From Technologic Systems Manuals

TS-7100

Note: This manual is incomplete at this time and is subject to change without warning while the TS-7100 is in Engineering Sampling phase.

Contents

TS-7100-Z

Product Page (http://www.embeddedarm.com/products/board-detail.php?product=TS-7100)

Documentation

Schematic (https://www.embeddedarm.com/documentation/ts-7100-schematic.pdf)

Mechanical Drawing (https://www.embeddedarm.com/documentation/ts-7100-mechanical.pdf)

FTP Path (http://ftp.embeddedarm.com/ftp/ts-arm-sbc/ts-7100-linux/)

Processor

NXP i.MX6UL

528MHz or 696MHz

i.MX6UL Product Page (http://www.nxp.com/products/microcontrollers-and-processors/arm-processors/i.mx-applications-processors/i.mx-6-processors/i.mx6qp/i.mx-6ultralite-processor-low-power-secure-arm-cortex-a7-core:i.MX6UL?)

CPU Documentation (http://www.nxp.com/products/microcontrollers-and-processors/arm-processors/i.mx-applications-processors/i.mx-6-processors/i.mx6qp/i.mx-6ultralite-processor-low-power-secure-arm-cortex-a7-core:i.MX6UL?fpsp=1&tab=Documentation_Tab#)

1 About This Manual

The TS-7100 series of products all incorporate two PCBs connected by a high density connector. One of these PCBs contains the CPU and some basic peripherals. The second PCB is considered to be the "I/O board" which breaks out GPIO, networking, as well as other peripherals that can vary from product to product. Every product in the TS-7100 series uses the same CPU board and offers the same base features. All of the TS-7100 series products will have a full product number, for example "TS-7100-Z" is the first product in this line. This manual and all TS-7100 series product manuals will both use "TS-7100" as well as the full product number.

When the "TS-7100" name is used, this indicates that this feature is common to all products in the series. For example, soldered down eMMC flash is soldered to the CPU board. When the full product number is used, it is used to indicate that the feature or peripheral being discussed is available on that product, but not necessarily every product in the TS-7100 series.

As always, if there are any questions or concerns, please reach out to Technologic Systems' support team (https://www.embeddedarm.com/support/) .

2 Overview

The TS-7100 is a small embedded CPU module with an NXP i.MX6UL 696 MHz CPU with 512 MB DDR3 RAM. The CPU module provides soldered down eMMC flash, non-volatile FRAM for 2 KiB of storage, dual Ethernet PHYs, support for a 16-bit 240x360 LCD display with resistive touch, and many more.

Configuration flexibility can be achieved in the TS-7100's mated I/O board. The I/O board provides additional peripherals, industrial rated I/O and connectors, Ethernet, CAN, UARTs, WiFi and Bluetooth, etc. to expand functionality and interact with the real world.

The TS-7100-Z is the TS-7100 mated with the TS-7100-Z I/O board. The TS-7100-Z offers a DIN rail mountable enclosure with LCD and touch screen display, as well as wireless connectivity and industrial I/O. Peripherals available on the TS-7100-Z board include: soldered down WiFi with built in Bluetooth, our TS-SILO supercapacitor technology for safe shutdown upon power loss, dual 10/100 Ethernet ports, two relays, dual USB host ports, 30 VDC high-current low-side switch outputs or digital inputs, dedicated 30 VDC digital inputs, dedicated high-side switch output, 0-12 V or 4-20 mA ADC inputs, with all I/O protected by a hardware over-voltage or over-current breaker system, and a 8 V to 48 V DC input range.

3 Getting Started

A Linux PC is recommended for development, and will be assumed for this documentation. For users in Windows or OSX we recommend virtualizing a Linux PC. Most of our platforms run Debian and if there is no personal distribution preference this is what we recommend for ease of use.

- Debian.org (https://www.debian.org/)

Virtualization

- Virtualbox (Windows or OSX hosts) (https://www.virtualbox.org/wiki/Downloads)
- VMware Player (https://www.vmware.com/products/player)
- Parallels (OSX) (http://www.parallels.com/)

Suggested Linux Distributions

- Debian (https://www.debian.org/distrib/)
- Ubuntu (http://www.ubuntu.com/desktop)

It may be possible to develop using a Windows or OSX system, but this is not supported. Development will include accessing drives formatted for Linux and often Linux based tools.

3.1 Connect USB Console

The TS-7100 includes a USB Micro B device port; this uses a 8051 based microcontroller to create a debug/console serial interface on a host PC. The serial console is provided through this port at 115200 baud, 8n1, with no flow control. The USB serial device is a CP210x Virtual COM Port. Most operating systems have built-in support for this device. If not however, drivers are available for the device here (https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers) .

Console from Linux

There are many serial terminal applications for Linux, three common used applications are 'picocom', 'screen', and 'minicom'. These examples demonstrate all three applications and assume that the serial device is "/dev/ttyUSB0" which is common for USB adapters. Be sure to replace the serial device string with that of the device on your workstation.

'picocom' is a very small and simple client.

```
picocom -b 115200 /dev/ttyUSB0
```

'screen' is a terminal multiplexer which happens to have serial support.

```
screen /dev/ttyUSB0 115200
```

Or a very commonly used client is 'minicom' which is quite powerful but requires some setup:

```
minicom -s
```

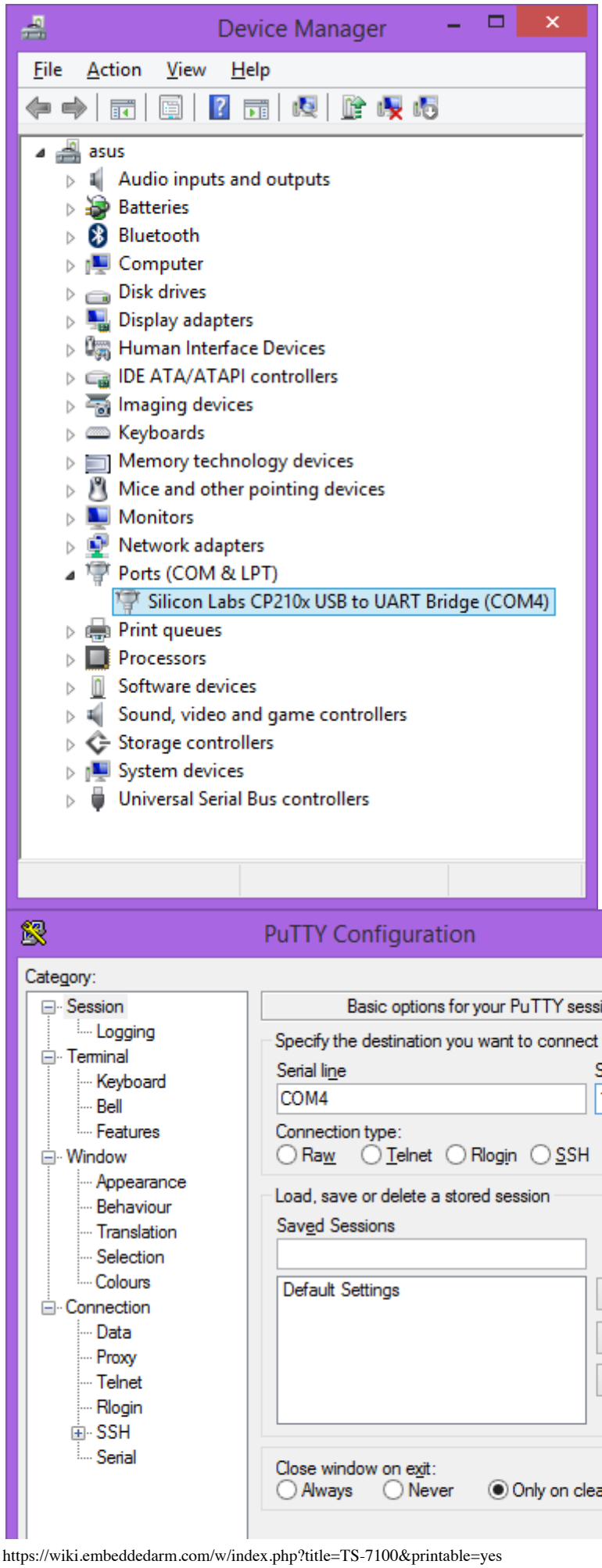
- Navigate to 'serial port setup'
- Type "a" and change location of serial device to '/dev/ttyUSB0' then hit "enter"
- If needed, modify the settings to match this and hit "esc" when done:

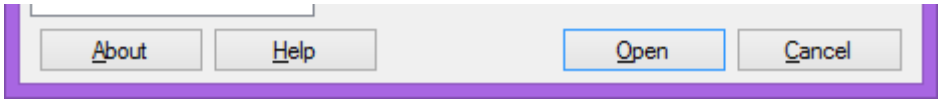
```
E - Bps/Par/Bits          : 115200 8N1
F - Hardware Flow Control : No
G - Software Flow Control : No
```

- Navigate to 'Save setup as dfl', hit "enter", and then "esc"

Console from Windows

Putty is a small simple client available for download here (http://www.chiark.greenend.org.uk/~sgtatham/putty/) . Open up Device Manager to determine your console port. See the putty configuration image for more details.


Device Manager
File Action View Help
asus
Audio inputs and outputs
Batteries
Bluetooth
Computer
Disk drives
Display adapters
Human Interface Devices
IDE ATA/ATAPI controllers
Imaging devices
Keyboards
Memory technology devices
Mice and other pointing devices
Monitors
Network adapters
Ports (COM & LPT)
Silicon Labs CP210x USB to UART Bridge (COM4)
Print queues
Processors
Software devices
Sound, video and game controllers
Storage controllers
System devices
Universal Serial Bus controllers
PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Serial
Basic options for your PuTTY session
Specify the destination you want to connect to
Serial line
Speed
COM4
115200
Connection type:
Raw
Telnet
Rlogin
SSH
Serial
Load, save or delete a stored session
Saved Sessions
Default Settings
Load
Save
Delete
Close window on exit:
Always
Never
Only on clean exit



3.2 Powering Up

WARNING: Be sure to take appropriate Electrostatic Discharge (ESD) precautions. Disconnect the power source before moving, cabling, or performing any set up procedures. Inappropriate handling may cause damage to the board.

