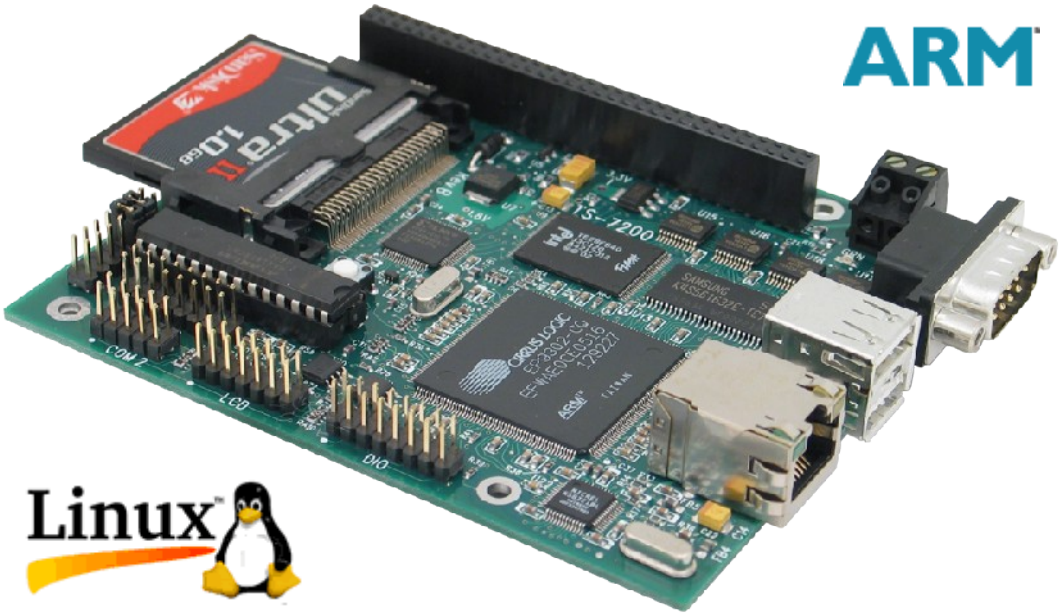
Technologic Systems

www.embeddedARM.com

Linux for ARM on TS-72XX User's Guide



Technologic SYSTEMS

2.5

Feedback and Update to this Manual

To help our customers make the most of our products, we are continually making additional and updated resources available on the **Technologic Systems website** (www.embeddedARM.com).

These include manuals, application notes, programming examples, and updated software and firmware. Check in periodically to see what's new!

When we are prioritizing work on these updated resources, feedback from customers (and prospective customers) is the number one influence. If you have questions, comments, or concerns about your Embedded Computer, please let us know at **support@embeddedARM.com**.

Limited Warranty

Technologic Systems warrants this product to be free of defects in material and workmanship for a period of one year from date of purchase.

During this warranty period Technologic Systems will repair or replace the defective unit in accordance with the following process:

A copy of the original invoice must be included when returning the defective unit to Technologic Systems, Inc.

This limited warranty does not cover damages resulting from lightning or other power surges, misuse, abuse, abnormal conditions of operation, or attempts to alter or modify the function of the product.

This warranty is limited to the repair or replacement of the defective unit. In no event shall Technologic Systems be liable or responsible for any loss or damages, including but not limited to any lost profits, incidental or consequential damages, loss of business, or anticipatory profits arising from the use or inability to use this product.

Repairs made after the expiration of the warranty period are subject to a repair charge and the cost of return shipping. Please, contact Technologic Systems to arrange for any repair service and to obtain repair charge information.

FCC Advisory Statement

This equipment generates, uses, and can radiate radio frequency energy and if not installed and used properly (that is, in strict accordance with the manufacturer's instructions), may cause interference to radio and television reception. It has been type tested and found to comply with the limits for a Class A computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference, in which case the owner will be required to correct the interference at his own expense.

If this equipment does cause interference, which can be determined by turning the unit on and off, the user is encouraged to try the following measures to correct the interference:

- Reorient the receiving antenna.
- Relocate the unit with respect to the receiver.
- Plug the unit into a different outlet so that the unit and receiver are on different branch circuits.
- Ensure that mounting screws and connector attachment screws are tightly secured.
- Ensure that good quality, shielded, and grounded cables are used for all data communications.

If necessary, the user should consult the dealer or an experienced radio/television technician for additional suggestions. The following booklets prepared by the Federal Communications Commission (FCC) may also prove helpful:

- How to Identify and Resolve Radio-TV Interference Problems (Stock No. 004-000-000345-4)
- Interface Handbook (Stock No. 004-000-004505-7)

These booklets may be purchased from the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 About this Manual

This manual is a brief introduction to the use of Linux on a Technologic Systems' (TS) ARM-based Single Board Computer (SBC) – TS-72XX series. Many technical questions are answered within this document This manual is not meant as a general tutorial on Linux (or Linux Development), but is designed specifically to cover issues related to using Linux on Technologic Systems' ARM SBCs.

Technologic Systems' ARM products offer a different set of tools than the x86 products. This manual documents the basic knowledge needed to work with the ARM products.

> **Warning**
> This document was created to be used with the **TS-7200**, **TS-7250** and **TS-7260** boards (TS-72XX series) and contains sections specific to these products only, such as the **second chapter**. However, most of the remaining sections will still be useful guidelines for other TS-ARM computers (**TS-7300**, **TS-7400**, etc).

If your project demands rapid application development, Technologic Systems does offer software engineering services. We also offers custom hardware design. Please contact us for more information.

1.2 TS-72XX Series

The TS-72XX series Single Board Computers (SBC's) run on a 200 MHz ARM9 processor with power as low as 1/2 Watt. Low board complexity, low component count, and low power/heat makes for an extremely reliable embedded engine. The TS-72XX SBC's are available in thousands of configurations, many of which are Commercial off the Shelf (COTS) and available to ship today.

The EP9302 processor from Cirrus is the highly integrated 200Mhz ARM9 processor that the TS-72XX SBC's are built around and includes an on-chip 10/100 ethernet, USB, serial, and Flash/SDRAM controller. For example, on the TS-7200 model there is 32 Mb of Micron SDRAM running at 66 Mhz and 8 Mb Intel Strata flash on-board. A supplemental PLD provides glue logic, watchdog timer, Compact Flash IDE, and 8 bit PC/104 support. Integer CPU performance is about 20% faster than our 133 Mhz x86 offerings.

Even with the standard power consumption of 2 Watts, the TS-72XX SBC's run without fans or heat sinks in the temperature range of -20° to +70°C. Extended Temperature -40° to +85°C is also standard, but CPU clock must be decreased to about 166MHz for higher temperatures. Digital Signal Processing (DSP) is enabled through a standard 5 channel, 12bit A/D converter (Optional 8 channel, 12 bit A/D converter), 20 DIO lines and 2 standard serial ports.

The 8/16 bit PC/104 interface enables additional functionality through Technologic Systems' broad product line of PC/104 peripheral daughter boards. The TS-7KV adds video, CAN, Com Ports, and A/D conversion. The TS-ETH10 allows the addition of Ethernet ports. The TS-CAN adds CAN connectivity. The TS-Modem boards add both wired and cell phone capabilities.

The TS-72XX rugged ARM9 SBC's have found their way into many embedded applications. Customers are using the TS-72XX series SBC's in: energy generation, manufacturing process control, traffic management, printing system management, communication infrastructure, website hosting, data gathering and laboratory test equipment. We use a TS-7200 to host our complete website and to prepare and test your SBC prior to shipping.

2 STARTUP

> **Warning**
> This chapter is only applicable to the TS-72XX computers.

2.1 Console and Power Up

The TS-72XX SBCs have no video controller or keyboard interface. This was done to keep the board size small and the cost low. COM1 is typically used as a console port to interface the TS-72XX to a standard terminal emulation program on a Host PC.

An ANSI terminal or a PC running a terminal emulator is required to communicate with your Embedded PC. Simply connect an ANSI terminal (or emulator) to COM1 (DB9 female connector) using a null modem cable (this is included in the TS-ARM Development Kit), using serial parameters of 115,200 baud, 8 data bits, no parity, no flow control, 1 stop bit (8N1), and make sure jumper **JP2** is installed. If you are running Linux, the minicom program works well, Windows users can run the Hyperterm application. Technologic Systems offers a null modem cable with both 25 pin and 9 pin connectors at each end as part number CB7-05. Some systems also require the 10-pin header to 9-pin Sub-D adapter which is P/N: RC-DB9.

The console can be changed to COM2 by installing **JP4** (with JP2 also installed). If your application does not require a console or both COM ports are required, then removing the jumper **JP2** easily disables all console output.

Connect a regulated 5VDC, (1A minimum) power source using the included 2 screw terminal strip/connector. Please note the polarity printed on the board. The boot messages, by default, are all displayed on COM1 at 115200 baud.

2.2 Boot Sequence

The boot sequence has four distinct stages:

1. TS-BOOTROM messages
2. RedBoot ROM monitor messages
3. Linux Kernel messages
4. Login prompts

After applying 5VDC power, the board mounted LED will blink, followed immediately by the display of TS-BOOTROM messages and RedBoot messages. RedBoot, if not interrupted by the user within one second, loads the Linux kernel from flash and boots to the on-board flash chip.

> **Warning**
> Ensure that JP5 is removed when booting from the default flash file-system. Special scripts look for it and, when found, run special test programs that will use up system resources.

2.3 TS-BOOTROM

Upon power up, the board executes proprietary Technologic Systems boot-code, then immediately executes RedBoot. The TS-BOOTROM is stored in flash memory and is used to perform the first hardware configuration, such as the EP9301 initialization and TS-72XX hardware specifics, and to load the eCos/RedBoot system. The TS-BOOTROM serves as reusable way of initializing and checking hardware in a way that it is not necessary to rewrite the code for every desirable OS. Some of the TS-BOOTROM configuration tasks include:

- ✔ initializes the processor

- ✔ turns off the watchdog timer
- ✔ configures the serial UART to 115200 bps
- ✔ initializes and tests SDRAM and FLASH
- ✔ loads and runs RedBoot from flash

There is an intermediate boot loader code on the TS-72XX SBCs which use the NAND flash technology, named TS-NANDBOOT. It is needed because the EP9301 processor can't handle NAND devices. Then, the TS-NANDBOOT contains specific routines to read the NAND flash memory and load the TS-BOOTROM.

2.4 eCos/RedBoot

RedBoot is a feature rich boot-ROM monitor, that allows manipulation of the on-board flash, JFFS and YAFFS images, loading and execution of a kernel or executable from either tftp (trivial ftp), http or flash, and gdb debugging stubs. From RedBoot, one can load and execute any standalone binary. Most commonly, a Linux kernel or a Windows CE binary is used. One can also write applications within the eCos environment and load them with RedBoot.

Using Redboot

By default, a pre-existing RedBoot script is executed on initialization time, if not interrupted by the user within one second. The default script instructs RedBoot to load the Linux kernel from the flash, and instructs the Linux kernel to use the JFFS/YAFFS image on the flash chip for its root file system. One can view the RedBoot defaults for the board, as well as the default script, by entering at the RedBoot command prompt (Ctrl+C within one second after power up):

```
RedBoot> fconfig -l
Run script at boot: true
Boot script:
.. fis load vmlinux
.. exec -c "console=ttyAM0,115200 root=/dev/mtdblock1"
Boot script timeout (100ms resolution): 1
Use BOOTP for network configuration: false
Gateway IP address: 192.168.0.11
Local IP address: 192.168.0.50
Local IP address mask: 255.255.255.0
Default server IP address: 192.168.0.11
dns_ip: 192.168.0.11
Network hardware address [MAC]: 0x00:0xD0:0x69:0x4F:0x34:0xA5
GDB connection port: 9000
Force console for special debug messages: false
Network debug at boot time: false
```

The defaults can be changed by simply entering “fconfig” at the RedBoot prompt and answering the prompts. A final chance to write or discard the changes to the board will be given by RedBoot. Also, the main RedBoot commands can be viewed by entering “help” at the prompt, and further information about a single command can be viewed by typing “help <command name>”.

Loading and execute kernel from RedBoot

RedBoot can load a kernel or executable via the serial console, a tftp server, http server, or directly from flash.

> **Important**
> The Linux kernel must be loaded into memory address 0x00218000.

FLASH

Loading the kernel from flash is done automatically by RedBoot in the default script with the following command:

```
RedBoot> fis load zimage -b 0x00218000
```

One can see where in flash the kernel is and where in memory it will be loaded to by typing “fis list”, which shows the various areas of flash RedBoot is aware of. The following is a typical output of “fis list”:

```
Name              FLASH addr  Mem addr    Length      Entry point
(reserved)        0x60000000  0x60000000  0x00620000  0x00000000
RedBoot           0x60620000  0x60620000  0x00040000  0x00000000
vmlinux           0x60660000  0x00218000  0x00160000  0x00218000
RedBoot config    0x607C0000  0x607C0000  0x00001000  0x00000000
FIS directory     0x607E0000  0x607E0000  0x00020000  0x00000000
```

From the above example, the kernel executable labeled vmlinux is stored at flash address 0x60660000, and will be loaded into memory address 0x00218000.

