

# HiCO.SH7780 / QNX 6.4.1

---

## Starterkit Manual

Rev3 / 04.02.2010

© Copyright 2010 **emtrion GmbH**

All rights reserved. This documentation may not be photocopied or recorded on any electronic media without written approval. The information contained in this documentation is subject to change without prior notice. We assume no liability for erroneous information or its consequences. Trademarks used from other companies refer exclusively to the products of those companies.

Revision: **3 / 04.02.2010**

Rev	Date/Signature	Changes
<b>2</b>	3.2.2010/ny	Added some diagrams
<b>3</b>	4.2.2010/ny	Added explanations for the Windows batch files

## Contents

1	Introduction.....	4
1.1	Prerequisites .....	4
1.2	Supported hardware interfaces .....	4
2	Getting started .....	5
2.1	Download and install the latest BSP .....	5
2.2	Build the BSP.....	5
2.3	Downloading and building on Linux.....	5
2.4	Establish a serial connection to the target .....	6
2.5	Download OS image to the target board.....	7
3	BSP files and directories .....	8
4	Flash partitioning.....	9
4.1	Emtrion bootloader.....	9
4.2	IFS area.....	9
4.3	EFS area .....	9
4.4	Using only IFS image .....	10
4.5	IFS and EFS(s-record) Image files.....	10
5	Getting your applications to the target.....	11
5.1	Add your application to the IFS image .....	11
5.2	Add your applications to the initial EFS image content .....	12
5.3	Download your application directly on the EFS file-system .....	13
5.4	Formatting EFS area on the target .....	13
6	System startup.....	14
6.1	Startup/boot configuration.....	14
6.1.1	Target Network configuration.....	14
6.2	Default network services.....	15
6.2.1	Telnet.....	15
6.2.2	FTP .....	15
6.2.3	Graphics with photon.....	15
7	Hardware interfaces.....	16
7.1	Real Time Clock.....	16
7.2	Touch Interface, Temperature and ANI1 (tsc2000).....	16
7.3	Graphics driver (devg-smi5xx.so).....	16
7.4	Ethernet (devn-smc9118.so).....	17
7.5	Serial ports (devc-ser8250, devc-sersci).....	17
7.6	Status LED.....	17

## 1 Introduction

This guide will help you get started with the QNX Neutrino 6.4.1 RTOS on your HiCO.SH7780 platform. The Starterkit includes ready built file-system images (photon, networking, etc.), device drivers to use the HiCO.SH7780 peripheral and documentation to get started.

### 1.1 Prerequisites

- Before you do anything else you should verify that the hardware works properly and the preinstalled OS image boots up.
- In order for the BSP to function properly you have to have QNX 6.4.1 development environment installed on your PC. If you don't already have a working QNX installation, you can get an evaluation version from address:  
<http://www.qnx.com/products/getmomentics/>
- free serial port on your development host (cable delivered in the Starterkit).
- free Ethernet (RJ-45) connection to your local network (cable delivered in the Starterkit). The boot loader uses DHCP to obtain an IP address.

### 1.2 Supported hardware interfaces

The BSP currently supports following interfaces on HiCO.SH7780:

- IPL and Startup code
- 10/100Mbit/s TCP/IP networking with Ethernet
- Read/write Flash file system with (`devf-generic`)
- 3 Serial ports
- Touch screen
- Graphics with 2D acceleration
- 1 Analog input and board Temperature
- Real Time Clock (RTC)

## 2 Getting started

### 2.1 Download and install the latest BSP

Download the latest BSP from address <http://www.support.emtrion.de/qnxbsp/>. Always take the latest release if you don't have any specific reasons to use an old one. After downloading, extract the BSP zip file into a working directory.

#### Source and Binary BSPs

The BSP you can download from above address contains the driver binaries and all the necessary configuration and build files to build custom images (ADK). The BSP is also available with source code (SDK). Please contact emtrion for more information.

### 2.2 Build the BSP

After unpacking the BSP into a working directory, you can simply double click on the **Make.bat** batch file.

If you have QNX 6.4.1 correctly installed on your system, the build should run through without any errors. And after a successful build you should find the ifs and efs srecord images (`hico7780-ipl-ifs.sre`, `hico7780-ipl-ifs-efs.sre`) in the `images` directory.