Power input to the TS-7100 is supplied via the power input connector, refer to that section for information on voltage ranges for this device.

Once power is applied to the whole device, there will be output on the debug console port. The following section of the manual provides information on getting the serial console connected.

```
U-Boot 2016.03-00408-gd450758c91 (Oct 10 2019 - 11:59:08 -0700)

CPU:   Freescale i.MX6UL rev1.2 at 396 MHz
Reset cause: POR
I2C:   ready
DRAM:  512 MiB
MMC:   FSL_SDHC: 0
Net:   FEC0 [PRIME]
Warning: FEC0 (eth0) using random MAC address - 72:12:64:ca:3e:4a

Press Ctrl+C to abort autoboot in 1 second(s)
starting USB...
USB0:   Port not available.
USB1:   USB EHCI 1.00
scanning bus 1 for devices... 1 USB Device(s) found
       scanning usb for storage devices... 0 Storage Device(s) found
No storage devices, perhaps not 'usb start'ed..?
Booting from the eMMC ...
** File not found /boot/boot.ub **
31526 bytes read in 103 ms (298.8 KiB/s)
5253608 bytes read in 354 ms (14.2 MiB/s)
NO CHRG jumper is set, not waiting

Kernel image @ 0x80800000 [ 0x000000 - 0x500220 ]
## Flattened Device Tree blob at 83000000
   Booting using the fdt blob at 0x83000000
   Using Device Tree in place at 83000000, end 8300a909

Starting kernel ...
```

Note: The "*** Warning - bad CRC, using default environment" message can be safely ignored when the unit is first booted. This means that no environment variables have been saved to disk, and U-Boot is falling back to the default. If "env save" is run, this will save the environment to disk, and this message will go away unless there is a further issue.

The default U-Boot boot process will check for USB updates before attempting to boot from on-board eMMC. Details about the bootup process, features, and other U-Boot information can be found in the U-Boot sections.

3.3 First Linux Boot

When booting with the default settings, a shipped board will boot to the eMMC. The eMMC by default is pre-programmed with our default Debian 9 Stretch image. After Debian boots it will ask the user to log in with a username and password. This uses "root" as the username with no password. This can be changed after logging in with the command 'passwd' to set an account password. Note that this login will only work over the serial console. Debian SSH defaults to not only disallowing password-less logins, but root logins altogether are denied.

From the Linux prompt, the hardware can be tested out or application development can be begin.

4 U-Boot

TS-7100 U-Boot Sections

5 Debian Stretch(9)

5.1 Getting Started

By default, the TS-7100 ships with a Debian image installed and ready to boot. It is not necessary to download and install the latest image below for operating the device. However please check the image changelog to verify the running image.

The stock image uses a Debian Stretch distribution and Linux kernel version 4.9. The latest image can be downloaded below.

- ts7100-linux4.9-latest.tar.xz (http://ftp.embeddedarm.com/ftp/ts-arm-sbc/ts-7100-linux/distributions/ts7100-linux4.9-latest.tar.xz) (md5 (http://ftp.embeddedarm.com/ftp/ts-arm-sbc/ts-7100-linux/distributions/ts7100-linux4.9-latest.tar.bz2))

This image can then be written to the on-board eMMC flash in order to be booted on the TS-7100.

5.2 Debian Networking

> **Note:** The first physical port on the TS-7100, socket T1, is given the name "eth1", while the second port, socket T2, is "eth0".

By default, Debian Stretch does not configure any interfaces to be brought up or configured.

Debian can automatically set up the networking based on the contents of "/etc/network/interfaces.d/" files. For example, to enable DHCP for "eth0" by default on startup:

```
echo "auto eth0
iface eth0 inet dhcp" > /etc/network/interfaces.d/eth0
```

To set up a static IP:

```
echo "auto eth0
iface eth0 inet static
    address 192.168.0.50
    netmask 255.255.255.0
    gateway 192.168.0.1" > /etc/network/interfaces.d/eth0
echo "nameserver 1.1.1.1" > /etc/resolv.conf
```

To make this take effect immediately:

```
service networking restart
```

To configure other interfaces, replace "eth0" with the other network device name. Some interfaces may use predictable interface names (https://www.freedesktop.org/wiki/Software/systemd/PredictableNetworkInterfaceNames/) . For example, the traditional name for an ethernet port might be "eth1", but some devices may use "enp1s0" for PCIe, or "enx00D069C0FFEE" (the MAC address appended) for USB ethernet interfaces. Run 'ifconfig -a' or 'ip a' to get a complete list of interfaces, including the ones that are not configured.

5.2.1 Debian Wi-Fi Client

Wireless interfaces are also managed with configuration files in "/etc/network/interfaces.d/". For example, to connect as a client to a WPA network with DHCP.

Install wpa_supplicant:

```
apt-get update && apt-get install wpasupplicant -y
```

Run:

```
wpa_passphrase youressid yourpassword
```

This command will output information similar to:

```
network={
        ssid="youressid"
        #psk="yourpassword"
        psk=151790fab3bf3a1751a269618491b54984e192aa19319fc667397d45ec8dee5b
}
```

Use the hashed PSK in the specific network interfaces file for added security:

```
echo "auto wlan0
iface wlan0 inet dhcp
    wpa-ssid youressid
    wpa-psk 151790fab3bf3a1751a269618491b54984e192aa19319fc667397d45ec8dee5b" > /etc/network/interfaces.d/wlan0
```

To have this take effect immediately:

```
service networking restart
```

For more information on configuring Wi-Fi, see Debian's guide here (https://wiki.debian.org/WiFi/HowToUse#wpa_supplicant) .

5.2.2 Debian Wi-Fi Access Point

This section will discuss setting up the WiFi device as an access point that is bridged to an ethernet port. That is, clients can connect to the AP and will be connected to the ethernet network through this network bridge. The ethernet network must provide a DHCP server; this will be passed through the bridge to WiFi client devices as they connect.

The 'hostapd' utility is used to manage the access point of the device. This is usually installed by default, but can be installed with:

```
apt-get update && apt-get install hostapd -y
```

Note: The install process may start an unconfigured 'hostapd' process. This process must be killed before moving forward.

Modify the file "/etc/hostapd/hostapd.conf" to have the following lines:

```
ssid=YourWiFiName
wpa_passphrase=Somepassphrase
interface=wlan0
bridge=br0
auth_algs=3
channel=7
driver=nl80211
hw_mode=g
logger_stdout=-1
logger_stdout_level=2
max_num_sta=5
rsn_pairwise=CCMP
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP CCMP
```

Note: Refer to the kernel's hostapd documentation (http://wireless.kernel.org/en/users/Documentation/hostapd) for more wireless configuration options.

The access point can be started and tested by hand:

```
hostapd /etc/hostapd/hostapd.conf
```

Systemd auto-start with bridge to eth0

It is possible to configure the auto-start of 'hostapd' through systemd. The configuration outlined below will set up a bridge with "eth0", meaning the Wi-Fi connection is directly connected to the ethernet network. The ethernet network is required to have a DHCP server present and active on it to assign Wi-Fi clients an IP address. This setup will allow Wi-Fi clients access to the same network as the ethernet port, and the bridge interface will allow the platform itself to access the network.

Set up hostapd

First, create the file "/etc/systemd/system/hostapd_user.service" with the following contents:

```
[Unit]
Description=Hostapd IEEE 802.11 AP
Wants=network.target
Before=network.target
Before=network.service
After=sys-subsystem-net-devices-wlan0.device
After=sys-subsystem-net-devices-br0.device
BindsTo=sys-subsystem-net-devices-wlan0.device
BindsTo=sys-subsystem-net-devices-br0.device

[Service]
Type=forking
PIDFile=/run/hostapd.pid
ExecStart=/usr/sbin/hostapd /etc/hostapd/hostapd.conf -P /run/hostapd.pid -B

[Install]
WantedBy=multi-user.target
```

Then enable this in systemd:

```
systemctl enable hostapd_user.service
systemctl enable systemd-networkd
```

Set up bridging

Create the following files with the listed contents.

"/etc/systemd/network/br0.netdev"

```
[NetDev]
Name=br0
Kind=bridge
```

"/etc/systemd/network/br0.network"

```
[Match]
Name=br0

[Network]
DHCP=yes
```

"/etc/systemd/network/bridge.network"

```
[Match]
Name=eth0

[Network]
Bridge=br0
```

5.2.3 Cellular Data Network

5.2.3.1 NimbeLink Skywire modem

The CN16 XBee Socket is able to support NimbeLink Skywire Embedded modems. Information on setting up and configuring the power and USB interface for Skywire modules can be found here. Please note that there are various models of the Skywire modules that all support different interfaces. These include "cdc_ether", "cdc_ncm", USB serial, and a simple TTL UART. Both the USB ethernet and NCM interfaces present a network device to the system, while the USB serial and UART interfaces require PPP to manage the connection.

Please see the NimbeLink documentation for the specific module in use for more detailed information on establishing connection with a cellular network via the modem.

5.3 Debian Application Development

5.3.1 Debian Stretch Cross Compiling

Debian Stretch provides cross compilers from its distribution. An install on a workstation can build for the same release on other architectures. A Linux desktop or laptop PC, virtual machine, or chroot will need to be used for this. Debian Stretch for a workstation can be downloaded from here (https://www.debian.org/releases/stretch/) .

From a Debian workstation (not the target), run these commands to set up the cross compiler:

```
# Run "lsb_release -a" and verify Debian 9.X is returned.  These instructions are not
# expected to work on any other version or distribution.
su root
# Not needed for the immediate apt-get install, but used
# so we can install package:armhf for cross compiling
dpkg --add-architecture armhf
apt-get update
apt-get install curl build-essential crossbuild-essential-armhf -y
```

This will install a toolchain that can be used with the prefix "arm-linux-gnueabihf-". The standard GCC tools will start with that name, eg "arm-linux-gnueabihf-gcc".

The toolchain can now compile a simple hello world application. Create hello-world.c on the Debian workstation:

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

To compile this:

```
arm-linux-gnueabihf-gcc hello-world.c -o hello-world
file hello-world
```

This will return that the binary created is for ARM. Copy this to the target platform to run it there.