It is possible to write a new kernel loaded by the user in memory to flash, thereby overwriting the pre-existing kernel stored in flash. First, one must delete the existing kernel image file from flash, then write the new loaded one to disk. The following commands accomplish this:

```
fis delete <kernel name>
fis create -b 0x00218000 -l 0x160000 <new kernel name>
```

Note that the delete command specified <kernel name>, which refers to the name of the fis entry to delete. The create command specified <new kernel name>, which is the name of the executable loaded into memory. If the name of the downloaded executable is different, please specify that name when doing a create command.

HTTP

Prior to loading a new kernel into RebBoot using HTTP, make sure you have configured RedBoot with a network configuration that can reach the network. You may use the RedBoot "fconfig" command to set network parameters.

Get to the RedBoot prompt by hitting Ctrl+C key immediately after power on and type the following command:

```
load -v -r -b 0x00218000 -m http -h <http sever IP> <kernel name>
```

For example:

```
load -v -r -b 0x00218000 -m http -h 67.40.67.44
/ftp/ts7kv/vmlinuxts7200ts9.bin
```

If RedBoot gives you an error about not understanding the "m http" option, you have an older version of RedBoot and must instead load the kernel via a local TFTP server.

TFTP

To load a kernel from a simple TFTP server, the following command is needed:

```
load -r -b 0x00218000 -h <tftp server IP> <kernel name>
```

For example:

```
load -b 0x00218000 -h 192.168.0.1 vmlinux
```

Executing the Kernel and Booting the Root File System

Now that a kernel has been loaded into memory, it can be executed. This is accomplished with the following command:

```
exec -c "<kernel parameters>”
```

For example:

```
exec -c "console=ttyAM0,115200 ip=dhcp root=/dev/mtdblock1"
```

The "exec" command executes the loaded kernel image, passing to the kernel the arguments specified via the "-c" switch. In the previous example, kernel messages are sent out on the first serial port (note that ttyAM0 is used instead of the familiar ttyS0) at 115200 baud and the root file-system is on the first mtdblock of the flash chip.

On the TS-7200 modem, to load the root file system from the Compact Flash card, the following command should be used instead:

```
exec -c "console=ttyAM0,115200 ip=dhcp root=/dev/hda1"
```

To load a NFS Root file system from a NFS server, use the following command:

```
exec -c "console=ttyAM0,115200 ip=dhcp nfsroot=<IP of NFS server>:</path/to/NFSROOT>
```

Booting to an initrd Image from RedBoot

One may want to instruct the kernel to use an initrd image during the Linux initialization. This might be useful in situation additional hardware support or system configuration in needed prior to root file system load. Using RedBoot, an initrd image can be loaded through three main steps:

1. Load the initrd.gz image into memory (memory location used is 0x0080_0000 or 0x00600000, for example)

```
load -h <server IP> -r -b <memory address>
```

2) Load the kernel into memory (Follow previous intructions for kernel loading)

3) Using the "exec" comannd, instruct RedBood to execute the Kernel and initrd, for example:

```
exec -r 0x0080_0000 -b 0x00218000 -c "console=ttyAM0,115200 root=/dev/ram0 init=/linuxrc rw"
```

2.5 Logging In

After the desired Linux Kernel is loaded and executed through RedBoot, the file system loads and networking, logging, Apache web server, etc. are all started. When the login prompt is displayed, type "**root**" to login, with no password. A Bash login prompt will then appear. At this point, you are ready to enjoy your TS-72XX SBC running Linux.

2.6 Shutdown

Use the "shutdown –h now" command to halt the Linux system when running from Compact Flash, SD or USB memory card to avoid a potentially lengthy file system check on the next boot, since the file system running is EXT2 formatted.

On the other hand, the JFFS/YAFFS file systems are highly tolerant of power cycles while the file systems are mounted. Therefore, the "shutdown" command is not required when the root file system is JFFS/YAFFS, but is still recommended.

3 LINUX FOR ARM OPERATING SYSTEM SUPPORT

The ARM processor (the EP9302) comes from Cirrus and the platform is very similar to the Cirrus EDB9302 evaluation board. Cirrus has strongly promoted running Linux on this chip and has done most of the legwork in creating a patch set to the Linux 2.4 kernels, but we have also had to modify the Linux Kernel (TS-Kernel) so it can support the 8MB on-board Flash chip (via mtd drivers), the compact flash IDE driver, and the A/D converter. If you want to use Linux and aren't tied to the x86 architecture, the TS-72XX boards can be very cost-effective.

The TS-72XX SBCs are shipped standard with the compact TS-Linux embedded operating system installed in the on-board Flash memory. The full-featured Debian Linux can also be used with an NFS root file system or larger Flash drives, such as Compact Flash cards, SD cards and USB flash drives. The TS-Kernel used is based upon the version 2.4.26, patched and compiled for the Cirrus EP9301 ARM920T processor, and is real-time capable through RTAI.

The root file system used by the Linux OS can be any of the following:

- ✔ JFFS/YAFFS file system image in the on-board Flash (RedBoot should include the option root=/dev/mtdblock1 to instruct the kernel to boot here)
- ✔ EXT2 file system image in the Compact Flash card (RedBoot should include the option root=/dev/hda)
- ✔ NFS root, via Ethernet port (RedBoot should include the option root=/dev/nfs nfsroot=<IP>:<DIRECTORY> ip=dhcp)

3.1 TS-Kernel

In order to support the Linux Kernel on TS-72XX ARM SBCs, Technologic Systems added modifications to the Linux Kernel for the EP9301 and EDB9301 platform shipped by Cirrus Logic, manufacturer of the processor. The Linux Kernel used is based upon the version 2.4.26, and additional patches are applied to the main 2.4.26 source tree so that the kernel will support ARM processors (RMK patches), the Cirrus EP9301 processor and, finally, TS-72XX hardware specifics. The final Linux Kernel is a Technologic Systems specific source, therefore called TS-Kernel.

TS-Kernel modifications include drivers for the Compact Flash IDE, for the extra A/D converter chip, for the NAND flash memory devices, MTD drives and YAFFS file system, framebuffer driver for video interface, second ethernet driver for TS-7300, driver support for the SJA1000 CAN controller on TS-CAN1, drivers for the SD Card interface, etc.

Technologic Systems is constantly improving the TS-Kernel so that it will support new ARM SBCs and PC/104 peripheral boards. Complete TS-Kernel source code is provided at Technologic Systems web site, allowing developers to implement modifications and compile their own kernels.

Note
The TS-Kernel supports the **Real-Time** Application Interface (**RTAI** project), making the embedded operating system capable of handling applications with hard real-time restrictions.

For further information on the real-time extension and kernel development procedures, refer to the development section of this manual.

Note
It is also possible to run the 2.6 version of the Linux Kernel on TS-72XX SBCs. However, this development is not officially supported by Technologic Systems yet. Contact Technologic Systems if your solution requires 2.6 kernel version.

3.2 TS-Linux

TS-Linux is a compact Linux distribution, based on Busybox, ideal for small footprint systems and used as a demonstration OS on the TS-72XX computers. Therefore, the TS-72XX SBCs come with TS-Linux installed by default in the on-board flash. RedBoot is also configured to load the TS-Linux from on-board flash by default.

The use of TS-Linux is very similar to other common Linux systems. A experienced Linux user should find no problems when using TS-Linux. The initialization and configuration processes are all managed by the files located in the “/etc” directory. Devices entries are in the “/dev”, while libraries and kernel modules are inside “/lib”. Network services such as FTP and Telnet servers are also available. Even an HTTP server comes installed by default with TS-Linux and is loaded during initialization time.

Most of the Linux utilities you would expect are present on the TS-Linux distribution. In total there are over 120 utilities installed on the file system. Many of the basic utilities are implemented using Busybox. BusyBox combines tiny versions of many common UNIX utilities into a single small executable The utilities in BusyBox generally have fewer options than their full-featured GNU cousins; however, the options that are included provide the expected functionality and behave very much like their GNU counterparts. BusyBox provides a fairly complete environment for any small or embedded system.

Note
In order to keep the distribution small, TS-Linux has small support for the GNU tools. Therefore, a C/C++ development environment is not installed. If the user's main target is development, the Debian Linux distribution is highly recommended A cross-compile environment based on a PC is also an option for development.

3.3 Debian Linux

The Debian is a full-featured and powerful Linux distribution mostly based upon GNU tools. It includes everything necessary to easily run Linux and develop Linux applications. In addition, various original Linux utilities and installation tools are available to make system utilization and packages updating easy. The Debian Linux is ported to ARM processors and runs with TS-72XX SBCs. With Debian Linux, experienced Linux users have a complete Linux system to take full advantage of their knowledge, and new users have an easy environment to get started with the Linux world.

Technologic Systems makes use of Debian Linux as a development distribution. For example, the 256 MB Compact Flash card is pre-installed with Debian. Along with the basic core utilities, some developer tools have been installed, including a native arm gcc toolchain, for C/C++ application development. In addition, the Perl interpreter and a wide variety of network services are available, such as FTP, Telnet, SSH and Apache HTTP servers. One can also use the Debian Linux distribution via an NFS root file system or an USB flash memory device.

apt-get

When using the Debian Linux file system, adding new packages and removing undesired ones is done all through Debian's package management. “apt“, “dpkg“, all behave as expected. With Debian, one can easily install and remove software packages. For a quick demonstration of how easy it is to remove and install programs with Debian, try the following commands:

```
apt-get install hexedit
```

```
hexedit /etc/passwd
^C (hit CTRL+C to safely exit)
apt-get remove hexedit
```

apt-get install installs a package name, while apt-get remove removes the named package. Visit the Debian home-page for further information, since a full in-depth discussion on Debian is outside the scope of this document.

- http://www.debian.org

Note
Technologic Systems supports newer versions of the Debian Linux distribution, such as the 3.0, named Sarge. For newer Debian Linux support with your TS-72XX SBC, contact Technologic Systems.

3.4 TS-ARM Linux CD

The TS-ARM Linux CD is an useful resource for TS-72XX users and developers. It contains the supporting files for Linux, including the TS-Kernel source code, sample applications, cross toolchain tarballs, Linux distributions, manuals, and other Linux utilities and binaries of interest. The files mentioned on the Linux for ARM User's Guide are all included on this CD. The following is the directory structure of the CD:

- **root dir**: Linux for ARM on TS-72XX manuals in PDF format
- **/binaries**: various binary packages for ARM, including Java, QtEmbedded, RTAI, etc
- **/binaries/ts-bitstreams**: default TS-Bitstreams for FPGAs
- **/binaries/ts-images**: ARM binaries of file systems images and other
- **/binaries/ts-kernels**: compiled TS-Kernels for TS-72XX SBCs
- **/binaries/ts-modules**: other drivers (Linux Kernel modules) for TS-72XX SBCs
- **/binaries/ts-scripts**: some Linux shell scripts used with the TS-72XX SBCs
- **/binaries/ts-utils**: some useful utilities distributed within a Linux distribution
- **/cross-toolchains**: pre-compiled GNU toolchain binaries for cross platform development with C/C++
- **/distributions**: binaries of linux distributions ready to install and run on TS-72XX SBCs
- **/docs**: various documents related to Linux for ARM on TS-72XX series, including data sheets and technical articles
- **/manuals**: all the hardware manuals for the TS-72XX series, including schematics
- **/pictures**: TS-72XX product views and related pictures
- **/samples**: source code and ARM binaries of some samples ready to compile and run
- **/sources**: source code of TS-Kernel and other related Linux for ARM packages

The TS-ARM Linux CD is included in the TS-ARM Development Kit and also can be downloaded through Technologic Systems website at:

- http://www.embeddedarm.com/linux/ARM.htm

Note
Notice that the FTP server is frequently updated and will always contain the latest version of the TS-ARM Linux CD content. **ftp://ftp.embeddedarm.com**

3.5 TS-ARM Development Kit

The TS-ARM Development Kit for the **TS-7200** Single Board Computer includes all equipment necessary to boot into the operating system of choice and start working. The development kit is highly recommended for a quick start on application development.

The TS-ARM Development Kit contains a 256 or 512 MB Flash drive (Compact Flash for 7200, USB for 7250 and 7260) which includes:

- ✔ a self-hosting ARM installation of the **Debian Linux** 2.0 distribution compiled for ARM
- ✔ gcc 2.95.4 and gcc 3.0 compiler with full tool-chain
- ✔ Build tools and source for JFFS/YAFFS file system in on-board NAND Flash.
- ✔ Hardware test routines source code and other example source code
- ✔ Debian package system: apt-get, tasksel, dselect

The development kit additionally includes:

- ✔ USB Compact Flash reader for **TS-7200**
- ✔ 5 VDC regulated power supply (international versions available)
- ✔ NULL modem cable
- ✔ Adapter cable from 10-pin header to DB9
- ✔ Various cables for connection DIO, LCD, Keypad, etc.
- ✔ Development CD with complete TS-Kernel source, manuals, example code, etc.
- ✔ Printed supporting documentation for TS-72XX's Hardware, Linux for ARM and Development Kit.

