

# Adventures with an ARM Single Board Computer - TS-7200

Jim Jackson <jj@franjem.org.uk>

November 14, 2006

version 6

---

## Contents

<b>1 Introduction</b>	<b>3</b>
<b>2 TS-7200 Features</b>	<b>4</b>
<b>3 What it Doesn't Have.....</b>	<b>5</b>
<b>4 How robust is flash memory?</b>	<b>5</b>
<b>5 TS7xx0 Support</b>	<b>6</b>
<b>6 TS7200 Software</b>	<b>7</b>
<b>7 Setting the Board up</b>	<b>8</b>
<b>8 Linux on the TS-7200</b>	<b>9</b>
8.1 Software in the Base System.....	9
8.2 Connectivity options.....	9
<b>9 Security on the basic Flash Linux image</b>	<b>10</b>
<b>10 Some small Deficiencies in built-in Linux setup</b>	<b>11</b>
<b>11 Connecting up COM2 - the second RS232 port</b>	<b>12</b>
<b>12 RedBoot - the boot loader</b>	<b>13</b>
<b>13 Cross Compiling applications</b>	<b>14</b>
<b>14 Compiling Busybox on i386 for ARM target.....</b>	<b>16</b>
14.1 Install Source .....	16

<i>CONTENTS</i>	2
14.2 Compile and Install . . . . .	16
<b>15 Inserting the new BusyBox in the flash image</b>	<b>17</b>
<b>16 Setting up and booting using an NFSROOT</b>	<b>18</b>
<b>17 Setting up and booting from a Compact Flash Disk</b>	<b>19</b>

---

## 1 Introduction

Always wanted a computer to do control and measurement, but not had time to get to grips with PIC or the serious embedded stuff. Most easy to use SBC's are pretty pricey.

In Summer 2004 I saw an advert in Linux Journal for an ARM SBC that seemed somewhat more affordable, yet had a range of options and facilities that looked like it would be easy to get up and running for doing things.

In September 2004 I took the plunge and bought the TS-7200 with ADC option from:

Technologic Systems,  
16610 East Laser Drive Suite 10,  
Fountain Hills, AZ 85268

Web: <http://www.embeddedx86.com> Email: [info@embeddedx86.com](mailto:info@embeddedx86.com)

The cost was \$184 plus import duty etc and postage & packing. Import duty ended up at about 30% and P&P was a pretty hefty \$48. All in all more than I originally planned on spending.

Since then Technologic have produced the TS-7250 board, with no CFDisk interface, and the TS-7260, a low power version of the board. Technologic provide a comparison of the features of the various boards at <http://www.embeddedarm.com/epc/prod-matrix-arm.php>

The latest version of this document can be found at <http://www.comp.leeds.ac.uk/jj/linux/arm-sbc.html>

Picture showing the RS232, USB and Ethernet connectors [http://www.comp.leeds.ac.uk/jj/linux/images/TS7200\\_ARM\\_SBC\\_05.jpg](http://www.comp.leeds.ac.uk/jj/linux/images/TS7200_ARM_SBC_05.jpg)

Picture showing the Jumpers, ADC, COM2, "LCD" and DIO connectors. Pin 1 is marked with a white dot. [http://www.comp.leeds.ac.uk/jj/linux/images/TS7200\\_ARM\\_SBC\\_05.jpg](http://www.comp.leeds.ac.uk/jj/linux/images/TS7200_ARM_SBC_05.jpg)

---

## 2 TS-7200 Features

- 166 MHz ARM processor (Cirrus EP9301) - versions supplied from late Nov 04 have the 200M EP9302)
- 32 MB of High Speed SDRAM
- 2 PC/AT standard serial ports with 16 Byte FIFOs
- 8 MB Flash disk used for RedBoot boot-loader, Linux kernel etc
- IDE Compact Flash interface
- 10/100 Ethernet interface - autosense, LED indicators
- 20 DIO lines
- PC/104 8/16 bit bus
- SPI bus header
- 2 USB ports (1.1)
- Watchdog timer unit
- Dimensions are 3.8 “ x 4.5 ” (PC/104 mounting holes)
- Power requirements are 5V DC @ 375mA (max. current @ full speed)
- Operating Temperature Range: Fanless 0°C to 70° C
- Option: Battery Backed Real Time Clock, RS-485 support on COM1, 8 ch. 12-bit A/D converter

for more details of Technologies ARM Products, see [http://www.embeddedarm.com/epc/prod\\_SBC.htm](http://www.embeddedarm.com/epc/prod_SBC.htm)