Debian Stretch supports multiarch which can install packages designed for other architectures. On workstations this is how 32-bit and 64-bit support is provided. This can also be used to install armhf packages on an x86 based workstation.

This cross compile environment can link to a shared library from the Debian root. The package would be installed in Debian on the workstation to provide headers and libraries. This is included in most "-dev" packages. When run on the arm target it will also need a copy of the library installed, but it does not need the -dev package.

```
apt-get install libcurl4-openssl-dev:armhf

# Download the simple.c example from curl:
wget https://raw.githubusercontent.com/bagder/curl/master/docs/examples/simple.c
# After installing the supporting library, curl will link as compiling on the unit.
arm-linux-gnueabihf-gcc simple.c -o simple -lcurl
```

Copy the binary to the target platform and run on the target. This can be accomplished with network protocols like NFS, SCP, FTP, etc.

If any created binaries do not rely on hardware support like GPIO or CAN, they can be run using 'qemu'.

```
# using the hello world example from before:
./hello-world
# Returns Exec format error
apt-get install qemu-user-static
./hello-world
```

5.4 Debian Installing New Software

Debian provides the apt-get system which allows management of pre-built applications. The apt tools require a network connection to the internet in order to automatically download and install new software. The update command will download a list of the current versions of pre-built packages.

```
apt-get update
```

A common example is installing Java runtime support for a system. Find the package name first with search, and then install it.

```
root@ts:~# apt-cache search openjdk
default-jdk - Standard Java or Java compatible Development Kit
default-jdk-doc - Standard Java or Java compatible Development Kit (documentation)
default-jdk-headless - Standard Java or Java compatible Development Kit (headless)
default-jre - Standard Java or Java compatible Runtime
default-jre-headless - Standard Java or Java compatible Runtime (headless)
jtreg - Regression Test Harness for the OpenJDK platform
libreoffice - office productivity suite (metapackage)
openjdk-8-dbg - Java runtime based on OpenJDK (debugging symbols)
openjdk-8-demo - Java runtime based on OpenJDK (demos and examples)
openjdk-8-doc - OpenJDK Development Kit (JDK) documentation
openjdk-8-jdk - OpenJDK Development Kit (JDK)
```

```
openjdk-8-jdk-headless - OpenJDK Development Kit (JDK) (headless)
openjdk-8-jre - OpenJDK Java runtime, using Hotspot JIT
openjdk-8-jre-headless - OpenJDK Java runtime, using Hotspot JIT (headless)
openjdk-8-jre-zero - Alternative JVM for OpenJDK, using Zero/Shark
openjdk-8-source - OpenJDK Development Kit (JDK) source files
uwsgi-app-integration-plugins - plugins for integration of uWSGI and application
uwsgi-plugin-jvm-openjdk-8 - Java plugin for uWSGI (OpenJDK 8)
uwsgi-plugin-jwsgi-openjdk-8 - JWSGI plugin for uWSGI (OpenJDK 8)
uwsgi-plugin-ring-openjdk-8 - Closure/Ring plugin for uWSGI (OpenJDK 8)
uwsgi-plugin-servlet-openjdk-8 - JWSGI plugin for uWSGI (OpenJDK 8)
java-package - Utility for creating Java Debian packages
```

In this case, the wanted package will likely be the "openjdk-8-jre" package. Names of packages can be found on Debian's wiki pages (http://wiki.debian.org/) or the packages site (https://packages.debian.org/stretch/) .

With the package name apt-get install can be used to install the prebuilt packages.

```
apt-get install openjdk-8-jre
# More than one package can be installed at a time.
apt-get install openjdk-8-jre nano vim mplayer
```

For more information on using apt-get refer to Debian's documentation here (http://wiki.debian.org/AptCLI) .

5.5 Debian Setting up SSH

To install the SSH server, install the package with apt-get:

```
apt-get install openssh-server
```

Debian Stretch by default disallows logins directly from the user "root". Additionally, SSH will not allow remote connections without a password or valid SSH key pair. This means in order to SSH to the device, a user account must first be created, and a password set:

```
useradd --create-home --shell /bin/bash newuser
passwd newuser
```

After this setup it is now possible to connect to the device as user "newuser" from a remote PC supporting SSH. On Linux/OS X this is the "ssh" command, or from Windows using a client such as PuTTY (http://www.chiark.greenend.org.uk/~sgtatham/putty/) .

5.6 Debian Starting Automatically

A systemd service can be created to start up headless applications. Create a file in /etc/systemd/system/yourapp.service

```
[Unit]
Description=Run an application on startup

[Service]
```

```
Type=simple
ExecStart=/usr/local/bin/your_app_or_script

[Install]
WantedBy=multi-user.target
```

If networking is a dependency add "After=network.target" in the Unit section. Once you have this file in place add it to startup with:

```
# Start the app on startup, but will not start it now
systemctl enable yourapp.service

# Start the app now, but doesn't change auto startup
systemctl start yourapp.service
```

Note: See the systemd documentation (http://www.freedesktop.org/software/systemd/man/systemd.service.html) for in depth documentation on services.

6 Buildroot Configuration

Note: Incomplete at this time

The full-featured stock image may be too cumbersome for some applications. Applications that require faster bootup time or a smaller root filesystem will benefit greatly from using a lighter distribution like Buildroot. To assist customers heading down this path we have forked a stable snapshot of Buildroot (specifically 2018.02) and have added on top of it everything that is required for operation with one of our products. In order to provide consistency, the Buildroot image we provide and the default configuration are fairly large; but it includes a number of tools that are present on our stock image so that transitioning from one to the other is much easier. The Buildroot configuration could be customized to provide a much smaller footprint with a faster bootup time. Our current buildroot averages about 10 seconds of bootup time (much of this is spent on networking). Reducing the configuration can bring this time down to 5 seconds from power on to login prompt.

6.1 Installing Buildroot

We offer a pre-made filesystem tarball that is based on our default Buildroot configuration. It can downloaded here: ftp://ftp.embeddedarm.com/ts-arm-sbc/ts-7100-linux/distributions/ts7100-Buildroot-2018.02-latest.tar.xz

Using that tarball, it's possible to create a bootable eMMC for the TS-7100.

The default configuration was designed to be as close to our stock Debian distribution. This includes our ts7100-utils (https://github.com/embeddedarm/ts7100-utils) like tsmicroctl, our TS-SILO monitor daemon, drivers and firmware for the WiFi and Bluetooth module.

6.2 Building Buildroot

The Buildroot image can be built from source if needed. This process will create a cross compiler, use that to build all target applications including the kernel, and then create a filesystem tarball of a bootable image. The following instructions can be used to build Buildroot.

Clone the repository:

```
git clone https://github.com/embeddedarm/buildroot-2018.02
cd buildroot-2018.02/
```

Configure the build:

```
make ts7100_defconfig
```

At this point, the default configuration can be modified if desired:

```
make menuconfig
```

And finally, start the build process:

```
make
```

The buildroot process can take a large amount of time to build, depending on available system resources. Note that if any changes occur in the config file, it is recommended to clean the build tree and start the process over. Additionally, ccache is enabled by default in the default configuration. This will speed up a re-build of Buildroot after a clean. However it will take up additional hard drive space, and if any changes are made to the cross compiler configuration the ccache directory must be removed first. See the Buildroot manual (https://buildroot.org/downloads/manual/manual.html#ccache) for more information about ccache and Buildroot.

Once it is finished building, Buildroot will output a filesystem tarball to "output/images/rootfs.tar.xz". This file can be used with Installing Buildroot in lieu of the tarball provided on our FTP site.

6.3 Configuring the Network

Buildroot implements the 'ip', 'ifconfig', and 'route' commands to manipulate the settings of interfaces. The first ethernet interface is set up to come up automatically with our default configuration. The interfaces can also be manually set up:

```
# Bring up the CPU network interface
ifconfig eth0 up

# Set an ip address (assumes 255.255.255.0 subnet mask)
ifconfig eth0 192.168.0.50

# Set a specific subnet
ifconfig eth0 192.168.0.50 netmask 255.255.0.0

# Configure your route.  This is the server that provides your internet connection.
route add default gw 192.168.0.1

# Edit /etc/resolv.conf for your DNS server
echo "nameserver 192.168.0.1" > /etc/resolv.conf
```

Most commonly, networks will offer DHCP which can be set up with one command:

```
# To setup the default CPU ethernet port
udhcpc -i eth0
# You can configure all ethernet ports for a DHCP response with
udhcpc
```

To have network settings take effect on startup in Buildroot, edit /etc/network/interfaces:

```
# interface file auto-generated by buildroot

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp
  pre-up /etc/network/nfs_check
  wait-delay 15
```

Note: The default network startup may timeout on some networks. This can be resolved by adding either of the following under the "iface eth0 inet dhcp" section: "udhcpc_opts -t 0" to infinitely retry, or "udhcpc_opts -t 5" to fail after five attempts.

See the man page for interfaces(5) for further information on the syntax of the file.

For more information on network configuration in general, Debian provides a great resource here (http://wiki.debian.org/Network) that can be readily applied to Buildroot in most cases.

6.4 Installing New Software

By default, Buildroot does not include a package manager. This means installing software directly on the platform can be cumbersome and is not the intended path. It is possible to modify the Buildroot configuration to include additional packages. See the Building Buildroot section for information on adding new packages.

If a desired package is not available in Buildroot, there are a number of options available when moving forward. It is possible to add packages to the build process, though this does require some knowledge of Buildroot internals. Another option is to use the cross compiler that is output by buildroot in order to compile packages on a host system and then copy them over to the target. It is also possible to install a toolchain directly on the device, and compile applications natively. The last option is the least recommended as it greatly increases the final image size and adds unnecessary complexity.

6.5 Setting up SSH

The default configuration has Dropbear set up. Dropbear is a lightweight SSH server.

Make sure the device is configured on the network and set a password for the remote user. SSH will not allow remote connections without a password set. The default configuration does not set a password for the root user, nor are any other users configured.

```
passwd root
```

After this setup it is now possible to connect from a remote PC supporting SSH. On Linux/OS X this is the "ssh" command, or from Windows using a client such as putty (http://www.chiark.greenend.org.uk/~sgtatham/putty/) .