Note
Single Board Computer is not included on the Development Kit (sold separately).

4 CONFIGURING AND USING LINUX

This section provides basic and general information on how to configure and use Linux for ARM on TS-72XX SBCs. The following information is intended to be useful for both TS-Linux and Debian Linux users. However, some of the mentioned utilities may not be available in the compact TS-Linux distribution.

This section is not intended to be a complete Linux Utilities tutorial. For further information, find the proper Linux documentation widely available on the Internet.

4.1 Basic Commands

Some very basic commands for one beginner user to start using Linux are:

- pwd: informs the current directory
- ls: lists current directory contents
- cd: changes directory
- man: accesses the system's manual pages of a given command
- cat: displays the entire content of a given file
- vi: Linux most common file editor (reading further documentation is recommended)

The most common file handling commands are “cp”, “mv”, “rm”, “mkdir”. Help information is provided by supplying “--help” to any given command, for example “cp –help”.

4.2 Users and Passwords

The TS-72XX SBCs come by default with the root user configured with no password. No additional user is added. All the existing users and groups can be viewed in the “/etc/passwd” and “/etc/group” files. The following utilities can be used for users and groups management:

- passwd: changes password for the logged user
- groupadd: add a group of users
- useradd: add a user
- chmod: change access permissions for files and directories
- chown: change owner of files

4.3 System Log and Information

Generally, all the log files of the Linux OS are stored on the “/var/log/” directory. Therefore, important system messages such as the kernel boot or some applications (HTTP server) outputs can be viewed by inspecting the log files. In addition, one can make use of the “dmesg” command to get the latest kernel messages. This might be useful when developing device drivers or watching kernel status.

Further utilities provide useful Linux environment information, such as:

- uname: prints system information such as kernel version and system date
- hostname: prints the hostname of the system
- printenv: prints the user's environment variable
- whoami: prints the current logged user name
- who: prints the current logged users

Further Linux system information of the currently running processes and running Kernel may be obtained through the files at the /proc virtual file system.

4.4 Initialization Scripts

After the “exec” command on RedBoot, the Kernel boots and drivers are loaded. Then, the initialization process reads the file “/etc/inittab”. The inittab file will call “/etc/rc.d/rcS.sysinit” as part of the system initialization. The run level then defaults to 3, which will run the “/etc/rcS” script and call all the scripts linked in the “/etc/rc3.d/” directory in numerical order. For example, the following are the initialization scripts for run level 3 found at TS-Linux:

```
/etc/rc.d/rc3.d# ls
S10Network  S11portmap  S20inetd  S30telnetd  S40apache
```

Changing the run level or re-invoking the initialization scripts is possible through the “init” command. A “halt” or “reboot” command will change the run level to 0 or 6 and execute the “/etc/rc0.d” scripts or “etc/rc6.d” scripts respectively.

4.5 Network Setup

The main utilities for network configuration under Linux are:

- ifconfig: prints network settings and configures ethernet interfaces
- ifup: turns given network interface up
- ifdown: turns given network interface down

Entering “ifconfig” shows the current ethernet settings. These utilities require a network device as parameter. On Linux, the ethernet devices are generally named eth0, eth1, etc. Therefore, the command “ifup eth0” or “ifconfig eth0 up” brings up the on-board ethernet interface on TS-72XX SBCs.

To configure the network, one needs to manage the proper configuration files. On TS-Linux systems, these files are located in the “/etc/sysconfig/” directory. “/etc/network/” is used for Debian Linux. By default, Linux systems on TS-72XX boards are configured to assign the IP 192.168.0.50 to the on-board ethernet interface.

Setting Up the networking with TS-Linux

To configure the network when booting to the TS-Linux image on the flash chip, the files in “/etc/sysconfig/” must be edited. Network interfaces are configured on a file per interface basis. The first Ethernet device, eth0, is controlled by the file “/etc/sysconfig/ifcfg-eth0”. An example of “ifcfg-eth0” is shown below:

```
DEVICE=eth0 #Name of ethernet interface
IPADDR=192.168.0.50 #IP address of this ethernet interface
NETMASK=255.255.255.0 #Used with NETWORK to determine local IPs
NETWORK=192.168.0.0 #Used with NETMASK to determine local IPs
BROADCAST=192.168.0.255 #Broadcast IP for system wide messages
BOOTPROTO=static #Static IP (change “static” to “DHCP”)
ENABLE=yes #Load device on boot
```

The TCP/IP network settings are configured in the file ‘/etc/sysconfig/network_cfg’, here is a listing:

```
NETWORKING=yes #Enable networking on startup
GATEWAY=“192.168.0.1” #Gateway for internet access
GW_DEV=eth0 #Default gateway
HOSTNAME=ts7200 #Host name of this computer
BOOTPROTO=no
FORWARD_IPV4=no
DEFRAG_IPV4=no
```

The TCP/IP name resolution server is configured in ‘/etc/resolv.conf’. Here is a listing:

```
Nameserver 192.168.0.1 #Name server for domain name lookups
```

Those lines starting with a # symbol are comments. As the above example shows, eth0 is given the static address of 192.168.0.50. If one wishes eth0 to obtain its IP from a DHCP server, then change the line BOOTPROTO=static to BOOTPROTO=dhcp

Setting Up the networking with Debian

To configure the network interfaces when booting into Debian Linux, edit the file "/etc/network/interfaces". A typical interfaces file would contain the following:

```
auto lo eth0
# The loopback interface
iface lo inet loopback
# The first network card
auto eth0
#iface eth0 inet dhcp
iface eth0 inet static
address 192.168.0.50
netmask 255.255.255.0
gateway 192.168.0.1
```

Those lines starting with a # symbol are comments. The line "auto lo eth0" means both the loopback interface and the first ethernet interface will be started automatically by the Debian networking scripts. The above example shows that eth0 would be assigned the static address of 192.168.0.50, using 192.168.0.1 as the default gateway. If one was to comment out those lines, and then uncomment the line iface eth0 inet dhcp, then eth0 would use a dhcp client to obtain it's IP and other relevant network information.

Note
In order to test the default network settings with both TS-Linux and Debian Linux, open a web browser and use the embedded Apache web server by entering the default IP 192.168.0.50, or simple "ping" or "telnet" to 192.168.0.50.

4.6 Network Services

Linux is a well known OS suitable for network applications. As such, Linux offers a wide range of networking services which fully integrates the embedded systems to the external world and provides connectivity options for the end-users. TS-Linux and specially Debian Linux include solutions for the main network services, including Telnet (telnet utility), HTTP, FTP (ftp utility), SSH (ssh/scp utilities), NFS and Mail.

Some of these services can be started, restarted or stopped by management scripts located at the "/etc/init.d" directory. For example, the following command will restart the apache server:

```
/etc/init.d/apache restart
```

Also, the "/etc/inet.conf" file is used to configure the initialization and parameters of other network services.

4.7 Setting Data and Time

The Linux Kernel keeps the date and time of the day internally. In order to synchronize the Kernel/System time with an official and precise time source, one can use either the Real Time Clock (RTC) hardware or an NTP time server. System clock synchronization can be done during or after boot time with the help of the following commands:

- ✔ date: prints or sets the system date and time
- ✔ ntpdate: sets the system time according to a remote time server
- ✔ hwclock: sets the system time according to the RTC (if available)

- cal: displays a calendar

On the Debian distribution, the "/etc/init.d/hwclockfirst.sh" and "/etc/init.d/ntpdate" scripts provide system clock synchronization services.

4.8 Configuring and loading kernel modules

Kernel modules are binaries that can be inserted into the Linux Kernel during execution time. It is useful in order to load needed kernel features and services on demand. For example, one can turn on the support for a USB flash memory card by inserting the proper USB modules into the kernel. The main utilities used to handle kernel modules are:

- lsmod: lists the loaded kernel modules
- insmod: insert a given module into the kernel
- modprobe: insert a given module and its dependencies into the kernel
- modinfo: provides information about a kernel module

4.9 Memory

There are some useful utilities to check RAM memory and storage memory status:

- df: shows storage memory usage
- du: shows the total size of a given location
- free: shows RAM memory usage
- top: provides running processes and RAM memory utilization information

4.10 TS-Utilities

Besides the standard Linux set of utilities, the TS-Linux and Debian Linux distributions include Technologic Systems proprietary utilities which perform TS-72XX hardware specific and auxiliary operations. Some of the TS-Utilities are:

- bootload: Linux-based bootloader that enables booting of another Kernel
- busybox: compiled busybox for ARM with lots of Linux utilities
- peek8: reads 1 byte from the given memory location
- poke8: writes the given 1 byte to the given memory location
- peek16: reads 2 bytes from the given memory location
- poke16: writes the given 2 bytes to the given memory location
- peek32: reads 4 bytes from the given memory location
- poke32: writes the given 4 bytes to the given memory location
- ts7xxxctl: configures the hardware and turns on/off parts of the TS-72XX and EP9301
- jbi: programs the on-board CPLD through the JTAG interface using the DIO2 header
- jp: returns status of a given jumper
- hammer: blinks the on-board red led
- load_7kv: loads a hardware core (bitstream file) to the TS-7KV FPGA
- load_7300: loads a hardware core (bitstream file) to the TS-7300 FPGA
- bitblt-demo-ts7kv: performs bit blitting demo operations using the TS-7KV video
- bitblt-demo-ts7300: performs bit blitting demo operations using the TS-7300 video
- tsbootrom-update: updates the EEPROM with various bootroms options
- sdlock: handles SD Card security issues as password lock and write-protected mode

bootload: Linux Bootloader

Technologic Systems has developed a Linux application "bootload" that allows arbitrary booting of Linux and other OS kernels within Linux itself. The "bootload" program allows one to use the full facilities of Linux to retreive kernel files. Doing so also allows the use of standard shell scripts for the programmatic selection of appropriate kernels, Linux initrd's, and kernel command line arguments for maximum flexibility.

The “bootload” program is mostly implemented as a Linux user-space program using C. However, it requires a kernel module "bootloader.o" to be installed using "insmod" prior to invocation. No kernel patches or in-source modifications are required though:

```
insmod -f bootloader.o
```

The Technologic Systems' version of Linux uses a special device driver at /dev/misc/bootloader to accomodate the hooks needed by the "bootload" program to allow Linux to act as a bootloader and boot other Linux kernels and operating systems.

Command usage information for the “bootload” program follows:

```
$ bootload –help
Usage: bootload [OPTION] FILE
Linux to Linux bootloader - (re)boots a TS-7xxx board to another
kernel, OS
image, or raw executable by replacing the running Linux kernel.

General options:
-c, --cmdline=CMD Use CMD as the Linux kernel boot args
-r, --initrd=FILE Use FILE for Linux initial ramdisk
-s, --initrdsz=SZ Only read SZ bytes from the initrd file
-b, --base=ADDR Load the image at ADDR instead of 0x218000
--version Print version and copyright information
-h, --help This help

When FILE is -, reads from standard input.
Report bugs to <support@embeddedARM .com>
```

Some "one-line" examples of usage:

```
# Boot a compressed kernel image:
bunzip -c vmlinux.bin.bz2 | bootload -

# Reboot a kernel, but pass the 1MB running ramdisk to the new:
mount -o remount,ro /dev/rd/0 / bootload -c \
"console=ttyAM0,115200 root=/dev/ram0" -r /dev/rd/0 -s 0x100000

# Boot one of 2 kernels based on the state of DIO line #7:
if dio_data_get 7; then bootload vmlinux.backup.bin; else \
bootload vmlinux.bin; fi
```

tsbootrom-update: Changing/Updating the TS-BOOTROM

The tsbootrom-update is a Linux utility used to update the onboard EEPROM bootup firmware on the, allowing the board to perform different types of boot processes. For example, with the TS-7400, should you wish to actually load the kernel and initrd from an SD card, the default TS-FLASHBOOT bootup program, which boots from NAND flash, must be replaced with TS-SDBOOT using the tsbootrom-update utility. Usage and command line help for this command follows:

```
Usage: tsbootrom-update [OPTION] ...
Updates TS-BOOTROM bootup program stored on EEPROM
```

```
General options:
  -n                    Do not actually write EEPROM
  -s, --sdboot          Write TS-SDBOOT bootup program
  -f, --flashboot       Write TS-FLASHBOOT bootup program
  -u, --burninboot      Write TS-BURNINBOOT bootup program
  -p, --spiflashboot    Write TS-SPIFLASHBOOT bootup program
  -b, --blastboard      Write to blast board EEPROM instead of SBC
  -h, --help            This help

EEPROM security block options:
  -m, --mac=X           Write X as ethernet MAC address
  -l, --verifylen=N     Checksum includes first N 512 byte sectors
  -d, --device=FILE     Use FILE to re-compute checksum value
  -V, --verifydat=N     Use N as pre-computed checksum value
  -L, --lockdat=X       Use X for the SD unlock data token from
                        previous "sdlock --set" command
  -k, --verifylock      Do not boot to an unlocked SD card
  -c, --noconsole       Disable serial console bootup messages

TS-production specific options:
    -a,  --alloc-mac              Get  MAC  address  from  /var/ts-
production/mac
```

The TS bootup programs will by default print a banner message on bootup to the serial port displaying its build date, etc. If you wish to use all serial ports for your application and do not wish to have a serial console, the tsbootrom-update program can be used to silence early bootstrap banner messages so as to not confuse any potential external device UART#0 may be connected to.