### 2.3 Downloading and building on Linux

Here are the shell commands to download install and build the latest BSP on Linux.

```
cd ~  
wget http://www.support.emtrion.de/qnxbsp/hico7780qnx641-latest.zip  
unzip hico7780qnx641-latest.zip  
cd hico7780qnx641-*/  
make
```

## 2.4 Establish a serial connection to the target

Use Hyperterminal, Teraterm, putty or any other serial terminal program with following parameters to establish RS232 connection to the target

Bps	115200
Databits	8
Parity	None
Stopbits	1
Flow control	None

In order to test the connection, set dip switch 4 to 'on' position. This will activate the bootloader. After resetting the board you should see the bootloader prompt on the terminal screen:

```
emtrion GmbH
Bootloader for HiCO.SH7780
Version 1.4 from $Date: 2008/04/01 10:48:38Z $
Copyright (c) 2000 - 2008
All rights reserved

Bootloader menu

1.) Execute stored Image
2.) Download Image via serial port and store in Flash
3.) Download Image via serial port, store in SDRAM and execute
4.) Download Image via LAN9218 Ethernet controller and store in Flash
5.) Download Image via LAN9218 Ethernet controller, store in SDRAM and
execute
6.) Extended functionality
7.) Download image over standard TFTP connection
```

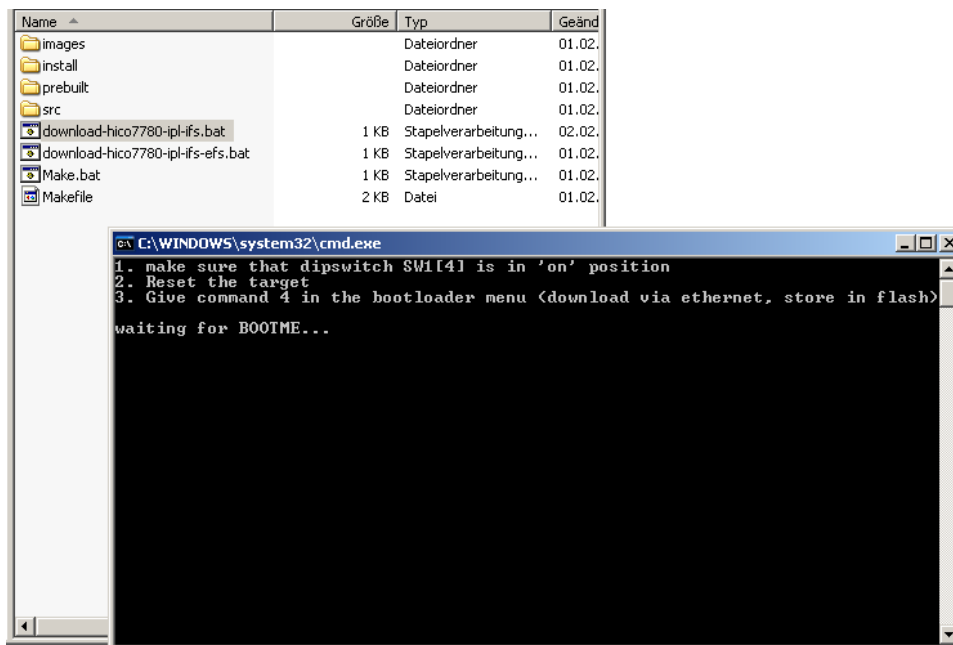
## 2.5 Download OS image to the target board

There are two batch files to do this for you in the top level BSP directory.

- `Download-hico7780-ipl-ifs.bat` – Download only the IFS image.
- `Download-hico7780-ipl-ifs-efs.bat` – Download both IFS and EFS images.

In order to download the image, proceed as follows:

1. Double clicking any of these batch files will start the download procedure on the development host.
2. Give command 4 (download via Ethernet and store in flash) in the bootloader Menu to start the download process.



**NOTE:** Check that you have the dip switch SW1[4] in 'on' position. This activates the bootloader menu on the rs232 serial port.

