital value to represent an analog input
Applet	Java software program distributed in an HTML page and run in a web browser
API	Application programming interface
CAT5	Category 5 wiring used for Ethernet local area networks
CPU	Central processing unit
DNS	Domain name server, translates URL names to IP addresses
DHCP	Dynamic host control protocol, used to get dynamic IP addresses
FLASH	Nonvolatile memory type that is in-circuit programmable
GUI	Graphical user interface
HTML	Hypertext markup language, used to make web pages
HTTP	Hypertext transfer protocol, used to serve web pages
INTEGOS	INTEG operating system derived from Slush.
I/O	Analog or digital input/output point
JavaKit	Java application provided by Dallas Semiconductor, used to program TINI's FLASH
JNIOROS	The Java Network I/O Resource Operating System. Made up of two individual Java components, INTEGOS and JNIORServer.
JNIORServer	Java application that runs JNIOR
JVM	Java Virtual Machine, a software interpreter that runs Java bytecode
LAN	Local area network
OS	Operating system software
PCB	Printed circuit board
ROM	Read only memory, usually used to hold non-volatile programs
RS232	Serial port interface standard used in PC COM ports
RTC	Real time clock and calendar
SDK	Software developers kit
SPDT	Single pole double throw, switch contact type
SRAM	Static random access memory, usually used for volatile variable storage
TCP/IP	Transport control protocol/ Internet protocol
TFTP	Tiny file transfer protocol, used to transfer files and boot images to network devices
TINI	Tiny Internet Interface
TTY	Unix nomenclature for RS232
UTP	Unshielded twisted pair, a type of wiring and connector style
VCC	Digital circuit supply voltage, usually +5V

Appendix B: INTEGOS Command Set

Table 7 INTEGOS Command Set

Command	Description
arp	Dumps all ARP cache entries.
append	Appends source file to the destination file.
cat	Displays the contents of a file.
cd	Changes the current working directory.
chmod	Change the permissions of a file.
chown	Changes the owner of a file to a specific user.
clear	Clear the screen.
copy	Copies the source file to the destination file. Alias cp.
cp	Copies the source file to the destination file. Alias copy.
date	Display or set the system time information.
del	Deletes a file. Alias rm.
df	Get amount of free RAM.
dir	Returns a listing of the files. Alias ls.
downserver	Shuts down the specified server. Alias stopserver.
echo	Echo text to the output device.
gc	Run the system garbage collector.
genlog	Toggles system log generation on boot.
help	Lists all of the available commands or the specified usage
history	Displays the command history list.
hostname	Displays or sets the host and domain name.
ipconfig	Configure or display the network settings.
java	Executes the specified Java class.
kill	Kills the specified process.
ls	Returns a listing of the files. Alias dir.
md	Create a directory. Alias mkdir.
mkdir	Create a directory. Alias md.
move	Move the file from the source to the destination. Alias mv.
mv	Move the file from the source to the destination. Alias move.
nslookup	Displays the name or IP of the given argument.
passwd	Sets the password for the specified user.
ping	Sends ICMP ECHO_REQUESTS to network hosts.
ps	Gets a list of currently running processes.
pwd	Displays the current working directory.
rd	Deletes the specified directory. Alias rmdir.
reboot	Shuts down all servers and cleanly reboots the system.
rm	Deletes a file. Alias del.
rmdir	Deletes the specified directory. Alias rd.
sendmail	Send an email to designated recipients.
setenv	Set or display value of an environment variable.
source	Executes each line in a file.
startserver	Starts up the specified server.
stats	Displays current system status information.
stopserver	Shuts down the specified server. Alias downserver.
su	Switch user to specified or root.
touch	Sets the last modified time to the current time.
useradd	Adds a user to the system.
userdel	Removes user from the specified system.
wall	Broadcast message to all users.
wd	Set or display the Slush watchdog timer settings.
who	Displays all users currently logged into the system.
whoami	Displays the current user’s user name.