Other bootstrap programs are available for-pay from Technologic Systems should you need them and custom ones may be designed for volume customers. For instance, the TS-ETHBOOT bootup program may be used to completely boot Linux from the network without NAND flash or SD card installed.

sdlock: SD flash card security features

Technologic Systems provides a "sdlock" Linux command which can be used to manipulate SD card hardware-enforced password locks and set the card's permanent write-protect feature. Using a password protected SD card is a great way to ensure software security and/or to make sure your TS-72XX SBC based product cannot be used in an unintended matter once deployed.

```
Usage: sdlock [OPTION] ...
Controls SD card lock and permanent write-protect features.

General options:
  -p, --password=PASS   Use PASS as password
  -c, --clear           Remove password lock
  -s, --set             Set password lock
  -u, --unlock          Unlock temporarily
  -e, --erase           Erase entire device (clears password)
  -w, --wprot           Enable permanent write protect
  -h, --help            This help
```

When the TS-72XX SBC is configured with the TS-SDBOOT bootup firmware, the SD unlock password can be stored in onboard EEPROM for automatic unlocking and booting of password protected SD cards. By default, TS-SDBOOT will still boot unlocked cards, but this behavior can be changed with the "--verifylock" option to the "tsbootrom-update"

command described above-- with the "--verifylock" option the TS-72XX SBC will only boot locked SD cards.

TS-SDBOOT contains several features for high security. One feature is the ability to store a checksum of the SD card on the board to verify before bootup. If the checksum fails, the bootup firmware will refuse to boot the inserted SD card. TS-SDBOOT can also verify an arbitrary number of sectors of the SD flash card before allowing bootup. If the stored CRC does not match the actual CRC, the board will refuse to boot and blink the red LED continuously.

Another feature is the ability to boot a password protected SD card. With this, it is possible to make an SD unreadable to any device except the TS-72XX SBC to which it is assigned. Although not directly a function of TS-SDBOOT, an SD card can also be made permanently write-protected through a software command. The combination of these features allows product designers several options on the security of their software and of their deployed TS-72XX SBC based devices.

The various SD commands that manipulate the password lock are marked as "optional" in the SD card specification. This means that not all SD card vendors may implement them in their devices. If they are not implemented, you will not be able to set the SD lock with the "sdlock" command.

4.11 TS-Scripts

Besides the standard Linux set of utilities, the TS-Linux and Debian Linux distributions include Technologic Systems proprietary utilities which perform TS-72XX hardware specific and auxiliary operations. Some of the TS-Utilities are:

- ts7kvfb: script that loads the framebuffer driver and linux console for TS-7KV
- loadUSBmodules.sh: loads modules for USB flash memory cards support
- loadUSB.sh: mounts the USB flash card and changes root file system to it
- ts7xxx.subr: various shell functions to handle hardware, including bits and peekpoke

ts7xxx.subr: Shell Functions for Hardware Access

The ts7xxx.subr shell functions enable control to some of the TS-72XX hardware functions without having to write/compile C code. These shell functions allow you to access various functions such as DIO, ADCs and so forth from the command line or in your own shell scripts.

There are two new commands added to the provided TS busybox to support these shell functions. These commands are "peekpoke" which is used to provide direct access to memory, and ADCread, which will provide a continuous stream of analog to digital readings (either on one channel or all channels) to stdout in text format. Both of these commands provide more info on what options are available if you type them without any arguments.

- dio_dir_get <n>: get the current direction (0=input,1=output) of DIO #n (n in 0..15)
- dio_dir_set <n> <dir>: set the direction of DIO #n to the specified direction (0=input,1=output)
- dio_data_get <n>: get the current value (0=Low,1=High) of DIO #n
- dio_data_set <n> <val>: set the current value (0=Low,1=High) of DIO #n (specified pin should be set to be an output already)
- usb_init: call this function to turn the USB ports on and initialize them
- usb_off: call this function to turn off the USB ports
- usb_numports: prints the number of USB ports on the board. this may be greater then
- the actual number of physical ports. the first port is port 0

- usb_port_devexists <port>: prints '1' if something is plugged into the specified port, or '0' otherwise.
- eth_off: turns off the ethernet port and phy
- eth_on: turns on the ethernet port and phy
- led0 <state>: sets the state of the green LED (0=OFF,1=ON)
- led1 <state>: sets the state of the red LED (0=OFF,1=ON)
- cpu_speed_max: set the CPU and bus speed to their maximum (normal) values, e.g. 200Mhz/100Mhz.
- cpu_speed_166: set the CPU speed to 166Mhz and the bus speed to 66Mhz.
- cpu_speed_42: set the CPU speed and bus speed to 41.5Mhz. This is the slowest speed that Ethernet will work at.
- cpu_speed_min: set the CPU and bus speed to their minimum values (14.7456Mhz)
- temp_read: returns the current Celsius temperature reading from the TMP124 chip.

5 STORAGE DEVICES AND FILE SYSTEMS

This section describes the storage memory devices and the file system types used with TS-72XX SBCs. Occasionally, one may wish to boot into one device and access the other. This requires knowing what those devices are and where they are.

5.1 On-board Flash File Systems

The on-board flash contains a custom-made JFFS or YAFFS formatted file system image. JFFS2 is a compressed, Journaling Flash File System, for the NOR Strata Flash memory on TS-7200. The YAFFS file system is used for the on-board flash on the TS-7250 and TS-7260, which utilize NAND flash technology.

The NOR and the NAND flash devices are divided in three partitions. The first is a 16KB sized partition that contains the TS-BOOTROM. The last partition, which occupies 2MB-3MB, contains the eCos/RedBoot system and the Linux Kernel. The remaining space in the middle partition is either for the JFSS or the YAFFS image used as the root file system. The following shows how a 8MB NOR chip is recognized by the Linux Kernel MTD system on a TS-7200:

```
TS-7200 flash: Found 1 x16 devices at 0x0 in 16-bit bank
 Intel/Sharp Extended Query Table at 0x0031
Creating 3 MTD partitions on "TS-7200 flash":
0x00000000-0x00020000 : "TS-BOOTROM"
0x00020000-0x00620000 : "Linux"
0x00620000-0x00800000 : "RedBoot"
```

In both NOR and NAND cases, the Linux block device entry corresponding to the on-board flash is the “/dev/mtdblock”. Therefore, the following command can be used to mount the flash file system if you are not booted to the on-board flash:

```
mount /dev/mtdblock/1 /mnt
```

Updating the JFFS Image

The JFFS2 image can be created on your host computer, with binaries created via a cross compiler, then placed in a directory structure on the host computer. A new JFFS2 image can be constructed from that directory structure with the following command:

```
mkfs.jffs2 -p6291456 -e131072 -o /path/to/new_jffs.img
```

Pre-made JFFS images can be found in the Developer's CD or on Technologic Systems' web-site. The JFFS image file can then be copied over to the Single Board Computer and then written to the flash chip by using the following command:

```
dd if=/path/to/new_jffs.img of=/dev/mtdblock/1
```

Updating the YAFFS Image

To update the YAFFS image, erase the file system completely (either with “rm -fr” or “eraseall /dev/mtd/1”) and extract your new image onto the mounted file system. This way, YAFFS will auto-discover the bad sectors as you are extracting and work-around them.

The first step is to set up an NFS root file system and boot to it. Then, erase the second partition of your on-board flash using the char device, not the block device.

```
eraseall /dev/mtd/1
```

Mount the second partition of your on-board flash and extract the new filesystem:

```
mount /dev/mtdblock/1 /mnt
tar xzvf /path/to/7250_root_fs.tar.gz -C /mnt
```

Unmount the second partition of your on-board flash and reboot:

```
umount /mnt
shutdown -r now
```

Note
The YAFFS file system runs on the TS-72XX boards that feature NAND chips with 512 byte page size, enabling up to 128 MB of on-board flash. The YAFFS2 file system supports the new NAND technology, with 2k page size, hence it will be installed on TS-72XX boards that are configured with 128-256 MB of flash.

5.2 Flash Memory Cards

Compact Flash

The format of the CF must be in EXT2 for proper operation with Linux as a root file system. The CF card is seen as the Primary Master IDE device, otherwise known as "/dev/hda". The following command demonstrates mounting the Compact Flash card to the currently running system:

```
mount /dev/ide/host0/bus0/target0/lun0/part1 /mnt
```

SD Card

SD Card sockets are available on the TS-7260 and TS-7300 models. Technologic Systems has written a binary Linux driver module and a set of generic, OS-independent read/write routines for accessing the SD flash inside of an ARM object (.o) file. After the SD Card module is loaded, the device entry "/dev/sdcard0" is used to mount into the file system.

```
mount -t ext2 /dev/sdcard0/disc0/part3 /mnt
```

By default, the SD Card driver will be loaded during bootup time. However, in some cases the driver may not be loaded if a SD Card is not present in the socket. Therefore, it is necessary to call insmod for SD Card driver insertion. For example, In order to use a second SD Card on the TS-7300, the following command could be used to load the driver module prior to mount the filesystem:

```
insmod sdcard.o -o sdcard1 io=0x72000020 name="sdcard1"
```

After plugging the card into the socket and inserting the second SD Card driver, the device entry will appear at /dev/sdcard1" and the proper mount command can be used.

USB Flash Drives

USB Flash drives can supply additional storage that behaves very much like a CF card except that a Compact Flash card can be the root drive whereas a USB drive can not be the root file system without using an initrd and appropriate kernel. Drivers are available in the Linux distribution to support USB Flash Drives.

After booting into the TS-Linux OS, one can 'boot' into the Debian OS hosted on a USB thumb-drive with two scripts found on the TS-Linux distribution. First, invoke "/usr/bin/loadUSBModules.sh", then run the script "/usr/bin/loadUSB.sh" to change root into the Debian OS on the USB thumb-drive to take full advantage of such things as the Debian Arm toolchain.

The script "loadUSBModules.sh" loads the following Kernel modules, in the right sequence: usbcore, pcipool, usb-ohci, usb-ohci-ep93xx, scsi_mod, sd_mod, usb-storage.

In addition, it is possible to use an initial ram disk and a kernel that supports USB internally to boot to the USB file system on system initialization time. The kernel is loaded and then the initial ram disk script loads the USB modules and "pivot_root" to the USB file system.

For those who simply wish to load in the correct device drivers to access the USB thumb-drive when booted into the Linux OS, invoke the “/usr/bin/loadUSBModules.sh” script, and then use the following command to mount the file system:

```
mount /dev/scsi/host0/bus0/target0/lun0/part1 /mnt/
```

Note
The TS-7200 always needs to be powered-off before swapping Compact Flash cards. USB Flash drives and SD cards can be hot swapped.

Updating the EXT2 Image

To create a CF/SD/USB flash card image from scratch, one must format the entire memory card as ext2, then unpack the file system (the Development Debian file system for NFS root or memory cards, for example). Latest versions of our pre-made Debian install can be found on our website or on the TS-ARM CD.

This should all be done from the host PC running Linux. The walk-through below assumes the memory card is plugged into the USB dongle and the memory card has been assigned to /”dev/sda”:

```
fdisk /dev/ide/host0/bus0/target0/lun0/disc
```

1) d (to delete existing partitions. Repeat for all partitions)
2) n (for new partition)
3) p (for Primary partition)
4) 1 (make the new partition primary number 1)
5) hit the enter key for the default starting cylinder
6) hit the enter key again for the default last cylinder
7) p (to print out the partition table)
8) If the first partition does not have a star in the boot field, then enter 'a' at the prompt and then '1' to make the first partition bootable.
9) To commit these changes to the disk, enter 'w' to write out the new partition table to the disk.

Now that the memory card has been partitioned, it must be formatted. The following command will format the first partition on the memory card as an ext2 file-system.

```
mkfs.ext2 /dev/sda1
```

All that is left is to mount the memory card and unpack the tar file of the Development Debian File System.

```
mount /dev/sda1 /mnt
tar -C /mnt -xvjf /path/to/debian-file-system.tar.bz2
```

The filename may change as updates are made and posted on the CD or on the website. A simple fsck will ensure file-system integrity.

```
fsck /dev/sda1
```

Note
Compact Flash and SD Cards must use EXT2 format for file system. USB drives can use other formats, such as VFAT.