6.6 Starting Automatically

From Buildroot the most straightforward way to add an application to startup is to create a startup script. This is an example simple startup script that will toggle the red led on during startup, and off during shutdown. In this case the file is named customstartup, but you can replace this with any application name as well.

Edit the file /etc/init.d/S99customstartup to contain the following. Be sure to set the script as executable!

```
#! /bin/sh
# /etc/init.d/customstartup

case "$1" in
  start)
    echo 1 > /sys/class/leds/red-led/brightness
    ## If you are launching a daemon or other long running processes
    ## this should be started with
    # nohup /usr/local/bin/yourdaemon &
    ;;
  stop)
    # if you have anything that needs to run on shutdown
    echo 0 > /sys/class/leds/red-led/brightness
    ;;
  *)
    echo "Usage: customstartup start|stop" >&2
    exit 3
    ;;
esac

exit 0
```

Note: The $PATH variable is not set up by default in init scripts so this will either need to be done manually or the full path to your application must be included.

To manually start and stop the script:

```
/etc/init.d/S99customstartup start
/etc/init.d/S99customstartup stop
```

7 Backup / Restore

7.1 Creating A Backup / Production Image

Note: This section is incomplete at this time.

7.2 Restoring Stock / Backup / Production Image

7.2.1 Booted from USB / NFS

These instructions assume the TS-7100 is booted to Linux from network via NFS or USB mass storage. They also assume that the eMMC is unmodified, with a single partition. If the partition table has been modified, a utility such as 'gparted' or 'fdisk' may be needed to remove the existing partition table and recreate it with a single partition. Note that the partition table must be "MBR" or "msdos", the "GPT" partition table format is not supported by U-Boot.

Once booted to any device that is not the eMMC:

```
# Verify nothing else has the first eMMC partition mounted
umount /dev/mmcblk0p1

mkfs.ext3 /dev/mmcblk0p1
mount /dev/mmcblk0p1 /mnt/emmc
wget http://ftp.embeddedarm.com/ftp/ts-arm-sbc/ts-7100-linux/distributions/ts7100-linux4.9-latest.tar.xz
tar -xf ts7100-linux4.9-latest.tar.xz -C /mnt/emmc
umount /mnt/emmc
sync
```

Note: The ext4 filesystem can be used instead of ext3, but it may require additional options. U-Boot does not support the 64bit addressing added as the default behavior in recent revisions of mkfs.ext4. If using e2fsprogs 1.43 or newer, the options "-O ^64bit,^metadata_csum" must be used with ext4 for proper compatibility. Older versions of e2fsprogs do not need these options passed nor are they needed for ext3.

Once written, the files on disk can be verified to ensure they are the same as the source files in the archive. To do so, run the following commands:

```
mount /dev/mmcblk0p1 /mnt/emmc
cd /mnt/emmc/
md5sum --quiet -c md5sums.txt
cd -
umount /mnt/emmc
sync
```

The 'md5sum' command will report any differences between files and their checksums. Any differences are an indication of failure to write data or a damaged disk. Note that the "/md5sums.txt" file is present in our stock tarballs and is created in custom images as a part of our scripts to do so. This file may not be present in custom images created without our tools, or it may be present but not properly updated. This will result in reported errors.

8 Compile the Kernel

Compiling the kernel requires an armhf toolchain. We recommend development under Debian Stretch which includes an armhf compiler in the repositories. See the Debian Stretch cross compilation section for instructions on installing a proper cross compiler.

```
git clone git clone https://github.com/embeddedarm/linux-4.9.y
cd linux-4.9.y

# These next commands set up some necessary environment variables
export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabihf-
export LOADADDR=0x80800000

# This sets up the default configuration that we ship with
make tsimx6ul_defconfig

## Make any changes in "make menuconfig" or driver modifications, then compile
make && make zImage && make modules
```

The following will install the kernel and modules to a temporary directory, and then pack them up in to a single tarball:

```
TEMPDIR=$(mktemp -d)
mkdir "${TEMPDIR}/boot/"
cp arch/arm/boot/zImage "${TEMPDIR}"/boot/zImage
cp arch/arm/boot/dts/imx6ul*ts*.dtb  "${TEMPDIR}"/boot/
INSTALL_MOD_PATH="${TEMPDIR}" make modules_install
make headers_install INSTALL_HDR_PATH="${TEMPDIR}"
tar czf linux-tsimx6ul-"$(date +"%Y%m%d")"-"$(git describe --abbrev=8 --dirty --always)".tar.gz -C "${TEMPDIR}" .
rm -rf "${TEMPDIR}"
```

This will output a tarball with the kernel version and short git hash, as well as the date the tarball was created. For example "linux-tsimx6ul-20190823-v4.9.171-60-g01e2117e.tar.gz"

This tarball can be directly unpacked to the root folder of a bootable media for the device. It is also possible to unpack it directly on a booted system, however we do not recommend doing so on an active deployed system without extensive testing.

```
# Unpack it to a mounted disk, this assumes the disk is mounted to "/mnt"
tar xf linux-tsimx6ul...tar.gz -C /mnt

# Unpack it to the root directory of a booted system
tar xf linux-tsimx6ul...tar.gz -C /
```

9 Production Mechanism

The TS-7100's U-Boot has the ability to locate and run a U-Boot script file named /tsinit.ub on the root of a USB drive. This process occurs when attempting to boot to the U-Boot shell. If this script exists, U-Boot will load and run it automatically. This is intended for the initial production of units and allows mass programming various media from a USB mass storage device.

The USB blasting image can be downloaded here (http://ftp.embeddedarm.com/ftp/ts-arm-sbc/ts-7100-linux/usb-blaster/tsimx6ul_usb_blaster-latest.tar.bz2) . This includes a basic Linux kernel and a small initramfs that will mount the USB drive at "/mnt/usb/" and execute "/mnt/usb/blast.sh".

The blast image and scripts require a minimum of 50 MB; this plus any disk images or tarballs used dictate the minimum disk size required. The USB drive must have at least 1 partition, with the first partition being formatted ext2/3 or fat32/vfat.

Note:	The ext4 filesystem can be used instead of ext3, but it may require additional options. U-Boot does not support the 64bit addressing added as the default behavior in recent revisions of mkfs.ext4. If using e2fsprogs 1.43 or newer, the options "-O ^64bit,^metadata_csum" must be used with ext4 for proper compatibility. Older versions of e2fsprogs do not need these options passed nor are they needed for ext3.

```
# This assumes USB drive is /dev/sdc:
sudo mkfs.ext3 /dev/sdc1
sudo mkdir /mnt/usb/
sudo mount /dev/sdc1 /mnt/usb/
sudo tar --numeric-owner -xf /path/to/tsimx6ul_usb_blaster-latest.tar.bz2 -C /mnt/usb/
```

At this point, disk images or tarballs would be copied to the /mnt/usb/ folder and named as noted below. The latest disk images we provide can be downloaded from our FTP site, see the backup and restore section for links to these files. Note that the script expects images and tarballs to have specific names. When using an ext* filesystem, symlinks can be used.

The formatted USB drive boots into a small buildroot initramfs environment with filesystem and partitioning tools installed. This can be used to format SD, eMMC, or other disks. The buildroot starts up and calls /blast.sh on the USB device. By default this script is set up to look for a number of of specific files on the USB disk and write to media on the host device. Upon completion of the script the green or red LEDs will blink to visually indicate a pass or fail of the script. This script can be used without modification to write images from USB with these filenames:

SD Card	sdimage.tar.bz2	Tar of the filesystem. This will repartition the SD card to 1 ext4 partition and extract this tar to the filesystem. If present, a /md5sums.txt will be checked and every file can be verified on the filesystem. This md5sums file is optional and can be omitted, but it must not be blank if present.
	sdimage.dd.bz2	Disk image of the card. This will be written to mmcblk0 directly. If present a sdimage.dd.md5 will cause the written data on the SD card to be read back and verified against this checksum.
eMMC	emmcimage.tar.bz2	Tar of the filesystem. This will repartition the eMMC to 1 ext4 partition and extract this tar to the filesystem. If present, a /md5sums.txt will be checked and every file can be verified on the filesystem. This md5sums file is optional and can be omitted, but it must not be blank if present.
	emmcimage.dd.bz2	Disk image of the card. This will be written to mmcblk1 directly. If present a emmcimage.dd.md5 will cause the written data on the eMMC to be read back and verified against this checksum.

Most users should be able to use the above script without modification. Our buildroot sources are available from our github repo (https://github.com/embeddedarm/buildroot-2017.05) . To build the whole setup and create a USB drive, the following commands can be used. This will wipe any data on the specified partition and replace it with an ext2 formatted filesystem. This filesystem will have all of the necessary files written to it to create a bootable USB drive. Note that this must be the first partition of the disk.

```
# Assuming /dev/sdc1 is your usb drive's first partition
make tsimx6ul_defconfig && make && sudo ./make_usb_prog.sh /dev/sdc1 tsimx6ul
```

10 Features

10.1 ADC

TS-7100 ADC

10.1.1 0-50 V

TS-7100 50V ADC

10.1.2 0-12 V

TS-7100 12V ADC

10.1.3 4-20 mA

TS-7100 20mA ADC

10.2 Battery Backed RTC

The TS-7100 implements an STMicro "M41T00S" Battery Backed RTC using an external battery. This RTC is connected to the CPU via I2C and is handled by the kernel and is presented as a standard RTC device in Linux. On the TS-7100 series, the RTC is located on the CPU module with the battery backup located on the I/O board.

The TS-7100-Z I/O board provides battery backed power to the RTC via a replaceable CR1632 coin cell.

10.3 Bluetooth

The Wi-Fi option for the unit also includes a bluetooth 4.0 LE module. Both Wi-Fi and Bluetooth can be active at the same time. However, in order for bluetooth to function the Wi-Fi device must first be brought up to load the necessary firmware. After the Wi-Fi is active, the Bluetooth module can be activated with the following commands:

```
# Ensure that the Wi-Fi device is active
ifconfig wlan0 up

# Enable Bluetooth, and load the driver firmware
echo BT_POWER_UP > /dev/wilc_bt
echo BT_DOWNLOAD_FW > /dev/wilc_bt
echo BT_FW_CHIP_WAKEUP > /dev/wilc_bt

hciattach /dev/ttymxc2 any 115200 noflow
hciconfig hci0 up
hcitool cmd 0x3F 0x0053 00 10 0E 00 01
stty -F /dev/ttymxc2 921600 crtscts
```

The Bluetooth module is now set up, and is running at 921600 baud with full flow control. At this point, the device is fully set up and can be controlled with various components of bluez-tools. For example, to do a scan of nearby devices:

```
hcitool lescan
```

This will return a list of devices such as:

```
3C:A3:08:XX:XX:XX Device_Name
```

Bluez has support for many different profiles for HID, A2DP, and many more. Refer to the Bluez documentation for more information.