Appendix C: INTEGOS I/O Command Set

Table 8 INTEGOS I/O Specific Command Set

Command	Description
AI n	Return value of analog input n
AI% n	Return % of full-scale of analog input n
AO n dddd	Set value of analog output
CR n	Return value of counter n
CS n dddd	Set counter n to count dddd
CZ n	Set counter n value to 0
DI n	Return value of digital input n
DO n d	Set value of digital output n to d
DP	Returns a dump I/O configuration information
RY n d	Set relay output n to d
TPC	Return temperature in C
TPF	Return temperature in F

* all command are completed with a <CR> and optional <LF>

Appendix D: Dallas 1-Wire I/O Device Containers

The following table contains the software container information for the 1-Wire devices that could be populated on the JNIOR PCB assembly.

Table 9 1-Wire Container Family

Device	Family	Description	Interfaces	Memory Banks
DS1990A, DS2401	01	1-Wire Address only		
DS1820, DS18S20, DS1920	10	Temperature and alarm trips	Temperature-Container	
DS2406, DS2407	12	1K EEPROM memory, dual switch	SwitchContainer	MemoryBank, PagedMemoryBankO, TPMemoryBank
DS2423	1D	4K NVRAM memory with external counters		MemoryBank, PagedMemoryBank
DS2450	20	Quad A/D	ADContainer	MemoryBank, PagedMemoryBank
DS18B20	28	Adjustable resolution temperature	Temperature-Container	
DS2890	2C	Single channel digital potentiometer	Potentiometer-Container	

Appendix E: Example JNIOR .ioconfig File

The following is an example of an .ioconfig file:

```
# JNIOR /etc/.ioconfig file
# analog inputs, device, channel =
# [name, resolution, range, units, equation slope, and offset, alarm, alarm lo, alarm hi]
ai1a, Flow Rate, 12, 10, gpm, -1.27, 50.4, 1, 0.5, 10
ai1b, Tank Level, 8, 5, cm, 0.234, 0, 1, 8, 10
ai1c, Tank Temperature, 10, 10, C, 1.84, 37.26, 0
ai1d, Battery Voltage 12, 5, Volts, 3.75, 0.6, 1, 11, 14

# analog outputs, device =
# [name, units, equation slope, and offset]
ao1, Speed Control, RPM(x100), 251, -5
ao2, Heater Setpoint, C, 9.5x+10.2

# counter inputs, device, channel =
# [name, units, equation slope, and offset , sample time(seconds), alarm, alarm lo, alarm hi]
ci1a, Pump Speed, RPM, 0.0015, 10, 120, 1, 100, 3000
ci1b, Crane Use, Cycles, 1, 0, 0, 1, na, 5000
# (read every sample time and apply equation, if 0 then display raw count)

# digital inputs, device, channel =
# [name, true, false, invert input logic, alarm enable, alarm state]
di1a, Room Motion, YES, NO, 0, 1, 1
di1b, Door, CLOSED, OPEN, 0, 1, 0
di2a, Utility Light, ON, OFF
di2b, Float Switch, CLOSED, OPEN, 1

# digital outputs, device, channel
# [name, true, false, invert output logic]
do1a, Pump Control, ON, OFF, 0
do1b, Heater Control, AUTO, ON, 0

# relay outputs, device, channel
# [name, true, false, invert output logic]
ry1a, Power Switch, ON, OFF, 0
ry1b, Water Level Alarm, WET!, off, 1

# temperature sensor, device
# [name, units, equation slope, and offset, alarm, alarm lo, alarm hi
t1, Cabinet Temperature, K, 0.01, -273, 1, na, 45

# TCP/IP sockes for monitor and control applets
mon_socket, 9100
con_socket, 9101
web_socket, 80

# email address and condition information
email_address,
email_alarm, true          # global email alarm enable
email_delay, 5             # condition email delay in seconds
email_max, 2               # maximum number of emails to be sent in a 24 hour period
```