5.3 Network File System - NFS

NFS uses the Ethernet connection to mount the root file system for the SBC as an exported directory on a Linux host PC. This bypasses the need to download any files from the host to the SBC, because any file needed on the SBC is just copied to the exported directory on the host system and is instantly available to the embedded PC. Mounting the root file system via NFS allows the developer to use the editors, compilers, etc. from the single board computer.

Mounting NFS roots requires that the ”portmap” daemon is running before executing the mount command. The following example demonstrates mounting an NFS file system hosted on a server at 192.168.0.1

```
portmap &
mount -t nfs 192.168.0.1:/path/to/nfsroot /mnt
```

Setting Up an NFS File System

To mount the root file system via NFS, Prepare the exported root directory on the NFS server:

1) Download the pre-made Debian tarfile from our website or the TS-ARM Linux CD to your Linux host machine.

2) Untar NFS root package to a directory on the host machine.

```
tar -C /path/to/nfsroot -xvjf /path/to/debian-file-system.tar.bz2
```

3) Export the directory by adding the following line to the file “/etc/exports”

```
/path/to/nfsroot 192.168.0.0/255.255.255.0
(rw,no_root_squash,insecure)
```

Note: The IP mask in the above example will only allow NFS connections from computers having an IP starting with 192.168.0. Substitute the appropriate parameters for your local network.

4) Restart NFS so the directory is available for export. Typically, this would be:

```
/etc/init.d/nfs-server restart
```

5) Modify the /path/to/nfsroot/etc/fstab file on your host system for your local network settings.

```
Add: "192.168.0.1:/path/to/nfsroot / nfs exec,dev,suid 1 1"
Comment out: "/dev/hda1 / ext2 defaults 1 1"
```

6) Load the kernel from Redboot with the following command line options:

```
fis load vmlinux
exec -c “console=ttyAM1,115200 ip=dhcp
nfsroot=192.168.0.1:/path/to/nfsroot”
```

6 TS-KERNEL AND DRIVERS DEVELOPMENT

This section explains how to configure and compile the TS-Kernel. In addition, useful information about TS-72XX driver support is provided.

6.1 Compiling the TS-Kernel

Technologic Systems provides the TS-Kernel source code for Linux Kernels that are shipped with TS-72XX boards. Please visit our website to download the appropriate TS-Kernel source or use the TS-ARM Linux CD. The officially supported TS-Kernel is based upon the version **2.4.26** and patches for the ARM architecture and for the Cirrus EP9301 processor. A unique version number follows the name of the TS-Kernel in order to identify Tehcnologic Systems releases. For example:

```
$ uname -sr
Linux 2.4.26-ts11
```

In order to compile the kernel from a non ARM host machine, a cross compiler is required. Technologic Systems offers some pre-compiled cross compilers, such as Dan Keegel's CrossTool. To compile the kernel, simply edit the top level Makefile and ensure that the CROSS_COMPILER variable is equal to your system's cross compiler prefix to gcc. For example:

```
CROSS_COMPILE=/usr/local/opt/crosstool/armlinux/gcc3.3.4glibc2.3.2
/bin/arm-linux-
```

The next step is TS-Kernel configuration. The user can enable/disable the compilation of the TS-Kernel features by typing:

```
make menu_config
```

Optionally, Technologic Systems provides default configuration files for the most common TS-72XX SBCs inside the TS-Kernel source tree: The “-ts10” release contains the following default configuration files:

```
# pwd
/usr/src/ts-kernel/linux24/arch/arm/def-configs
# ls ts*
ts7200  ts7250  ts7250_2k  ts7260  ts7300
```

To configure the TS-Kernel for the default TS-7200 setup, the following can be used:

```
make ts7200_config
make oldconfig
```

Generally, the next compilation steps include dependencies processing, kernel image creation and kernel modules compilation:

```
make dep
make vmlinux
make modules INSTALL_MOD_PATH=/lib/modules
```

The last step is modules installation:

```
make modules_install
```

The resulting modules and vmlinux files can be placed onto the flash chip, compact flash card or NFS root directory, as described before. RedBoot can be configured to load the new kernel into the TS-72XX SBC.

Different TS-Kernel 2.4 Release Versions

- **ts8**: Added framebuffer support for TS-7KV
- **ts9**: Added driver supprot for TS-7260 (tsuart serial driver)
- **ts10**: Added framebuffer support for TS-7300 and Added driver supprot for TS-7300 (tsuart73 serial driver, sdcard driver, open_eth ethernet driver)
- **ts11**: Minor bugs fixed and support for TS-7400 added

In order to avoid confusion and version incompatibility problems with kernel modules, Technologic Systems has decided to use release dates for version control instead of sequential numbers, which are no longer used. The ts11 version is the latest one.

Kernel 2.6 support

The Kernel 2.6 is not officially supported by Technologic Systems. Some customers and users are using Kernel 2.6 with TS-72XX boards though. One can find further information along with installation instructions and packages through the **TS-72XX mailing list**.

6.2 Real Time Extension for TS-Kernel

Real time systems are special applications that have to respect specific timing requirements and must be precise in their periodicity (small jitter) and/or be very fast in serving real time events such as interrupts or IO signals (small latency). Hard real time applications require latency and jitter accuracy of less than 10us (microseconds).

The Adaptative Domain Environment for Operating Systems (Adeos) implements a layer between the hardware and the OS that allows multiple OSs sharing the same hardware while running simultaneously. A software pipeline is used to service interrupts and dispatch them to the proper destination; one of its applications is real time. Real Time Application Interface (RTAI) is a set of kernel modules that provide real time services such as interprocess communication; and an API for real time application development. RTAI services run under the Adeos kernel, so it is necessary to build and run Adeos first. The Adeos kernel is based on the Linux kernel, providing the ability to make it fully preemptable.

Technologic Systems provides Adeos/RTAI solutions integrated with the TS-Kernel. RTAI source code and compiled RT-TS-Kernel and RTAI packages are available through Technologic Systems website or the TS-ARM Linux CD. The RTAI version used is the magma/vulcano 3.2.

Installing the Real Time environment

Before loading and running the real time kernel, you should copy the necessary files to the file system in use on the target. These files include the Linux modules for the new kernel, RTAI service modules, test files and documentation.

Technologic Systems provides a tarball containing the full RTAI compiled environment for the "ts9-rt" RT-TS-Kernel (patched with Adeos). The file "/binaries/rtai-3.2-magma-compiled.tar.gz" is distributed with the TS-ARM Linux CD. Change directory to "/usr" and untar this file to create the "realtime" directory with the RTAI environment:

```
tar zxvf rtai-3.2-magma-compiled.tar.gz -C /usr
```

The RTAI services are provided through the modules inside the "modules" directory. Use "insmod -f" to install the RTAI services. The "ts9-rt" kernel modules can be installed using the "/binaries/ts-modules/rt-tskernelmodules-2.4.26-ts9.tar.gz" file available on the CD. The "testsuite" directory contains RTAI sample applications. In order to salve save storage memory, one can remove the documentation directory.

Running the Real Time TS-Kernel

Technologic Systems provides compiled real time TS-Kernels for the TS-72XX SBCs. Those can found on the TS-ARM Linux CD, in the “binaries/ts-kernels” directory. For example, the “rt-vmlinux-7200-ts9.bin” is a compiled TS-Kernel version “ts9”, patched with Adeos and configured for TS-7200.

After selecting the proper compiled real time TS-Kernel, you can load it using Redboot’s “load” command and then execute the kernel by using “exec” command, as described in the RedBoot section.

Building the Real Time TS-Kernel and RTAI

It may be of interest to build your own real time TS-Kernel and RTAI system from source code for your specific TS-72XX SBC. Technologic Systems provides the following files through the TS-ARM Linux CD in order to help with that:

- ✔ /source/rt-tskernel-2.4.26-tsX-adeos.patch: Adeos patch for TS-Kernel
- ✔ /source/rtai-3.2-magma-src.tar.gz: RTAI source tree used by TS
- ✔ /source/rtai-examples-src.tar.gz: some RTAI sample applications and example code

Once you have a TS-Kernel source tree installed and the proper cross-compile environment configured, you can clean and patch your kernel using the Adeos patch:

```
cd /path/to/ts-kernel-directory
make distclean
patch -p1 < /path/to/rt-tskernel-2.4.26-tsX-adeos.patch
```

This is install the Adeos files and modifications to the TS-Kernel, enabling it to be used with RTAI on a real time system.

After the patch is successfully applied, the next step is kernel configuration. You can still use the default ts-kernel configuration files for TS-72XX SBCs. Make sure to select the **Adeos Support** to builtin option (not as a module), in the General Setup section of the kernel configuration. This will enable the Adeos system to be compiled inside the TS-Kernel. Save your configuration and proceed with the common steps on kernel compilation:

```
make oldconfig
make dep
make vmlinux
make modules
make modules_install INSTALL_MOD_PATH=/lib/modules
```

After you have suceed in the real time TS-Kernel building process, the next step is RTAI system compilation. For that, install the supplied RTAI source file and configure it:

```
tar zxvf  rtai-3.2-magma-src.tar.gz -C /path/to/rtai
cd /path/to/rtai
make ARCH=arm CROSS_COMPILE=/path/to/cross-compile/bin/arm-linux-
```

An RTAI configuration environment will show up. Make sure to provide the correct Adeos TS-Kernel build directory you used when filling the “Linux Build” option. Configure the RTAI system regarding the features and services you want. Finally, save and exit the configuration system to strat the compilation. The final step is the RTAI installation:

```
make install
```

This is will install the RTAI modules and environment to the configured installation directory, which must be copied to the root file system in use by your TS-72XX SBC.

6.3 Serial

The TS-Kernel contains drivers for all the on-board serial drivers. Driver's source code is also available within the TS-Kernel source tree.

COM1 and COM2

The Linux devices on the filesystem that should be used with the COM1 and COM2 serial ports are /dev/ttyAM0 and /dev/ttyAM1, respectively. Any Linux terminal emulator, as minicom, can be used with these ports. Simple echo and cat commands can also be used in order to test these ports:

```
cat < /dev/ttyAM1
echo > /dev/ttyAM1
```

RS-485 Support on COM2

A special ioctl command has been added to the Linux kernel's serial code to turn off and on the automatic RS-485 feature. The header files needed to be included from the Linux kernel source tree on an application source code are “linux/ts_sbc.h” and “linux/include/asm/ioctls.h”. These header files can be found in the TS-Kernel source tree. The following snippet of code demonstrates the use of this ioctl.

```
#include <linux/ts_sbc.h>
#include <linux/include/asm/ioctls.h>

#define TIOC_SBCC485 _IOW('T',0x70,int) /*TS RTS/485 mode Clear*/
#define TIOC_SBCS485 _IOW('T',0x71, int) /*TS RTS/485 mode Set */
#define AUTO485FD 1
#define RTSMODE   2
#define AUTO485HD 4

mcr = AUTO485FD;
//mcr = AUTO485HD; //for half duplex
ioctl (fd, TIOC_SBCS485, &mcr);
//write() and read() from fd
ioctl (fd, TIOC_SBCC485, &mcr);
//further reads() and writes() may not behave
```

TS-7260 2TTLCOM Option

The TS-7260 has 3 serial ports at RS232 levels (COM1, COM2 and COM3 headers) by default. When the 2TTLCOM option is enabled, extra 2 serial ports appears at the DIO2 header, using TTL levels, and the COM3 will also use TTL levels.

The “tsuart.o” module implements TS-Kernel support for the COM3 port and the 2 extra TTL serial ports. It is distributed inside the TS-Kernel tree, “ts9” version. The insertion of the “tsuart.o” module inside the TS-Kernel causes the creation of the device entries “/dev/ttyTS0”, “/dev/ttyTS1” and “/dev/ttyTS2” for COM3, COM4 and COM5, respectively.

An special file at the /proc virtual file system, “/proc/driver/tsuart/com3ttl”, is available in order to control the COM3 signal level. The user can set COM3 to TTL levels as follows:

```
echo "1" > /proc/driver/tsuart/com3ttl
```

The user can set COM3 to RS-232 levels as follows:

```
echo "0" > /proc/driver/tsuart/com3ttl
```

The user can query the current status of COM3 as follows:

```
cat /proc/driver/tsuart/com3ttl
"0" if COM3 is at RS-232 levels
"1" if COM3 is at TTL levels
```

TS-7300 Serial Ports

The "tsuart73.o" driver available in the TS-Kernel tree, version "ts10" implements support for the eight additional UARTs implemented on the on-board FPGA. After the "tsuart73.o" driver is installed, four device files are created in the "/dev" directory for each serial port, as follows.