Please note that the Bluetooth module requires the modem control lines CTS and RTS as flow control.

The module supports some other commands as well:

```
# Allow the BT chip to enter sleep mode
echo BT_FW_CHIP_ALLOW_SLEEP > /dev/wilc_bt

# Power down the BT radio when not in use
echo BT_POWER_DOWN > /dev/wilc_bt
```

10.4 CAN

The TS-7100-Z CPU has a single FlexCAN port that uses the Linux SocketCAN implementation. The port can be set up and used with the following command:

```
ip link set can0 up type can bitrate 1000000
```

The CAN transceiver is automatically controlled by the kernel. If the interface is brought up in Linux then the transceiver will be enabled. By default when the kernel boots, the interface is down, and therefore the transceiver is disabled.

At this point, the port can be used with standard SocketCAN libraries. In Debian, we provide the utilities 'cansend' and 'candump' to test the ports or as a simple packet send/receive tool. In order to test the port, tie CAN_H to the CAN_H pin of the bus, doing the same for the CAN_L pin. Then use the following commands:

```
candump can0
# This command will echo all data received on the bus to the terminal

cansend can0 7Df#03010c
#This command will send out the above CAN packet to the bus
```

The above example packet is designed to work with the Ozen Elektronik myOByDic 1610 ECU simulator to read the RPM speed. In this case, the ECU simulator would return data from candump with:

```
 <0x7e8> [8] 04 41 0c 60 40 00 00 00
 <0x7e9> [8] 04 41 0c 60 40 00 00 00
```

In the output above, columns 6 and 7 are the current RPM value. This shows a simple way to prove out the communication before moving to another language.

The following example sends the same packet and parses the same response in C:

```c
#include <stdio.h>
#include <pthread.h>
#include <net/if.h>
#include <string.h>
#include <unistd.h>
#include <net/if.h>
#include <sys/ioctl.h>
#include <assert.h>
#include <linux/can.h>
#include <linux/can/raw.h>

int main(void)
{
        int s;
        int nbytes;
        struct sockaddr_can addr;
        struct can_frame frame;
        struct ifreq ifr;
        struct iovec iov;
        struct msghdr msg;
        char ctrlmsg[CMSG_SPACE(sizeof(struct timeval)) + CMSG_SPACE(sizeof(__u32))];
        char *ifname = "can0";

        if((s = socket(PF_CAN, SOCK_RAW, CAN_RAW)) < 0) {
                perror("Error while opening socket");
                return -1;
        }

        strcpy(ifr.ifr_name, ifname);
        ioctl(s, SIOCGIFINDEX, &ifr);
        addr.can_family  = AF_CAN;
        addr.can_ifindex = ifr.ifr_ifindex;

        if(bind(s, (struct sockaddr *)&addr, sizeof(addr)) < 0) {
                perror("socket");
                return -2;
        }

        /* For the ozen myOByDic 1610 this requests the RPM guage */
        frame.can_id  = 0x7df;
        frame.can_dlc = 3;
        frame.data[0] = 3;
        frame.data[1] = 1;
        frame.data[2] = 0x0c;

        nbytes = write(s, &frame, sizeof(struct can_frame));
        if(nbytes < 0) {
                perror("write");
                return -3;
        }

        iov.iov_base = &frame;
        msg.msg_name = &addr;
        msg.msg_iov = &iov;
        msg.msg_iovlen = 1;
        msg.msg_control = &ctrlmsg;
        iov.iov_len = sizeof(frame);
        msg.msg_namelen = sizeof(struct sockaddr_can);
        msg.msg_controllen = sizeof(ctrlmsg);
        msg.msg_flags = 0;

        do {
                nbytes = recvmsg(s, &msg, 0);
                if (nbytes < 0) {
                        perror("read");
                        return -4;
                }

                if (nbytes < (int)sizeof(struct can_frame)) {
                        fprintf(stderr, "read: incomplete CAN frame\n");
                }
        } while(nbytes == 0);
```

```
        if(frame.data[0] == 0x4)
                printf("RPM at %d of 255\n", frame.data[3]);

        return 0;
}
```

See the Kernel's CAN documentation here (https://www.kernel.org/doc/Documentation/networking/can.txt) . Other languages have bindings to access CAN such as Python using C-types (https://bitbucket.org/hardbyte/python-can) , Java using JNI (https://github.com/entropia/libsocket-can-java) .

10.5 CPU

This device uses the i.MX6UL CPU running at 528MHz or 696MHz using a Cortex-A7 core targeting low power consumption.

Refer to NXP's documentation (http://www.nxp.com/products/microcontrollers-and-processors/arm-processors/i.mx-applications-processors/i.mx-6-processors/i.mx6qp/i.mx-6ultralite-processor-low-power-secure-arm-cortex-a7-core:i.MX6UL) for more detailed information on the CPU.

10.6 GPIO

> **Note:** This section is incomplete at this time.

The i.MX6UL CPU and FPGA GPIO are exposed using a kernel character device. This interface provides a set of files and directories for interacting with GPIO which can be used from any language that interact with special files in linux using ioctl() or similar. For our platforms, we pre-install the "libgpiod" library and binaries. Documentation on these tools can be found here (https://git.kernel.org/pub/scm/libs/libgpiod/libgpiod.git/tree/README) . This section only covers using these userspace tools and does not provide guidance on using the libgpiod library in end applications. Please see the libgpiod documentation for this purpose.

A user with suitable permissions to read and write /dev/gpiochip* files can immediately interact with GPIO pins. For example, to see if input power has failed:

```
gpioget 4 0
```

Multiple pins in the same chip can be read simultaneously by passing multiple pin numbers separated by spaces.

To write to a pin, the 'gpioset' command is used. For example, to set Relay 1:

```
gpioset 4 4=1
```

Multiple pins in the same chip can be set simultaneously by passing multiple pin=value pairs separated by spaces.

If a call with 'gpioset' or 'gpioget' fails with "Device or resource busy," that means that specific GPIO is claimed by another device. The command 'cat /sys/kernel/debug/gpio' can be used to get a list of all of the system GPIO and what has claimed them.

The 'gpiomon' tool can be used to monitor pins for changes.

Chip	Pin	Location
0	0	AIN 4 or Digital Input AIN 4 on CN32 Terminal
0	4	AIN 0 on CN32 Terminal
0	5	AIN 1 or Digital Input AIN 1 on CN32 Terminal
0	8	AIN 2 or Digital Input AIN 2 on CN32 Terminal
0	9	AIN 3 or Digital Input AIN 3 on CN32 Terminal
0	18	CPU Board Red LED
0	19	CPU Board Green LED
4	0	Power Input Failure
4	1	FPGA Interrupt input pin
4	4	Relay 1 on CN32 Terminal
4	5	Relay 2 on CN32 Terminal
5	0	DIO 1 Out or PWM on CN32 Terminal
5	1	DIO 2 Out or PWM on CN32 Terminal
5	2	DIO 1 In on CN32 Terminal
5	3	DIO 2 In on CN32 Terminal
5	4	DIO 3 In on CN32 Terminal
5	5	FPGA DIO 06
5	6	Digital In 1 on CN32 Terminal
5	7	Digital In 2 on CN32 Terminal
5	8	Digital In 3 on CN32 Terminal
5	9	AIN 1 4-20 mA current loop enable
5	10	AIN 2 4-20 mA current loop enable
5	11	AIN 3 4-20 mA current loop enable
5	12	Reserved
5	13	Reserved
5	14	AIN 4 4-20 mA current loop enable
5	15	High-Side Switch or HSPWM
6	1	AIN 1 0-12 V meas. mode [1]
6	2	AIN 2 0-12 V meas. mode [1]
6	3	AIN 3 0-12 V meas. mode [1]
6	4	AIN 4 0-12 V meas. mode [1]
6	5	en usb host 5v
6	6	eth PHY reset
6	7	wifi reset
6	8	I/O Board Red LED
6	9	I/O Board Green LED
6	12	Reserved

Chip	Pin	Location
6	13	DIO 3 Out or PWM on CN32 Terminal
6	14	HSPWM enable
6	15	PWM enable, both OE and dat
7	0	Touchscreen IRQ
7	2	FPGA Strapping Pin
7	3	FPGA Strapping Pin
7	4	FPGA Strapping Pin
7	5	FPGA Strapping Pin
7	6	Data 0: Select 3.3 V power on CN16 XBee Socket [2]
		Data 1: Select 4 V power on CN16 XBee Socket [2]
7	8	Enable MODEM on CN16 XBee Socket [3]
7	9	Enable USB interface on CN16 XBee Socket [4]
7	10	I/O over-current/over-voltage breaker tripped [5]
7	11	FPGA Strapping Pin
7	12	FPGA Strapping Pin
7	13	Reserved
7	14	LCD backlight enable

1. ↑ [1.0] [1.1] [1.2] [1.3] This bit is read only. Clearing the associated current loop enable bit will set this bit, setting the CL enable will clear this bit
2. ↑ [2.0] [2.1] To disable power on this pin, set the GPIO as an input with 'gpioset' or otherwise
3. ↑ Default enabled on P2 PCB
4. ↑ This will relocate the USB channel connected to the top USB host port
5. ↑ This bit must be cleared manually after a trip to de-assert the associated IRQ

10.6.1 Digital Inputs

The digital inputs on the TS-7100-Z are capable of supporting various voltage ranges and input modes. The digital inputs support dry contact switches as well as a driven input voltage. The table below lists each digital input, the bank and pin number for reading the input, the maximum input voltage range, the threshold voltages, as well as the location of the input. VIH Min is the minimum voltage on the input to trigger a logic 1 input. VIL Max is the maximum voltage on the input to trigger a logic 0 input. All of the digital inputs are hysteretic. The driving input must be able to at least sink current to drive the input low, but all digital inputs are compatible with push-pull drivers.