EP9301: <http://www.cirrus.com/en/products/pro/detail/P1053.html>

EP9302: <http://www.cirrus.com/en/products/pro/detail/P1066.html>

---

### **3 What it Doesn't Have.....**

- Any on board Video
- IDE interface
- keyboard interface
- memory expansion - SDRAM fixed at 32M, flash fixed at 8M - though I believe there are now boards with more memory
- Hardware FPU

### **4 How robust is flash memory?**

Technologic say there is guaranteed 100,000 rewrites, which at 10 full flash writes per day, means it will last aprox 27 years :-)

---

## 5 TS7xx0 Support

First check the Technologies Web site. The user guides and other documentation are under revision and it always helps to have the latest versions.

There is a Yahoo group for discussing TS7xxx stuff. It's at <http://groups.yahoo.com/group/ts-7000/> and there are several helpful people on the list. Some of the Technologies engineers contribute too.

Yan Seiner has setup a Wiki as a repository for TS7xxx stuff at <http://www.seiner.com/ts7000/index.php/Home>

Finally, there is Technologies themselves!

---

## 6 TS7200 Software

Here is some of the software I've written for the TS-7200. All the software is provided "as is", bugs and all :-). It is licensed under the GPL (GNU General Public License), <http://www.gnu.org/copyleft/gpl.html>, and is to be considered Beta release software.

I would appreciate feedback from you if you download and use the software, especially if you think you have found a bug.

**adc\_logger** command line program for getting regular repetitive readings from the TS7200 ADC. (Version 2.1) [http://www.comp.leeds.ac.uk/jj/linux/adc\\_logger.tgz](http://www.comp.leeds.ac.uk/jj/linux/adc_logger.tgz)

**adio** command line program to access the TS7200 ARM SBC DIO lines and the ADC channels, allows scripting. <http://www.comp.leeds.ac.uk/jj/linux/adio.tgz>

**cctl** command line program to turn on/off parts of the TS7200/EP930x and to alter things like the CPU, Memory and bus clock frequencies <http://www.comp.leeds.ac.uk/jj/linux/cctl.tgz>

**lcdd** a user space driver for HD44780 based char. LCDs Daemon listens to a named pipe for data to write to LCD <http://www.comp.leeds.ac.uk/jj/linux/lcdd.tgz>

For an overview of doing memory IO accesses from user space under Linux, read <http://www.simtec.co.uk/appnot>

---

## 7 Setting the Board up

You will need a source of regulated 5 volts capable of supplying upto 1amp. I had an old one lying around that I soak tested overnight before connecting up.

**It is important to get the polarity right - I understand there is no onboard protection and connecting the 5v DC the wrong way round can fry your board.**

The Linux console is redirected to the first serial port on the board ttyAM0 (aka COM1) so connect a 9-pin NULL modem cable between the board and a serial port on your development machine. You will need a terminal emulator on your development machine to act as a “Terminal” on the connected serial line. I use ckermit, minicom will also work.

The serial line should be configured at 115,200Baud, 8 data bits, 1 stop bit. I set no flow control and it worked fine.

Measured current consumption in operation was:

No Ethernet Connection	280mA @5v	1.4W
With Ethernet Connected	360mA @5v	1.8W

---

## 8 Linux on the TS-7200

See Technologies “Linux for TS-7000 Series ARM SBC’s” page at <http://www.embeddedarm.com/linux/ARM.htm>  
The TS-7200 boots using RedBoot <http://sources.redhat.com/redboot/> and by default boots an ARM Linux Kernel (On my Sept. 04 board Linux version 2.4.26-vrs1-cirrus-1-2-1-ts3) from the onboard 8M Flash memory, with a root file system of a small base linux system in the rest of the 8M Flash memory.