Port	tty 8-bit	cua 8-bit	tty 9-bit	cua 9-bit
1	ttyTS0	cuats0	ttyT9S0	cuat9s0
2	ttyTS1	cuats1	ttyT9S1	cuat9s1
3	ttyTS2	cuats2	ttyT9S2	cuat9s2
4	ttyTS3	cuats3	ttyT9S3	cuat9s3
5	ttyTS4	cuats4	ttyT9S4	cuat9s4
6	ttyTS5	cuats5	ttyT9S5	cuat9s5
7	ttyTS6	cuats6	ttyT9S6	cuat9s6
8	ttyTS7	cuats7	ttyT9S7	cuat9s7

To use the port in 8-bit mode, open the corresponding ttyTS or cuats file. The opened file can be read or written byte wise as expected. To use the port in 9-bit mode, open the corresponding ttyT9S or cuat9s file (e.g. /dev/ttyT9S0). In this mode the length of all reads and writes should be a multiple of two. Data read and written is packed into a character array in big-endian format, using two bytes for every 9 bit value (nontet). The following example C snippet illustrates a simple read and write:

```
...
FILE *f;
unsigned char buffer[4];
int data;

f = fopen("/dev/ttyT9S0","r+");
buffer[0] = 0x01;       // set bit 8 of first nontet
buffer[1] = 0x55;       // lower 8-bits of first nontet
buffer[2] = 0x00;       // msB of second nontet, bit 8 is cleared
buffer[3] = 0xAA;       // lsB of second nontet
fwrite(buffer,1,4,f);

fread(buffer,1,4,f);
data = buffer[0] << 8 + buffer[1]; // extract first nontet
data = buffer[2] << 8 + buffer[3]; // extract second nontet
...
```

In the above case, buffer[0] and buffer[2] are the most significant bytes (msB) of their respective nontets, and buffer[1] and buffer[3] are the least significant bytes (lsB).

Each serial port can only have one device file open on it at a time. This ensures that 8-bit and 9-bit mode are mutually exclusive and that the byte stream remains synchronized when switching modes.

When reading, the upper 7 bits of the msB will always be zero. This fact can be used for synchronization should the current state of the byte stream be unknown. If the upper 7 bits of a given byte are not all zeros, then the byte is not the msB.

Likewise when writing, the upper 7 bits of the msB must always be set to zero. The serial port driver will reject bytes which do not have the msB upper 7 bits set. This can be used to ensure that synchronization is maintained if the current synchronization state is unknown. For example, if 4 bytes of 0xFF are written, regardless of what state the serial

transmission is in, at most one 0xFF will be sent and then the driver will start skipping invalid msBs. As a result, the first byte of data sent after the 0xFF sequence with the upper 7 bits clear will be considered the new msB.

Normally these synchronization methods should not be necessary, as the port, when opened, is synchronized such that the first byte read and written will be considered the msB. However for debugging, diagnostic use, or recovery from application errors these features may be useful.

6.4 CAN Bus

The Linux device driver for a Philips SJA1000 CAN Controller was developed using the OCERA framework for CAN, which is composed of LINCAN, VCA, CAN/CANOpen Monitor and CANOpen Device. LINCAN is a generic Linux device driver implementation that supports many boards, including **TS-CAN1, TS-7KV** and **TS-732**. It is an evolution of other CAN open source projects for Linux.

Loading the Lincan driver

The Lincan driver is included in the TS-Kernel modules, version “ts9”, as a kernel module named “lincan.o”. Also, it is available through the Technologic Systems website or the TS-ARM Linux CD.

The Lincan driver implements a plug and play system. So, the driver is able to detect the hardware, I/O address and IRQ line being used. To take advantage of the implemented plug and play feature, load the driver into the kernel by executing the command below:

```
cd /path/to/lincan/modules/
insmod lincan.o hw=tscan1    #for TS-CAN1
insmod lincan.o hw=ts7kv     #for TS-7KV
insmod lincan.o hw=ts732     #for TS-732
```

After the Lincan driver is installed, the respective device entries will appear on the Linux filesystem at /dev/can*.

Testing the CAN network

You can use the testing tools available at "/path/to/lincan/binutils" directory. For example, to set up your CAN hardware to listen to messages from a working CAN network, use the following command:

```
cd /path/to/lincan/binutils
./readcan /dev/can0
```

To send messages out to a CAN network, you can use the sendburst tool. For example, to send out #3 can messages with identifiers #1 on each #1 second of period, do:

```
./sendburst --help
./sendburst -d /dev/can0 -w1 -b3 -i1
```

6.5 Video

Technologic Systems provides a framebuffer driver that handles the TS-VIDCORE video interface on the TS-7KV or TS-7300. The video registers are properly managed inside the driver so other Linux Kernel's layers can interact with the TS-VIDCORE using the Framebuffer Device API. The driver enables video interaction from user space using the entries at /dev/fb. The Linux Framebuffer driver is integrated in the official TS-Kernel release, version higher than ts9.

6.6 Ethernet

Driver support for the fast on-board EP9301 MAC core is compiled inside the TS-Kernel. The TS-Kernel also supports the 10/100 OpenMAC ethernet core, from opencores.org,

through a kernel module. Source code is available for both MAC interfaces through the TS-Kernel source tree.

The OpenMAC is integrated on the TS-7300 on-board FPGA and the “openeth.o” is the ethernet driver that must be loaded in order to support the OpenMAC core with the TS-Kernel, vesrion “ts-10”. After loading the “openeth.o” module, the “eth1” Linux network interface must be used along with the “ifconfig” command in order to configure the OpenMAC ethernet interface.

6.7 USB WiFi 802.11g Dongle

The TS-Kernel supports USB WiFi 802.11g devices based on the zd1211 chipset. The module binary is available at:

- ftp://ftp.embeddedarm.com/ts-arm-sbc/ts-7200-linux/binaries/ts-modules/wireless-zd1211/zd_b.o

In order to bring up a wireless network, one needs to load the required modules, use ifconfig to brings wlan0 up and then use iwconfig to associate the interface to a wireless provider. The following command sequence can be used:

```
insmod usbcore
insmod pcipool
insmod usb-ohci
insmod usb-ohci-ep93xx
insmod scsi_mod
insmod sd_mod
insmod zd_b.o
ifconfig wlan0 up
iwconfig wlan0 essid <ID>
ifconfig wlan0 <IP>
```

Other Linux tools for wireless networking can be used, such as iwlist, iwgetid, iwpriv, iwspy. They all are available through the TS-ARM-Linux-CD and TS ftp server.

6.8 USB Mouse and Keyboard

In order to use a USB Keyboard and a USB Mouse with your TS-72XX SBC, it is necessary to load the following modules into the running kernel:

```
insmod pcipool.o
insmod usbcore.o
insmod usb-ohci.o
insmod usb-ohci-ep93xx.o
insmod input.o
insmod hid.o
insmod keybdev.o
insmod mousedev.o
```

7 APPLICATION DEVELOPMENT

This section introduces the basics of application development with TS-72XX ARM computers. It is possible to develop using just the embedded system or by using a host PC and cross-compilation tools.

7.1 GNU Tools

Linux application development generally means development using the GNU tools set. The GNU project provides a wide range of development utilities, such as compilers, build tools and libraries. The Debian Linux distribution provided by Technologic Systems as a full featured development Linux OS, contains a native GNU installation for ARM, enabling embedded application development using only the embedded platform. The most commonly used development tools are:

- gcc: gnu c compiler
- g++: gnu c++ compiles
- ld: gnu linker
- make: tool for the compilation process automation

As an alternative, it is possible to use a cross compile toolchain, enabling binary generation for the ARM architecture from a x86 host architecture, for example.

7.2 Development on the Target with Debian Linux

The typical way of doing Embedded Linux development is actually on the board itself. Since the TS-72XX ARM9 CPU is a PC-class processor in everything but power consumption and performance, it has no problem running real PC-class operating systems such as Linux. By running the full version of Linux (and not scaled-down microcontroller project OS's such as ucLinux), the TS-72XX SBCs can run the entire suite of applications contained in the Debian Linux distribution including the compilers. Since almost every open source program available for Linux is contained within the Debian Linux binary distribution, one rarely has to compile the large code-bases that would otherwise have forced integrators to a complicated cross-compilation environment due to the limited RAM/Mhz of the embedded computer. All too often, open-source projects do not anticipate the possibility of cross-compilation in their build systems, leaving issues for the system integrator to resolve.

The default Debian Linux distribution provided by Technologic Systems contains compilers and everything needed for developing applications in C, C++, PERL, PHP, and SH. Java, BASIC, TCL, Python and others are also available for Debian, but not installed by default. More information on using and configuring Debian Linux can be found at:

- http://www.debian.org

One can still use cross-compilers hosted on just about any platform if there is a specific need. See the next section for more information on Cross Toolchains.

7.3 Development on the Host with a Cross Toolchain

While the Debian Linux includes a suite of compiler tools, including a native arm gcc c compiler, you may wish to use your desktop PC for compiling and development. CrossTool 0.28 or greater now supports Cygwin. Technologic Systems provides pre-made versions of crosstool for both Linux and Cygwin. They can be found on the Arm Development CD or on the Technologic Systems' web-site.

To install the Linux binaries, unpack the tar file at the root of your system as the root user. To install the Cygwin binaries, unpack the tar file at the root of your Cygwin environment. Be sure to include the path of the crosstool binaries in the PATH environment variable or in the proper Makefile. For example, the pre-made Linux crosstool binaries are located at “/usr/local/opt/crosstool/arm-linux/gcc-3.3.2-glibc-2.3.2/bin/”

Cygwin Hello World walkthrough

This is a quick demonstration on using Windows and Cygwin to create your own hello world application using Linux tools. It is assumed that either the Cygwin crosstool tar file has been unpacked to the root of the Cygwin environment or has been built from scratch. Naturally, it is assumed that Cygwin has been installed on your Windows PC.

1) Create hello-world.c

This can be done with any editor, either using Vim from within Cygwin, or using notepad from within Windows

```
include <stdio.h>
int main (void)
{
printf ("Hello World!\n");
return 0;
}
```

2) Ensure that the CrossTool binaries are in your systems path:

```
export    PATH=$PATH:/opt/crosstool/arm-unknownlinux-gnu/gcc-3.3.2-glibc-2.3.2/bin
```

3) Create the hello executable:

```
arm-unknown-linux-gnu-gcc -Wall -o hello helloworld.c
```

4) Verify that the resulting hello binary is an ARM binary file, using the file command:

```
file hello
```

5) Run inetd on the SBC:

```
inetd
```

6) Copy over the hello binary to the SBC via ftp:

```
ftp 192.168.0.50 (be sure to use the IP that matches your setup)
Log in with user root with no password
enter "binary" to ensure binary transfers
enter "put hello"
```

7) On the SBC, change the permissions of the hello binary to become executable:

```
cd /root
chmod +x ./hello
```

8) Run the binary

```
./hello
```

7.4 Programming IO Devices

When programming Linux applications that handle hardware devices on the TS-72XX, it is important to understand the Memory Map of the EP9301 processor and additional hardware features. Each hardware device or functional component has its reserved memory address space, where the specific management registers are located.

Device drivers implementation is mostly reading and writing operations to specific memory registers. This is accomplished from the Kernel space through straight access to the physical memory. The resulting device driver provides high level procedures to the user so that one is able to talk to the hardware without knowing memory map, bits and such. Linux applications run in a protected and separate environment where they can do no damage to either the kernel or other applications running simultaneously. This protected environment does not allow arbitrary manipulation of hardware registers by default.

It is also possible to talk to hardware devices from user space. In doing so, one does not have to be aware of the Linux Kernel development process. The special "/dev/mem" device implements a way to access the physical memory from the protected user space, allowing readings and writings to any specific memory register. Applications may be

allowed temporary access through memory space windows granted by the mmap() system call applied to the /dev/mem device node. For instance, to set up access to the GPIO registers at 0x12c00000, the following snippet of C code is provided as an example:

```
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

{
  int fd = open("/dev/mem", O_RDWR|O_SYNC);
  char *gpioregs;
  gpioregs = (char *)mmap(0, 4096, PROT_READ|PROT_WRITE,
     MAP_SHARED, fd, 0x12c00000);
  gpioregs[0] = 0xff; /* directions register set to all outputs */
  gpioregs[2] = 0x12; /* output 1 to DIO_01 & DIO_4, all else 0 */
}
```

Some notes about the preceding code:

- Make sure to open using O_SYNC, otherwise you may get a cachable MMU mapping which unless you know what you're doing, probably is not what you want when dealing with hardware registers.
- mmap() must be called only on pagesize (4096 byte) boundaries and size must at least have pagesize granularity.
- mmap() of /dev/mem is only allowed for processes with UID 0 (root)
- More information on mmap() and open() system calls can be had by running "man mmap" or "man open" from most any Linux shell prompt.
- When working with char * types to registers, make sure to compile with the "-mcpu=arm9" option otherwise the wrong ARM opcodes will be used and your byte reads/writes may be turned into 32-bit reads and writes

7.5 Applications Debugging

There are two main methods for debugging on the TS-72XX SBC: Debugging from a running Linux OS with gdb, or from RedBoot (before the Linux kernel is executed)

Low Level Debugging with RedBoot

To provide simple, direct access to the TS-72XX hardware, RedBoot has an integrated debugger that can perform standard low-level debugger functions. It can be used to view or set values in memory using the “dump” and “memfill” commands. For example, the command

```
dump -b 0x80840020
```

shows that the first byte is filled with zeros, indicating that both the green and red led are turned off. The -b refers to a location in memory to display.

```
mfill -b 0x80840020 -p 0x03 -l 0x04
```

will result in the both the green and red LEDs being turned on. The -b, again, refers to a location in memory. The -p indicates the pattern to write into memory, and the -l refers to the length of the data being written out.