Input Name	Bank	Pin	V Range	VIH Min	VIL Max	Location
Digital In 1	5	6	0-30 V	~2.57 V	~0.95 V	CN32 Terminal, pin 9
Digital In 2	5	7	0-30 V	~2.57 V	~0.95 V	CN32 Terminal, pin 11
Digital In 3	5	8	0-30 V	~2.57 V	~0.95 V	CN32 Terminal, pin 13
DIO 1 In [1]	5	2	0-30 V	~2.54 V	~0.90 V	CN32 Terminal, pin 14
DIO 2 In [1]	5	3	0-30 V	~2.54 V	~0.90 V	CN32 Terminal, pin 16
DIO 3 In [1]	5	4	0-30 V	~2.54 V	~0.90 V	CN32 Terminal, pin 18
AIN 1 In [2]	0	5	0-12 V	~8.60 V	~7.90 V	CN32 Terminal, pin 25
AIN 2 In [2]	0	8	0-12 V	~8.60 V	~7.90 V	CN32 Terminal, pin 23
AIN 3 In [2]	0	9	0-12 V	~8.60 V	~7.90 V	CN32 Terminal, pin 21
AIN 4 In [2]	0	0	0-12 V	~8.60 V	~7.90 V	CN32 Terminal, pin 19

1. ↑ [1.0] [1.1] [1.2] This GPIO should only be read as an input. Its value reflects the voltage on the physical CN32 pin, regardless of output status
2. ↑ [2.0] [2.1] [2.2] [2.3] The AIN pins can be used as Digital Inputs, but require software changes first. See the ADC section for more information

10.6.2 Digital Outputs

The TS-7100-Z supports a handful of digital output pins. These are able to act as high-current low-side switches. The table below lists each digital output, the bank and pin number for accessing it, the maximum voltage rating, the maximum current output, as well as the location of the pin.

DIO Name	Bank	Pin	Max V Rating	Max A Rating	Location
DIO 1 Out	5	0	30 V	700 mA (sink) [1]	CN32 Terminal, pin 14
DIO 2 Out	5	1	30 V	700 mA (sink) [1]	CN32 Terminal, pin 16
DIO 3 Out	6	13	30 V	700 mA (sink) [1]	CN32 Terminal, pin 18
High-Side Switch	5	15	48 V [2]	300 mA (source) [3]	CN32 Terminal, pin 27

1. ↑ [1.0] [1.1] [1.2] Not to exceed 1000 mA total across all three Digital I/O, doing so will cause the over-current breaker to trip
2. ↑ The output voltage is the same as the TS-7100-Z input voltage
3. ↑ Exceeding 330 mA will cause the over-current breaker to trip

10.6.2.1 Digital Output Over-Current Breaker

The TS-7100-Z I/O PCB in combination with the FPGA on the TS-7100, implements an electronic over-current breaker. When this breaker is tripped all three DIO Out paths will be disable, High-Side Switch output will be disabled, analog current loops will be disabled, and the red LED on the TS-7100-Z I/O board will be illuminated. That is, digital outputs will cease to sink or source any amount of current, and the AIN inputs will have 4-20 mA input disabled. The tripped breaker will also trigger a DIO fault breaker interrupt as well as set the associated GPIO

output, 7 10. The GPIO output, 7 10, must be cleared manually in order to reset the IRQ output. However, once the breaker trips, and the trip condition is cleared; all relevant GPIO settings can immediately be re-enabled without clearing this GPIO output bit.

Trip Conditions

See the table above for each DIO channel's maximum current rating. Note that the breaker does *NOT* enforce these ratings per DIO channel. The breaker will trip if the combined total amount of current sunk from all three digital outputs exceeds 1 A.

See the table above for the High-Side Switch's maximum current rating. If the rated max supply current is exceeded, the breaker will trip.

Note that all of these are in parallel. If the combined DIO sink current OR High-Side Switch current is exceeded, then the breaker will trip. The over-current breaker will also disable analog 4-20 mA current loop measurements.

10.7 eMMC Interface

> **Note:** This section is incomplete at this time.

The i.MX6UL SD card controller supports the MMC specification, the TS-7100 includes a soldered down eMMC IC to provide on-board flash media.

Our default software image contains 2 partitions:

Device	Contents
/dev/mmcblk0	eMMC block device
/dev/mmcblk0boot0	eMMC boot partition
/dev/mmcblk0boot1	eMMC boot partition
/dev/mmcblk0p1	Full Debian Linux partition

This platform includes an eMMC device, a soldered down MMC flash device. Our off the shelf builds are 4 GB, but up to 64 GB are available for customized builds. The eMMC flash appears to Linux as an SD card at /dev/mmcblk0. The eMMC includes additional boot partitions that are used by U-Boot and are not affected by the eMMC partition table.

The eMMC module has a similar concern by default to SD cards in that they should not be powered down during a write/erase cycle. However, this eMMC module includes support for setting a fuse for a "Write Reliability" mode, and a "psuedo SLC (pSLC)" mode. With both of these enabled all writes will be atomic to 512 B and each NAND cell will be treated as a single layer rather than a multi-layer cell. If a sector is being written during a power loss, a block is guaranteed to have either the old or new data. Even in cases where the wrong data is present on the next boot, fsck is often able to deal with the older data being present in a 512 B block. The downsides to setting these modes are that it will reduce the overall write speed and halve the available space on the eMMC to roughly 1.759 GiB. Please note that even with these settings, Technologic Systems strongly recommends designing the end application to eliminate any situations where a power-loss event can occur while any disk is mounted as read/write. The TS-SILO option available on certain TS-7100 I/O boards can help to eliminate the dangerous situation.

The 'mmc-utils' package is used to enable these modes. The command is pre-installed on the latest image. Additionally we have created a script to safely enable the write reliability and pSLC modes. Since the U-Boot binary and environment reside on the eMMC, care must be taken to save the current state of the boot partitions, enable the modes, restore the boot partitions, and re-enable proper booting options. This script can be used in combination with the production mechanism scripting to complete these steps as part of an end application production process.

WARNING:	Enabling these modes causes all data on the disk to become invalid and must be rewritten. Do not attempt to run the 'mmc' commands from the script individually, all steps in the script must occur as they are or the unit may be unable to boot. If there are any failures of the script, care must be taken to resolve any issues while the unit is still booted or it may fail to boot in the future.

Note:	Enabling these modes is a one-way operation, it is not possible to undo them once they are made. Because of this, setting these eMMC modes will invalidate Technologic Systems' return/replacement warranty on the unit. See the warranty section for more information on this.

The 'emmc_reliability' script can be found in the TS-7100 utilities github repository (https://github.com/embeddedarm/ts7100-utils) .

The script must be run when boot from any media other than eMMC, such as NFS, or USB. No partition of the eMMC disk can be mounted when these commands are run. Doing so may result in corruption or inability for the unit to boot. Once the pSLC mode is enabled, all data on the disk will become invalid. This means the partition table will need to be re-created, the filesystems formatted, and all filesystem contents re-written to disk. This is why we recommend using this script in conjunction with the production mechanism scripting. The 'emmc_reliability' script can be run first, then the rest of the production script can create and format the partitions as well as write data to disk.

The script requires a single argument, the device node of the eMMC disk, and will output verbosely to stderr. Any specific errors will also be printed out on stderr.

Example usage:

```
./emmc_reliability /dev/mmcblk0
```

Upon successful run, the script will return 0. Any errors will return a positive code. See the script for detailed error code information.

10.8 Ethernet

The NXP processor implements two 10/100 ethernet controllers with support built into the Linux kernel. Standard Linux utilities such as ifconfig/ip can be used to control this interface. See the Configuring the Network section for more details. For the specifics of this interface see the CPU manual (http://www.nxp.com/products/microcontrollers-and-processors/arm-processors/i.mx-applications-processors/i.mx-6-processors/i.mx6qp/i.mx-6ultralite-processor-low-power-secure-arm-cortex-a7-core:i.MX6UL?fpsp=1&tab=Documentation_Tab#) .

10.9 FPGA

10.9.1 FPGA Registers

The TS-7100 FPGA is connected to the CPU over the WEIM bus. This provides 8-bit, 16-bit, or 32-bit access to the FPGA mapped at 0x5000_0000.

For example, to read the FPGA information at the first register of the syscon:

```
root@ts-imx6ul:~# memtool md -l 0x50004000+4
50004000: 00000006
```

Offset	Description
0x0000	UART 16550 #0
0x0100	Opencore SPI controller #0
0x0120	Opencore SPI controller #1
0x4000	FPGA Syscon

10.9.1.1 FPGA 16550

This FPGA includes a 16550 UART peripheral that can be used as a standard Linux serial port. It is not recommended to interact directly with these registers.

10.9.1.2 FPGA SPI

This FPGA includes a pair of SPI master devices. These are used for the FRAM memory, accessing the flash used for the LCD splash screen image, and the LCD touch screen itself. All of these operations are handled via the Linux kernel. It is not recommended to interact directly with these registers

10.9.1.3 FPGA Syscon

The FPGA syscon is the main system control block of the FPGA. Contained in this region is the FPGA GPIO, PWM, and IRQ control. It is not recommended to interact directly with these registers unless directed to do so by other manual sections.

Some registers are dual purpose, having separate read and write functionality; while others may only have write functionality. Registers that do not read and write the same are indicated with "(RD)" and "(WR)" notation. All other registers read and write the same data set. Any unlisted register addresses are Reserved / Undefined.