### 8.1 Software in the Base System.....

- Busybox <http://www.busybox.net/>.
- Apache
- dropbear <http://matt.ucc.asn.au/dropbear/dropbear.html> a small footprint sshd replacement

### 8.2 Connectivity options.....

- telnet
- ftp
- apache
- NFS
- ssh

For a transcript of the output on boot and some general live system info, see <http://www.comp.leeds.ac.uk/jj/li>

---

## 9 Security on the basic Flash Linux image

- set a root password! The default installed image allows root login via telnet with a null password
- enable dropbear, the ssh daemon as a replacement for telnet, generate the host key.....

```
$ dropbearkey -t rsa -f /etc/dropbear/dropbear_rsa_host_key
```

- disable telnetd

```
$ cd /etc/rc.d/rc3.d
```

```
$ mv S30telnetd NOS30telnetd
```

---

## 10 Some small Deficiencies in built-in Linux setup

- the provided version of dropbear doesn't seem to support scp nor can it be used to ssh off the board
  - busybox not compiled with some needed functionality, e.g. rdate, nc (netcat), top
  - I've found problems cross compiling programs that use Floating point numbers.
-

## 11 Connecting up COM2 - the second RS232 port

The user manual omits the pinout for the 10 pin COM2 header, giving only the pinout of the DB9 connector when used with the manufacturer's own header connector. The documentation says that only RXD/TXD are supported on COM2. Here is the pin out of the RS232 lines from the 10 pin COM2 header, and the pins to connect to for a male DB9 or DB25 standard DTE connection which works for me.

COM2 Header		DB9	DB25
Pin 3	RS232 RXD	pin 2	pin 3
Pin 5	RS232 TXD	pin 3	pin 2
Pin 9	ground	pin 5	pin 7

Assuming that the Technologic header pin to DB9 connector is designed to be simple to make using IDC components and 9 way ribbon cable, I think it a fair bet that the full 10 pin COM2 header connections are...

COM2 Header		DB9
Pin 1	TX+(/RX+)	pin 1
Pin 2	TX-(/RX-)	pin 6
Pin 3	RXD	pin 2
Pin 4	RTS	pin 7
Pin 5	TXD	pin 3
Pin 6	CTS	pin 8
Pin 7	RX+	pin 4
Pin 8	RX-	pin 9
Pin 9	GND	pin 5

**WARNING: I have NOT tried this, and the information is only offered to assist those willing to experiment at their own risk!**

---

## 12 RedBoot - the boot loader

On booting the default image you have 1 second to press Ctrl-C on the console connection to interrupt the boot process and get a prompt to RedBoot...

```
.....
Platform: TS-7200 Board (ARM920T) Rev A
Copyright (C) 2000, 2001, 2002, Red Hat, Inc.
Copyright (C) 2004, Technologic Systems

RAM: 0x00000000-0x02000000, 0x00053ef0-0x01fdd000 available
FLASH: 0x60000000 - 0x60800000, 64 blocks of 0x00020000 bytes each.
== Executing boot script in 2.000 seconds - enter ^C to abort
^C
RedBoot>
```

From the RedBoot prompt you can print the boot configuration details...

```
RedBoot> fconfig -l
.....
```

or change the boot configuration details...

```
RedBoot> fconfig
Run script at boot: true
...
```

The fconfig program offers up the existing values for editing/changing, and at the end you are asked if you want to commit the changes to non-volatile memory. You can change the boot script, the amount of time to wait before executing the boot script (I set this to 20 (2secs) to give me time to get in), and network details to be used if RedBoot is asked to download via tftp.

If you want to boot everytime to a root image on CFDisk or NFSRoot then you will want to change the boot script. See below for details of setting up a CFDisk for booting and booting from an NFSRoot.

To do a default boot from RedBoot, use the **reset** command to reboot.

---

## 13 Cross Compiling applications

On their Linux page at <http://embeddedx86.com/linux/ARM.htm> Technologic Systems provide precompiled cross compilation tool chains, for Linux and for Windows users using the Cygwin tools. I strongly recommend using a Linux development environment. Being familiar with the Linux/Unix environment pays big dividends in getting stuff working on the TS-7200. Old Linux/Unix hands will have little trouble in getting stuff to go on the board.

On my desktop I created a new account **arm** so that I could customise setup for cross compiling ARM programs and not interfere with my default development setup.