To verify that the write was successful,

```
dump -b 0x80840020
```

shows the first byte being having a value of 0x03 (bits 0 and 1 being set).

Linux - GDB

The GNU debugger is a sophisticated open source debugger. It can be used with Java, C, C++, or even Fortran. Please see the GNU Debugger Documentation homepage for more information. The following quickly walks through debugging a sample hello world application. The source code in this example is:

```
int squareit(int n)
{
int x;
x = n * n;
return x;
}
main ( )
{
int i;
for (i = 0; i < 4; i++)
  {
  printf("number %d\t", i);
  printf("number squared: %d\n", squareit(i));
  }
return 0;
};
```

Boot to the Compact Flash card, login as root, “apt-get install vim” (for a text editor), “apt-get install gdb” (ensure that the debugger is installed). Write the above source code into “helloworld.c”, then compile it:

```
gcc -g -o hello helloworld.c (compile the source with debugging symbols)
gdb ./hello
(gdb) list 1 (shows the source code)
(gdb) break 6 (sets a breakpoint at line number 6... this line would be return x; from
the squareit function)
(gdb) run (start the program)
(gdb) c (continue past the break point)
(gdb) set x=0 (this time, the return value is set to 0 instead of 1)
(gdb) c (continue and you'll see displayed on the screen “number 1 number squared
0”)
quit (exit gdb)
```

Linux – GDB Client/Server Debugger

TO-DO

Using Other Debuggers

Other Linux-capable debuggers will work with the TS-72XX. Please refer to your debugger's manual for both installation and use.

7.6 Java

There are many options to enable Java on your TS-72XX Linux box. Many TS customers are using the JamVM. JamVM is a compact Java Virtual Machine, suitable for embedded systems, that conforms to the JVM specification version 2.

In order to run Java via JamVM on your TS-ARM Linux system you will need to obtain and

compile the Jikes/JamVM/GNUClasspath combo:

- http://jamvm.sourceforge.net
- http://jikes.sourceforge.net
- http://www.gnu.org/software/classpath

Jikes is required when Java compilation is required on the embedded system, otherwise only JamVM is enough to have a Java-run-environment system.

Another way to install JamVM is through the Debian apt-get tool, making use of the JamVM compilled packages for Debian:

```
apt-get install jikes
apt-get install jamvm
apt-get install jikes-classpath
```

7.7 Using Eclipse IDE

TO-DO

7.8 Graphical User Interfaces

Qt/Embedded

It is possible to build advanced graphical user interface (GUI) applications on top of a single TS-7300 SBC or the set TS-7KV plus a TS-72XX SBC by using the QT/Embedded Free libraries and the Linux Framebufer driver. The development using Qt/Embedded is based on C/C++ Linux tools.

To learn how to program GUI application using the Qt/Embedded API, please refer to the QT/Embedded specific documentation.

Xfree

TO-DO

7.9 Sample Applicaitons

Technologic Systems provides some sample applications with source code.

DIO

The following is a C sample code that implements a button on DIO pin 1. When the button is pressed, the LED starts blinking:

```
#include<unistd.h>
#include<sys/types.h>
#include<sys/mman.h>
#include<stdio.h>
#include<fcntl.h>
#include<string.h>

int main(int argc, char **argv)
{
volatile unsigned int *PEDR, *PEDDR, *PBDR, *PBDDR, *GPIOBDB;
int i;
unsigned char state;
unsigned char *start;
```

```
int fd = open("/dev/mem", O_RDWR | O_SYNC);
start = mmap(0, getpagesize(), PROT_READ|PROT_WRITE, MAP_SHARED,
fd, 0x80840000);
PBDR = (unsigned int *)(start + 0x04); //port b
PBDDR = (unsigned int *)(start + 0x14); //port b direction
PEDR = (unsigned int *)(start + 0x20); //port e data
PEDDR = (unsigned int *)(start + 0x24); //port e direction
GPIOBDB = (unsigned int *)(start + 0xC4);  // debounce on port b
*PBDDR = 0xf0; //upper nibble output, lower nibble input
*PEDDR = 0xff; //all output (just 2 bits)
*GPIOBDB = 0x01; //enable debounce on bit 0
state = *PBDR; // read initial state
while (state & 0x01) { // wait until button goes low
  state = *PBDR; // remember bit 0 is pulled up with 4.7k ohm
  }

// blink 5 times, sleep 1 second so it's visible
for (i = 0; i < 5; i++) {
  *PEDR = 0xff;
  sleep(1);
  *PEDR = 0x00;
  sleep(1);
}
close(fd);
return 0;
}
```

A/D Converter

The following is sample Linux C code for initiating and printing an ADC conversion on channel 1.

```
#include<unistd.h>
#include<sys/types.h>
#include<sys/mman.h>
#include<stdio.h>
#include<fcntl.h>
#include<assert.h>

int main(int argc, char **argv)
{
volatile unsigned short * complete;
volatile unsigned char * lsb, * msb, * control;
int res;
int fd = open("/dev/mem", O_RDWR | O_SYNC);
assert(fd != -1);
lsb = control = (unsigned char *)mmap(0, getpagesize(),
PROT_READ|PROT_WRITE, MAP_SHARED, fd, 0x10c00000);
msb = lsb + 1;
complete = (unsigned short *)mmap(0, getpagesize(),
PROT_READ|PROT_WRITE, MAP_SHARED, fd, 0x10800000);
// Initiate conversion, channel #1, unipolar, 5V
*control = 0x41;
// Wait for completion
```

```
while ((*complete & 0x80) != 0);
// Print result on a scale from 0 to 2^12 - 1
res = *lsb;
res |= *msb << 8;
printf("result: %d\n", res);
close(fd);
return 0;
}
```

8 FURTHER REFERENCE

8.1 Using the Technologic Systems Website

The official Technologic Systems website contains useful information on hardware and software support. The latest products releases, including SBCs and PC/104 Peripheral Boards, and updated news and documentation are available.

- http://www.embeddedARM.com

Software package updates and further software documentation can be found at the Linux for ARM webpage. Most of the files mentioned in this manual are available there:

- http://www.embeddedarm.com/linux/ARM.htm

8.2 Recommended Readings

For those who are new to Linux, it will be beneficial to own and work within a full featured desktop Linux OS. Commonly, those new to Linux use distributions such as Redhat's Fedora, Mandrake Linux, or Suse. For those who are unwilling to install a Linux OS onto their system, Knoppix is recommended. Knoppix is a Live-CD that boots into Linux, and while offering a feature rich Linux OS, it won't touch the hard-drive unless told to. For those with are familiar with Linux but not familiar with programming in Linux, the following web-sites and books are excellent resources:

- http://www.ibm.com/developerworks

Full of tutorials and articles for Linux programming. For example, they have an excellent set of tutorials on programming threads in Linux

- http://www.kernelnewbies.org

Good starting point for those who are new to Linux kernel development.

- http://www.montavista.com

Monta Vista is a global leader in embedded Linux development. They offer their own Operating System, as well as their set of universal development tools (including IDEs). Monta Vista tools are platform independent, and work well for a Windows only environment.

- Linux Device Drivers, 2nd Edition By Alessandro Rubini & Jonathan Corbet

Highly recommended book for those who wish to learn the fundamentals to device driver development in Linux

8.3 Using the TS-7000 Mailing List

The TS-7000 Yahoo mailing list is a self help group for the Technologic Systems TS-7000 series of ARM based single board computers running the Linux for ARM OS. The group aims to help other users with hardware and software issues relating to the use of the TS-7000 series. Further supporting information on Technologic Systems hardware and software, extra sample codes and useful tips & tricks for ARM Linux can be found in the Yahoo Group for TS-7000 series. This is a great place to get information from experienced embedded engineers using TS-7000 boards in a wide variety of embedded applications.

This group has no official ties with Technologic Systems, although Technologic Systems engineers do contribute.

- http://groups.yahoo.com/group/TS-7000/

8.4 Using the Technologic Systems FTP Server

Technologic Systems makes a wide variety of supporting files for TS-72XX SBC's and Linux for ARM available for downloading through its FTP server.

- ftp://ftp.embeddedarm.com

8.5 Useful Data Sheets

- **Getting Started with TS-Linux**
  (http://www.embeddedarm.com/documentation/software/arm-tslinux-ts72xx.pdf)
- **Linux for ARM on TS-7000 User's Guide**
  (http://www.embeddedarm.com/documentation/software/arm-linux-ts72xx.pdf)
- **TS-7200 Data Sheet**
  (http://www.embeddedarm.com/documentation/ts-7200-datasheet.pdf)
- **TS-7250 Data Sheet**
  (http://www.embeddedarm.com/documentation/ts-7250-datasheet.pdf)
- **TS-7260 Data Sheet**
  (http://www.embeddedarm.com/documentation/ts-7260-datasheet.pdf)
- **TS-7300 Data Sheet**
  (http://www.embeddedarm.com/documentation/ts-7300-datasheet.pdf)
- **TS-7400 Data Sheet**
  (http://www.embeddedarm.com/documentation/ts-7400-datasheet.pdf)
- **EP9301 User's Guide**
  (http://www.embeddedarm.com/documentation/third-party/ts-7000_ep9301-ug.pdf)
- **EP9301 Data Sheet**
  (http://www.embeddedarm.com/documentation/third-party/ts-7000_ep9302-ds.pdf)
- **TS-7000 Yahoo Users' Group** (http://groups.yahoo.com/group/ts-7000/)

8.6 Where to go for additional help

Technologic Systems makes available several engineers to provide free technical support regarding the TS-72XX series hardware. To request support, call 480-837-5200 or email to support@embeddedARM.com. For general Linux questions not specific to the TS-72XX boards (such as, "How to configure the Apache web server"), you may be referred to the TS professional services group, which is well equipped to research just about any level of question from kernel development to web server configuration.

Since Linux is open-source, support may also be received from the individual development communities throughout the internet. Several projects have internet posted FAQs, HOWTOs, and community supported mailing lists providing users of their authored programs free help.

8.7 Hardware/Software Customization

Should your embedded project approach quantites between 500-1000 boards per year, it may be a good time to talk to TS engineer about specific hardware or software customizations that become available to you. There may be modifications we can do to reduce cost or add specific features to directly accomodate your specific requirements. Several options are available:

- TS can load your software directly onto the boards you order. Often times this can save you from creating your own production and software loading process. This type of service is available no matter the quantity.
- CPLD's/FPGA's can be reprogrammed to include specific customer-specified functionality. For instance, a magic register your software can verify before enabling certain features of your product or extra serial ports in the CPLD.

- Simple daughter-boards can be designed. Often, a low-cost 2-layer companion board can be created to add functionality to the board without actually changing the more expensive multi-layer TS-72XX circuit board.
- The TS-72XX board can be given new features and/or physical dimensions. This is the most expensive, but most powerful form of customization. Often times, TS may subsidize the development costs of a new board if it has general appeal to our broader customer base. Much of our COTS product line was a joint development between TS and another company. Allowing TS to generally market a design is a great way to acheive economies of scale (thereby reducing per-unit cost) and to ensure stock and commercial-off-the-shelf (COTS) status for your custom board.

8.8 About Technologic Systems Inc.

Technologic Systems has been developing embedded computing solutions for the OEM market over 20 years. Our commitment to excellent products, low prices and exceptional customer support has allowed our business to flourish in a very competitive marketplace. We offer a wide variety of off-the-shelf PC/104 SBCs and peripherals that meets most embedded project needs. Our engineers are also enthusiastic about new projects and take pride in new designs that are on the cutting edge of the embedded marketplace. Custom designs are a large component of our business and, with our exceptional embedded engineers, we can often engineer a custom solution at a price that competes with an in-house design.