Offset	Bits	Description
0x00 (RD)	31:0	Revision and Info Register.
0x10 (RD)	15:0	DIO bank 0 Pin State
0x10 (WR)	15:0	DIO bank 0 Data Set
0x12 (WR)	15:0	DIO bank 0 Output Enable Set
0x14 (RD)	15:0	DIO bank 0 Data
0x14 (WR)	15:0	DIO bank 0 Data Clear
0x16 (RD)	15:0	DIO bank 0 Output Enable
0x16 (WR)	15:0	DIO bank 0 Output Enable Clear
0x20 (WR)	31:0	Fractional clock generator [1]
0x24 (RD)	31:0	IRQ Status
0x24 (WR)	31:0	Fractional PWM generator
0x40 (RD)	15:0	DIO bank 1 Pin State
0x40 (WR)	15:0	DIO bank 1 Data Set
0x42 (WR)	15:0	DIO bank 1 Output Enable Set
0x44 (RD)	15:0	DIO bank 1 Data
0x44 (WR)	15:0	DIO bank 1 Data Clear
0x46 (RD)	15:0	DIO bank 1 Output Enable
0x46 (WR)	15:0	DIO bank 1 Output Enable Clear
0x48	31:0	IRQ mask
0x50 (RD)	15:0	DIO bank 2 Pin State
0x50 (WR)	15:0	DIO bank 2 Data Set
0x52 (WR)	15:0	DIO bank 2 Output Enable Set
0x54 (RD)	15:0	DIO bank 2 Data
0x54 (WR)	15:0	DIO bank 2 Data Clear
0x56 (RD)	15:0	DIO bank 2 Output Enable
0x56 (WR)	15:0	DIO bank 2 Output Enable Clear

1. ↑ Note that this is also used for UART clock generation.

10.9.1.4 FPGA IRQs

Bit	Description
31:17	Reserved
16	Touch Screen IRQ
15:13	Reserved
12	DIO Fault Breaker IRQ
11	Reserved
10	Opencore SPI Controller #1 IRQ
9	Opencore SPI Controller #0 IRQ
8:1	Reserved
0	UART #0 IRQ

10.10 FRAM

The unit supports an optional non-volatile Ferroelectric RAM (FRAM) device. The Fujitsu MB85RS16N is a 2kbyte device, in a configuration not unlike an SPI EEPROM. However, the nature of FRAM means it is non-volatile, incredibly fast to write, and is specified with 1 trillion read/write cycles per each byte and a 200 year data retention. The device is connected to Linux and presents itself as a flat file that can be read and written like any standard Linux file.

The EEPROM file can be found at /sys/bus/spi/devices/spi32766.0/eeprom

10.11 I2C

The i.MX6UL supports standard I2C at 100 kHz, or using fast mode for 400 kHz operation.

The kernel makes the I2C available at /dev/i2c-# as noted above. Linux i2c-tools (i2cdetect, i2cget, i2cset) can be used to interface with devices, or custom clients can be written (https://github.com/embeddedarm/linux-4.9.y/blob/master/Documentation/i2c/dev-interface) .

10.12 Interrupts

Note: This section is incomplete at this time.

10.13 LCD

10.13.1 Splash Screen

The LCD on this device is able to display a customizable splash screen immediately after power on. This is accomplished by the on-board FPGA reading data from an SPI NOR flash device, and writing it directly to the LCD in its SPI mode. This image is then left on the screen during the rest of the bootup process until the kernel takes over and drives the LCD in parallel RGB mode. The SPI NOR flash can be updated from userspace in Linux to write the new splash screen data.

Since the LCD is a 240x320 display, the final image needs to be formatted to fit this resolution and put in a binary format that the LCD can properly display. We have created a simple script to accomplish this. The scrip uses ImageMagick's 'convert' tool, as well as 'gcc'. The script will take an input file, translate it to fit the display, build a small tool from C sources, run the translated image through said tool in order to put the image in the correct byte ordering, and then clean up all temporary files. Additionally, a PNG image is output by the tool to be used as a sample reference of what the translated image looks like.

A simple conversion would look like the following:

```
./splash-convert Logo.png

# Background color can also be passed:
./splash-convert -background blue NewLogo.jpg

# Any arguments to 'convert' can be arbitrarily passed:
./splash-convert -rotate 120 Logo.png
```

This will output "splash.out" which can be written directly to the SPI NOR flash as well as "splash.png" which is a sample image of what the splash screen will look like. Since ImageMagick is used to do the heavy lifting of the conversion process, the input file can be of nearly any image format.

The 'splash-convert' tool is available in the TS-7100 utilities (https://github.com/embeddedarm/ts7100-utils) repository.

The "splash.out" file can be written to the SPI NOR flash with the following command:

```
dd if=/path/to/splash.out of=/dev/mtdblock0 bs=4096 conv=sync
```

Note that the "bs" and "conv" arguments should always be specified when writing to this SPI NOR device with 'dd' to ensure that the eraseblocks do not receive unnecessary erases and that a full eraseblock is written every time.

Also note that on the TS-7100, the SPI NOR flash is 2 MiB but the splash screen only consumes 152 KiB of space. The rest of this flash space can be used for general storage if wanted.

10.14 LEDs

The red and green LEDs can be controlled from userspace after bootup using the sysfs LED interface. For example, to turn on the red LED:

```
echo 1 > /sys/class/leds/cpu-red-led/brightness
```

The following LEDs are available on this system:

- cpu-red-led

- cpu-green-led
- io-red-led
- io-green-led

A number of triggers are also available, including timers, disk activity, and heartbeat. These allow the LEDs to represent various system activities as they occur. See the kernel LED documentation (https://www.kernel.org/doc/Documentation/leds/leds-class.txt) for more information on triggers and general use of LED class devices.

10.15 Relays

The TS-7100 supports 2 relays on the #CN32 Terminal Block header. These are Emet EE2-5NU-L general purpose relays that will switch up to 2A at 220VDC or 250VAC.

These can be controlled with #GPIO.

```
# Connect Relay 1 NO to Common
gpioset 4 4=1

# Connect Relay 1 NC to Common (default power on state)
gpioset 4 4=0

# Connect Relay 2 NO to Common
gpioset 4 5=1

# Connect Relay 2 NC to Common (default power on state)
gpioset 4 5=0
```

10.16 Sleep

{{Notel This section is incomplete at this time.

10.16.1 Suspend-to-RAM

10.17 SPI

See the kernel spidev documentation (https://www.kernel.org/doc/Documentation/spi/spidev) for more information on interfacing with the SPI peripherals.

10.18 TS-SILO Supercapacitors

10.19 UARTs

The TS-7100-Z offers 4 total UARTs for communication:

UART Dev.	Type	TX / + Loc.	RX / - Loc.
ttymxc1	RS-232	CN32 Terminal, pin 1	CN32 Terminal, pin 7
ttymxc2	Bluetooth	N/A	N/A
ttymxc4	RS-232	CN32 Terminal, pin 3	CN32 Terminal, pin 5
ttyS0	RS-485	CN32 Terminal, pin 29	CN32 Terminal, pin 31

10.19.1 RS-485

RS-485 is implemented via a UART interface inside of the FPGA. This device handles automatic TXEN assertion and de-assertion for half-duplex RS-485 communication. See the UARTs section for the location of the RS-485 port.

10.20 USB Controller

The i.MX6ul provides 2 USB hosts supporting USB 2.0 (480Mbit/s). Typically this is interfaced with by using Linux drivers, but low level USB communication is possible using libusb (http://www.libusb.org/) .

10.21 Watchdog

The TS-7100 implements a WDT inside the supervisory microcontroller. A standard kernel WDT driver is in place that manages the WDT via the I2C bus. The WDT requires a userspace feeding system as the kernel has no provisions for self-feeding. Our stock distribution uses the 'watchdog' utility to check system health, set feed length, and perform feeds. Any arbitrary timeout value between 10 ms through 42949672.95 seconds is possible, with a resolution of 10 ms. Setting a timeout of 0 and issuing a feed will disable the WDT in hardware.

At power-on, the WDT in the microcontroller is live and running. This means that any failures in the boot process will cause a reboot and another attempt. From there, feeds must be manually done in U-Boot or the system booted to Linux within the default 60 second timeout. Once the kernel initializes the WDT driver, it will change the timeout and issue a single feed (default 5 minutes). Userspace must be fully booted and able to take over feeding within this time frame.

The default Linux timeout of 5 minutes was designed to accommodate very slow external I/O. For example, performing an 'fsck' on an external HDD of large sizes via USB mass storage interface, may take a number of minutes worst case. This timeout can be adjusted via the FDT file for the TS-7100. Setting this timeout to 0 will cause the WDT to be disabled in hardware as soon as the kernel is started.

The kernel driver supports the "Magic Close" feature of the WDT. This means that a 'V' character must be fed in to the watchdog file before the file is closed in order to disable the WDT. If this does not happen then the WDT is not stopped and it will continue it's countdown. This is the default behavior of our stock kernel.

Additionally, if the kernel is compiled with CONFIG_WATCHDOG_NOWAYOUT then the WDT can never be stopped once it is started at boot. This is not enabled by default in our stock kernel

See the Linux WDT API documentation (https://github.com/embeddedarm/linux-4.9.y/blob/master/Documentation/watchdog/watchdog-api.txt) for more information.

10.22 WiFi

This board uses an ATWILC3000-MR110CA (http://www.atmel.com/images/atmel-42569-atwilc3000-mr110ca-ieee802.11bgn-link-controller-with-integrated-bluetooth4.0_datasheet.pdf%7CAtmel) IEEE 802.11 b/g/n Link Controller Module With Integrated Bluetooth® 4.0. Linux provides support for this module using the wilc3000 driver.

Summary features:

- IEEE 802.11 b/g/n RF/PHY/MAC SOC
- IEEE 802.11 b/g/n (1x1) for up to 72 Mbps PHY rate
- Single spatial stream in 2.4GHz ISM band
- Integrated PA and T/R Switch Integrated Chip Antenna
- Superior Sensitivity and Range via advanced PHY signal processing
- Advanced Equalization and Channel Estimation
- Advanced Carrier and Timing Synchronization
- Wi-Fi Direct and Soft-AP support
- Supports IEEE 802.11 WEP, WPA, and WPA2 Security
- Supports China WAPI security
- Operating temperature range of -40°C to +85°C

11 Specifications

Note:	This section is incomplete at this time.

11.1 Power Specifications

The TS-7100-Z accepts a range of voltages from 8 V to 28 V DC.