The Linux Cross compilation tool chain installs into **/usr/local/opt/crostool/arm-linux/gcc-3.3.2-glibc-2.3.2/bin** with a set of gcc compile tools for generating and manipulating ARM binaries, but with names prefixed by “**arm-linux-**”

In my **arm** home directory I set up an **xbin** directory and created symlinks to the tools with both their original names and with the shorter normal tool names (e.g. gcc instead of arm-linux-gcc) to the programs under **/usr/local/opt/crostool/arm-linux...**

```
$ mkdir ~/xdir
$ cd ~/xbin
$ for f in /usr/local/opt/crostool/arm-linux/gcc-3.3.2-glibc-2.3.2/bin/*
> do
> ln -s $f .
> ln -s $f `echo $f | sed -e 's/^.*\arm-linux-//'`
> done
$
```

I then added commands **armon** and **armoff** to turn on/off the cross-compiling environment.....

```
armon () {
  if (echo $PATH | grep "$HOME/xbin" > /dev/null); then
    :
  else
    PATH=$HOME/xbin:$PATH
    export PATH
  fi
  echo "Environment set for cross compiling for ARM environment"
}

armoff () {
  PATH=$DEFPATH
  export PATH
  echo "Environment NOT set for cross compiling for ARM environment"
}

PATH=$HOME/bin:$PATH
DEFPATH=$PATH
export PATH
export DEFPATH
```

```
if [ ! $?MANPATH ] ; then
  MANPATH='manpath -q'
  export MANPATH
fi
MANPATH=$HOME/man:/usr/local/opt/crosstool/arm-linux/gcc-3.3.2-glibc-2.3.2/man:$MANPATH
export MANPATH
```

... to my **.bashrc** file - my **.bash\_login** file sources this too...

```
#!/bin/bash
#

. $HOME/.bashrc
armon
```

---

## 14 Compiling Busybox on i386 for ARM target.....

BusyBox Home Page <http://busybox.net/>

### 14.1 Install Source

I used <http://busybox.net/downloads/busybox-1.00-rc3.tar.gz> and downloaded it into ~arm/Src

```
$ cd ~/Src
$ tar vxzf busybox-1.00-rc3.tar.gz
.....
$ cd busybox-1.00-rc3
```

### 14.2 Compile and Install

The compile process needs to make some utilities to run on the host machine before it goes away and compiles busybox for the target machine (the ARM SBC). So we must disable the ARM Cross compile environment for part of the process and turn it back on again to finish off.....

```
$ armoﬀ
$ make config (or make menuconfig)
...
  answer questions etc.
...
$ make dep
$ armon
$ make
$ make PREFIX=/home/arm/export install
```

(substitute where you want the compiled version of busybox to be installed)

I added quite a few extra BusyBox components to the one provided on the TS7200. e.g. rdate, nc (netcat), top, netstat, telnetd (there are separate binaries for these last 2, not sure why), who ....

---

## 15 Inserting the new BusyBox in the flash image

**Beware** this is only an outline of the process needed. Be carefull. BusyBox provides nearly all the functionality for booting, if it doesn't work the flash image might not boot - though a CFdisk or NFSROOT boot would rescue the situation.

- **tar** up the flash image with the exception of /dev and /proc (because they are virtual file systems)

```
$ tar cf /tmp/flash.tar bin etc lib mnt root sbin usr var www
```

- move the tar ball to the host machine
- untar the tar ball
- rm bin/busybox
- rm all the symlinks from bin , sbin , usr/bin , usr/sbin . Use find to make a list of them all - only delete the busybox links.

```
$ find . -type l -exec ls -l {} \;
```

- install busybox into the area (see above)
  - tar up again to a new tar ball.
  - Test this first by either copying to a compact flash card or an NFSROOT and booting from that to check it is ok (see elsewhere in this document)
  - When happy and while booted from the CFdisk or NFSRoot, mount and delete the contents of the flash, and untar the new image into the flash etc.
-

## **16 Setting up and booting using an NFSROOT**

**TO BE COMPLETED**

---

## 17 Setting up and booting from a Compact Flash Disk

TO BE COMPLETED

---

HomePage <http://www.comp.leeds.ac.uk/jj>