Our Values

- **Innovative Hardware and Software Engineering**: Our engineers are on the cutting edge of embedded design. Our elegant and smart hardware/software solutions differentiate our products.
- **Real Ruggedness**: Technologic Systems builds rugged computers because we know how important reliability and longevity are to embedded applications. The internal electronics of our product are very simple to manufacture and operate with huge margins all around, and our thoughtful designs minimize the use of layers, vias, chips, solder points and exotic manufacturing technologies.
- **Fair Prices**: Our products and services are priced fairly and competively - we only charge you for what you really need, no unnecessary items like fancy cases or extra configuration services.
- **Open-Source Vision**: We provide complete Linux kernel source code, drivers and examples for our hardware, along with schematics and data sheets. There is no mystery when using our products for development.
- **Superior Support**: Our engineers answer technical support calls, and all our products have complete supporting documentation.
- **Extended Product Lifetime**: Technologic Systems has never discontinued a product in more than 20 years of business.

Additional Manufacturer (Technologic Systems) features:

- Engineers on free technical support.
- Each board undergoes many production tests and burn-in prior to shipment.
- Board customizations available with no minimum order.
- Factory loading of customer supplied software available.
- Readily available hardware and software professional services for hire.

APPENDIX A: DOCUMENT HISTORY

Date of Issue/Revision	Revision Number	Comments
Unknown	1.0	Old Linux for ARM Developer's Manuals
May 30, 2006	2.0	New and improved manual released
Dec 13, 2006	2.1	Some TO-DO sections completed, minor bugs fixed, new sections included
Dec 22, 2006	2.2	Debian info added, ARM feature matrix updated, marketing info added
Jul 05, 2007	2.3	Rename series to TS-72XX, minor corrections
Jul 08,2008	2.4	Fixed broken web links
Jul 09, 2009	2.5	Updated mailing address, minor corrections
Sep 30, 2011	2.6	Example code fixes
Jan 30, 2012	2.7	Example command fix

APPENDIX B: MEMORY AND REGISTER MAP

Address Region	Function
0xF000_0000 - 0xFFFF_FFFF	nCS0 (not used)
0xE000_0000 - 0xEFFF_FFFF	SDRAM (TS-7250)
0xD000_0000 - 0xDFFF_FFFF	SDRAM (not used)
0xC000_0000 - 0xCFFF_FFFF	SDRAM (not used)
0x8084_0000 - 0x8084_00C8	GPIO control registers
0x8000_0000 - 0x800F_FFFF	AHB mapped registers
0x71C0_0000 - 0x71FF_FFFF	CS7 PC/104 8/16 bit I/O (user selectable timing)
0x7180_0000 - 0x71BF_FFFF	CS7 PC/104 8/16 bit Memory (selectable timing)
0x7000_0000 - 0x70FF_FFFF	TS-9420 Flash
0x7000_0000 - 0x7FFF_FFFF	CS7 (bit bus cycles)
0x6000_0000 - 0x60FF_FFFF	on-board Flash (TS-7200)
0x6000_0000 - 0x6FFF_FFFF	CS6 (Flash)
0x3000_0000 - 0x3FFF_FFFF	CS3 (not used)
0x21C0_0000 - 0x21FF_FFFE	PC/104 16-bit I/O
0x2180_0000 - 0x21BF_FFFF	PC/104 16-bit Memory
0x2000_0000 - 0x2FFF_FFFF	CS2 (16-bit bus cycles)
0x11C0_0000 - 0x11FF_FFFF	PC/104 8-bit I/O
0x1180_0000 - 0x11BF_FFFF	PC/104 8-bit Memory
0x1000_0000 - 0x1FFF_FFFF	CS1 (8-bit bus cycles)
0x0001_0000 - 0x0000_FFFF	SDRAM region

Register Address	Function
0x8090_0020	Cirrus A/D lock register
0x8090_0018	Cirrus A/D channel select register
0x8090_0008	Cirrus A/D result register (RO)
0x808D_0000 - 0x808D_FFFF	UART2 control registers
0x808C_0000 - 0x808C_FFFF	UART1 control registers
0x808A_0000 - 0x808A_FFFF	SPI control registers
0x8084_0044	LCD_EN, LCD_RS, LCD_WR direction reg.(bits3-5)
0x8084_0040	LCD_EN, LCD_RS, LCD_WR data reg.(bits3-5)
0x8084_0034	DIO_8 direction register (bit 1)
0x8084_0030	DIO_8 data register (bit 1)
0x8084_0020	On-board LEDs register (bits 0, 1)
0x8084_0018	Port C direction register (TS-7300) LCD_7 direction register (bit 7)
0x8084_0014	DIO_0 thru DIO_7 direction register (R/W)
0x8084_0010	LCD_0 thru LCD_7 direction register (R/W)
0x8084_0008	Port C data register (TS-7250) 1000 mA driver output on DIO (bit 0) (TS-7300) LCD_7 data register (bit 7)
0x8084_0004	DIO_0 thru DIO_7 data register (R/W)
0x8084_0000	LCD_0 thru LCD_7 data register (R/W)
0x8081_0000 - 0x8081_FFFF	Timer Control registers
0x8080_0000 - 0x8FFF_FFFF	APB mapped registers
0x800B_0000 - 0x800B_FFFF	VIC 0 registers
0x8006_0000 - 0x8006_FFFF	SDRAM control registers
0x8002_0000 - 0x8002_FFFF	USB registers
0x8001_0000 - 0x8001_FFFF	Ethernet MAC registers
0x7220_0000 - 0x729F_FFFF	(TS-7300) FPGA SDRAM region
0x7210_0000 - 0x7200_FFFF	(TS-7300) FPGA MAC core 32-bit registers

Register Address	Function
0x7200_0044 - 0x7200_0047	(TS-7300) FPGA DIO2 TS-XDIO #2 registers
0x7200_0040 - 0x7200_0043	(TS-7300) FPGA DIO2 TS-XDIO #1 registers
0x7200_0030 - 0x7200_003A	(TS-7300) FPGA Video core 16-bit registers
0x7200_0020 - 0x7200_0027	(TS-7300) FPGA SD Card core 16-bit registers
0x7200_0000 - 0x7200_001F	(TS-7300) FPGA COM3-4-5-6-7-8 16-bit registers
0x6000_0000	(TS-7250) NAND Flash data register
0x6040_0000	(TS-7250) NAND Flash control register (bits 0-2)
0x6080_0000	(TS-7250) NAND Flash Busy status (bit 5)
0x23C0_0000	WDT Feed register (bits 0-2)
0x2380_0000	WDT Control register (bits 0-2)
0x2340_0000	PLD version (bits 0-2)
0x2300_0000	COM2 RS-485 control register
0x22C0_0000	COM2 RS-485 control register (bits 0-2)
0x2280_0000	JP6 (bit 0)
0x2280_0000	Booting from TS-9420 (bit 1)
0x2280_0000	TS-9420 present (bit 2)
0x2240_0000	MAX197 A/D option present (bit 0)
0x2240_0000	COM2 RS-485 option present (bit 1)
0x2200_0000	Model Number (bits 0-2)
0x21E0_0000 - 0x21E0_03FE	PC/104 16-bit I/O (legacy support)
0x2100_0000	CF IDE 16-bit register
0x13C0_0000 - 0x13C0_0001	(TS-7300) FPGA loader registers
0x1340_0000 - 0x1340_0002	(TS-7260) Additional COM5 registers
0x1300_0000 - 0x1300_0002	(TS-7260) Additional COM4 registers
0x12C0_0000 - 0x12C0_0003	(TS-7260) DIO2 Header TS-XDIO registers
0x12C0_0000 - 0x12C0_0001	(TS-7260) DIO2 Header Basic DIO registers
0x1240_0000 - 0x1240_0002	(TS-7260) COM3 serial port registers
0x1200_0000	(TS-7260) Power Management register
0x11E0_0000 - 0x11E0_03FF	PC/104 8-bit I/O (legacy support)
0x11A0_0000 - 0x11AF_FFFF	PC/104 8-bit Memory (legacy support)
0x1170_0000	RTC R/W data register
0x1100_0001 - 0x1100_0007	CF IDE 8-bit registers
0x10F0_0000 - 0x10F0_0001	MAX197 A/D registers
0x1080_0000	MAX197 A/D busy bit (bit 7) (RO)
0x1080_0000	JP2-JP5 (bits 0,1,3,4) (RO)
0x1080_0000	COM1 DCD (bit 6) (RO)
0x1080_0000	Write Only index register
0x1040_0006 - 0x1040_0007	CF AUX IDE 8-bit registers

APPENDIX C: DOWNLOADS - SCHEMATICS AND MECHANICAL DRAWING

- **TS-7200 schematic**
  (http://www.embeddedarm.com/documentation/ts-7200-schematic.pdf)
- **TS-7200 mechanical drawing**
  (http://www.embeddedarm.com/documentation/ts-7200-mechanical.pdf)
- **TS-7200's download section** (http://www.embeddedarm.com/epc/ts7200-spec-d.html)

- **TS-7250 schematic**
  (http://www.embeddedarm.com/documentation/ts-7250-schematic.pdf)
- **TS-7250 mechanical drawing** (http://www.embeddedarm.com/documentation/ts-7250-mechanical.pdf)
- **TS-7250's download section** (http://www.embeddedarm.com/epc/ts7250-spec-d.html)

- **TS-7260 schematic**
  (http://www.embeddedarm.com/documentation/ts-7260-schematic.pdf)
- **TS-7260 mechanical drawing**
  (http://www.embeddedarm.com/documentation/ts-7260-mechanical.pdf)
- **TS-7260's download section** (http://www.embeddedarm.com/epc/ts7260-spec-d.htm)

APPENDIX D: TS-ARM SBC FEATURE MATRIX

Feature/Product	TS-7200	TS-7250	TS-7260	TS-7300	TS-7400
CPU	200 Mhz AMR920T	200 Mhz AMR920T	200 Mhz AMR920T	200 Mhz AMR920T	200 Mhz AMR920T
Category	SBC	SBC	SBC	SBC	SoM
PC/104 Connector	Yes	Yes	Yes	Yes	No
On-board FPGA	No	No	No	Yes	No
RAM	32 MB	32 MB	32 MB	32 MB	32 MB
Optional RAM	64 MB	64 MB 128 MB	64 MB 128 MB	64 MB 128 MB	64 MB 128 MB
On-board Flash	8 MB	32 MB	32 MB	No	32 MB
Optional on-board Flash	16 MB	64 MB 128 MB 256 MB	64 MB 128 MB 256 MB	Possible Contact us	64 MB 128 MB 256 MB
Standard A/D	Yes - 2 ch	Yes - 5 ch	Yes - 2 ch	Yes - 1 ch	Yes - 4ch
Optional A/D	8 ch	8 ch	No	No	No
Ethernet	10/100 - 1	10/100 - 1	10/100 - 1	10/100 - 2	10/100 - 1
USB1.1 ports	Yes - 2	Yes - 2	Yes - 2	Yes - 2	Yes - 2
VGA Video Out	No	No	No	Yes	No
IDE Compact Flash	Yes	No	No	No	No
SD Card Interface	No	No	Yes - 1 on RevB	Yes - 2	Yes - 1
Digital I/O	20	20	30	55	20
TS-XDIO	No	No	Yes - 1	Yes - 2	No
RS-485	Opt. full/half	Opt. full/half	Opt. full/half	Opt. full/half	Opt. full/half
Standard COM Ports	2	2	3	10	3 TTL
Optional COM Ports	No	No	2	Yes	No
RS232 Console	Yes	Yes	Yes	Yes	No
RTC	Opt.	Opt.	Opt	Opt.	Opt.
LCD Interface	Yes	Yes	Yes	Yes	No
Keypad Interface	Yes	Yes	Yes	Yes	No
TS-Linux	Yes	Yes	Yes	Avail.	Yes
Debian Linux	Avail. CF	Avail. USB	Avail. SD	Yes	Avail. SD
Linux Fast Bootup	No	No	No	1.69 secs	1.1 secs
Linux Bootloader	Avail.	Avail.	Avail.	Yes	Yes
Real-Time RTAI	Yes	Yes	Yes	Yes	Yes
Java	Yes	Yes	Yes	Yes	Yes
USB WiFi Support	Opt.	Opt.	Opt.	Opt.	Yes
USB Flash Support	Opt.	Opt.	Opt.	Opt.	Yes
Extended Temperature	Yes	Yes	Yes	Yes	Yes
Switching-Mode Power Supply	No	No	Yes	No	Avail.
RoHS Compliance	Opt.	Opt.	Opt.	Yes	Yes
Quantity 1 Pricing	$149.00	$149.00	$179.00	$219.00	$129.00
Quantity 100 Pricing	$119.00	$119.00	$149.00	$189.00	$99.00

APPENDIX E: CONTACT TECHNOLOGIC SYSTEMS

16525 East Laser Drive
Fountain Hills, AZ 85268
TEL 1.480.837.5200
FAX 1.480.837.5300

www.embeddedARM.com
support@embeddedARM.com

Call us Monday-Friday, **from 9 am to 5 pm**, Arizona-USA time;
or email us at any time.

Our engineers answer tech support calls and are more than happy to talk to you about your needs and help you find the best solution for your project.