11.2 Power Consumption

11.2.1 TS-SILO SuperCaps

12 External Interfaces

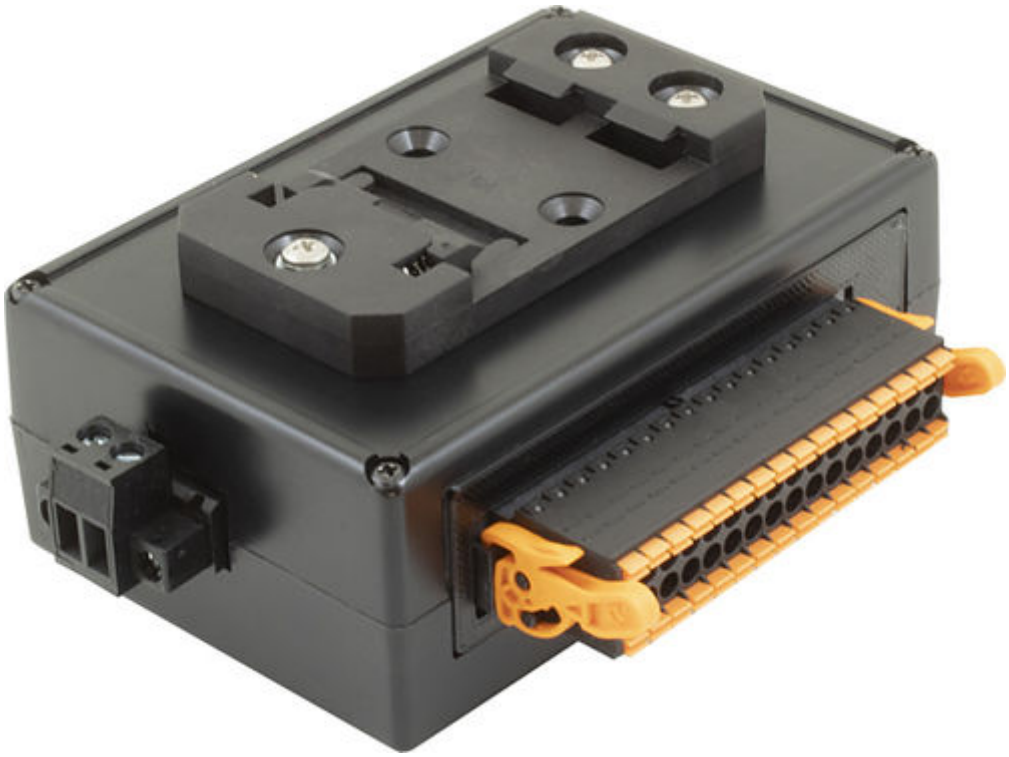
12.1 CN32 Terminal Block

The TS-7100 features a 2x16 IO header. The mating pluggable terminal block with spring clamps from On Shore (OSTNL321003) is included with the TS-7100. This connector supports AWG 24-16.

To plug in or unplug and individual pin, press and hold in the orange tab near the pin.

To remove the entire terminal block from the connector press the orange tabs on each side down. On plugging it back in the tabs will flip back up.

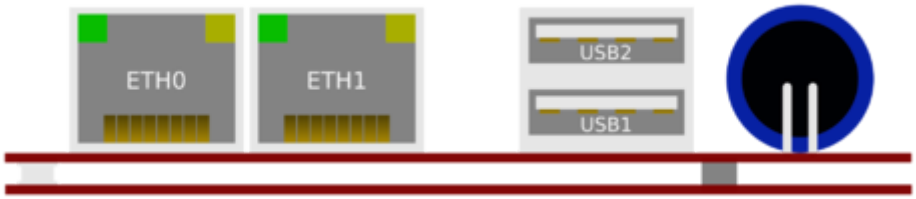

2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31



Pin	Description
1	ttymxc1 RS-232 TXD
2	Relay 1 Normally Open
3	ttymxc1 RS-232 RXD
4	Relay 1 Normally Closed
5	ttymxc4 RS-232 TXD
6	Relay 1 Common
7	ttymxc4 RS-232 RXD
8	Relay 2 Normally Closed
9	DIG_IN_1_STC GPIO Bank 5 IO 6
10	Relay 2 Normally Closed
11	DIG_IN_2_STC GPIO Bank 5 IO 7
12	Relay 2 Common
13	DIG_IN_3_STC GPIO Bank 5 IO 8
14	Output Bank 5 IO 0 or Input Bank 5 IO 2
15	CAN_L (can0)
16	Output Bank 5 IO 1 or Input Bank 5 IO 3
17	CAN_H (can0)
18	Output Bank 6 IO 13 or Input Bank 5 IO 4
19	AN_4_STC
20	GND
21	AN_3_STC
22	GND
23	AN_2_STC
24	GND
25	AN_1_STC
26	GND
27	AN_0 or High Side Switch Bank 5 IO 15
28	GND
29	ttyS0 RS-485+
30	GND
31	ttyS0 RS-485-
32	GND

12.2 Ethernet Ports

See the this section for further detail on networking.



12.3 CN16 XBee Socket

The XBEE header is intended for Digi XBEE modules, or Nimbelink's Skywire embedded modems (https://nimbelink.com/embedded-modems/) .

For XBEE modules, these will use 3.3V and require USB to be disabled:

```
# Enable 3.3V
gpioset 7 6=0

# Turn off USB
gpioset 7 9=0

# Access /dev/ttymxc3
```

Nimbelink modules require settings for the specific modems. Verify in the datasheet of the modem appropriate for your cell network to determine if the module uses USB. If not, verify the expected serial baud rate and settings. Some modems need 3.3V power, and some modems expect a typical of 4V.

```
# For modules wanting 4V:
gpioset 7 6=1

# For modules wanting 3.3V:
gpioset 7 6=0

# Power can be turned off if this GPIO is configured as an input
# gpioget 7 6

# If your modem supports USB, this must be enabled.
gpioset 7 9=1
```

Note: Some Nimbelink cell modems have a long startup and may not show up on USB for a minute before enumerating on USB.

Serial based modems can be tested with picocom

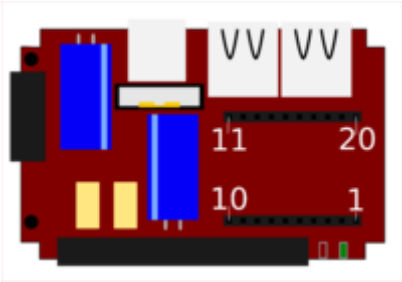
```
# Verify the baud rate for your specific modem
picocom -b 921600 /dev/ttymxc3
```

With picocom type "AT" and press enter and it should return "OK". Press Ctrl+a+x to close picocom.

Pin	Description
11	GND
12	/dev/ttymxc3 CTS
13	NC

Pin	Description
10	GND
9	GND
8	USB-

14	3.3V
15	GND
16	GND
17	NC
18	NC
19	NC
20	GND



7	USB+
6	4.7V
5	NC
4	GND
3	/dev/ttymxc3 TXD
2	/dev/ttymxc3 RXD
1	NIMBEL_PWR (3.3V or 4V)

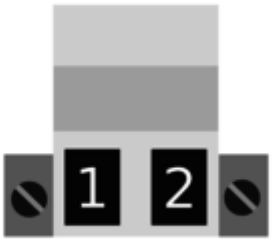
While pin 1 commonly lines up with the antennas on the modems, the pin 1 orientation should be verified in your modem's datasheet.

12.4 Power Terminal Block

The TS-7100 uses a 2 pin removable terminal block. The mating connector from On Shore Technologies OSTTJ025102 is included with the board.

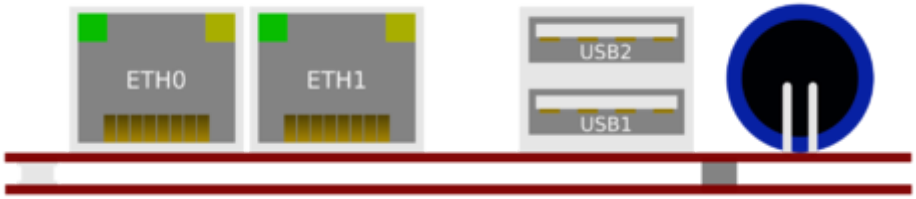
The 2 pin terminal block supports 8-48VDC input. A typical power supply will provide 1A at 12V. More details can be found in the #Power Consumption section.

Pin	Description
1 (Left)	8-48VDC
2 (Right)	GND



12.5 USB Ports

The TS-7100-Z offers two USB 2.0 type A host ports. See the USB Controller section for more details.



The bottom (USB1) port is always available on the USB Header. The top port (USB2) is switchable, and instead can be connected to the USB pins of an installed XBee device or NimbeLink modem inserted in to the CN16 socket. By default USB is connected to the top USB type A port instead of the Nimbelink Header.

```
# Enable top USB port (default):
gpioset 7 9=0

# Move USB to Nimbelink/XBEE Header
gpioset 7 9=1
```

Power to the host ports can be controlled with the LED subsystem under the LED device:

```
# USB off:
echo 0 > /sys/class/leds/en-usb-5v/brightness

# USB On:
echo 1 > /sys/class/leds/en-usb-5v/brightness
```

See the DIO section of the manual for more information on this. The USB A host port stack can provide 1 A total power output shared between the two ports.

13 Revisions and Changes

Note:	This section is incomplete at this time.

13.1 FPGA Changelog

13.2 Microcontroller Changelog

13.3 PCB Revisions

13.4 Software Images

13.4.1 Debian Changelog

13.5 U-Boot

14 Product Notes

14.1 FCC Advisory

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 digital device in accordance with the specifications in 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.

14.2 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 (https://www.embeddedarm.com/support/rma.php) to arrange for any repair service and to obtain repair charge information.

WARNING:	Writing ANY of the CPU's One-Time Programmable registers will immediately void ALL of our return policies and replacement warranties. This includes but is not limited to: the 45-day full money back evaluation period; any returns outside of the 45-day evaluation period; warranty returns within the 1 year warranty period that would require SBC replacement. Our 1 year limited warranty still applies, however it is at our discretion to decide if the SBC can be repaired, no warranty replacements will be provided if the OTP registers have been written.

WARNING:	Setting any of the eMMC's write-once registers (e.g. enabling enhanced area and/or write reliability) will immediately void ALL of our return policies and replacement warranties. This includes but is not limited to: the 45-day full money back evaluation period; any returns outside of the 45-day evaluation period; warranty returns within the 1 year warranty period that would require SBC replacement. Our 1 year limited warranty still applies, however it is at our discretion to decide if the SBC can be repaired, no warranty replacements will be provided if the OTP registers have been written.