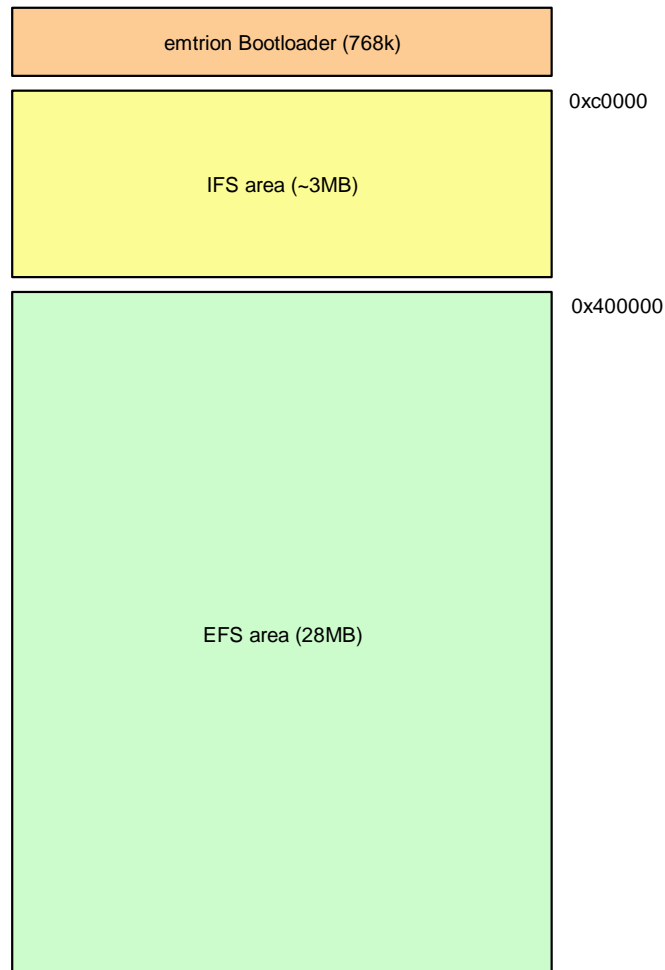
### 3 BSP files and directories

The HiCO.SH7760 QNX Neutrino 6.3 BSP follows the QNX BSP convention in the directory and Makefile structure with some exceptions. Here's a small description of the most important files and directories in the bsp:

File/Directory	Description
<code>Make.bat</code>	This batch file will build the BSP and the downloadable S-record images (windows only)
<code>download-hico7780-ipl-ifs.bat</code>	Batch file to download the IFS image to the target
<code>download-hico7780-ipl-ifs-efs.bat</code>	Batch file to download IFS and EFS image to the target
<b>Makefile</b>	Top level makefile for the BSP. It copies the prebuilt to install, builds all the sources in the <code>src</code> and installs the resulting binaries to <code>install</code> . It then runs the <code>images/Makefile</code> which builds the flash image. You only have to run this once if you're not doing changes in <code>src</code> .
<code>images/hico7780.build</code>	Example build file.
<code>images/hico7780-*.sre</code>	Prebuilt downloadable S-record files. Note that these images include the emptyefs partition (i.e. <code>/root</code> on the target is a read/write flash filesystem).
<code>images/mkflashimage</code> , <code>images/mkflashimage.bat</code>	Shell script to build flash images (see contents of <code>mkflashimage</code> for more information).
<code>install</code>	the build system uses this as a "scratch" directory. It is used as the primary search path when images are built. This directory also contains all prebuilt binaries
<code>prebuilt</code>	This directory contains all the prebuilt drivers, binaries and configuration files.
<code>src</code>	BSP source codes. The top level Makefile first builds all the sources located here and the installs them to the <code>install</code> directory. Note: In a binary release this directory doesn't exist. All the necessary drivers are already located in the <code>prebuilt</code> directory.

## 4 Flash partitioning

The 32MB NOR flash is divided into three parts; emtrion boot loader, IFS area and EFS area. Following illustrates the partitioning:



HiCO.SH7780 NOR Flash (32MB)

### 4.1 Emtrion bootloader

starts from address 0x0 and ends to address 0xbffff (768k). As it is possible to erase the bootloader - *You should avoid doing any flash operations on this area.*

### 4.2 IFS area

IFS image contains the QNX OS image created with `mkifs`. By default it starts from 0xc0000 and ends to address 0x400000 (4MB limit). This means that the IFS image can be a bit over 3MB of size. This should be enough since you would normally put your applications to the EFS area. If you wish, you can change the limit `hico778.build` by changing `EFS_START`

### 4.3 EFS area

is a read write filesystem, which by default contains the Photon installation and serves as a R/W storage for you applications.

Both **EFS** and **IFS** areas are mounted to the file-system root (/), where EFS file-system overlays the IFS system. That means that if you read a file, it might come from IFS or EFS depending whether the file exists in the EFS file-system. If you write a file it gets written to EFS, since IFS is read only.

#### 4.4 Using only IFS image

In some cases it is better to put all your applications to the IFS image and ignore any extra filesystems. You can achieve this, simply by setting EFS\_START value in hico7780.build at the end of flash (i.e. all the flash space is reserved for IFS)

#### 4.5 IFS and EFS(s-record) Image files

The build system delivered by emtrion builds two images.

- `hico7780-ipl-ifs.srec` image contains only the IFS filesystem
- `hico7780-ipl-ifs-efs.srec` image contains the IFS filesystem image and the initial content for the EFS filesystem (`<bsp>/images/efs/` directory).

`hico7780-ipl-ifs-efs.srec` is useful for production purposes, or whenever you want to have the entire flash content in one image file.

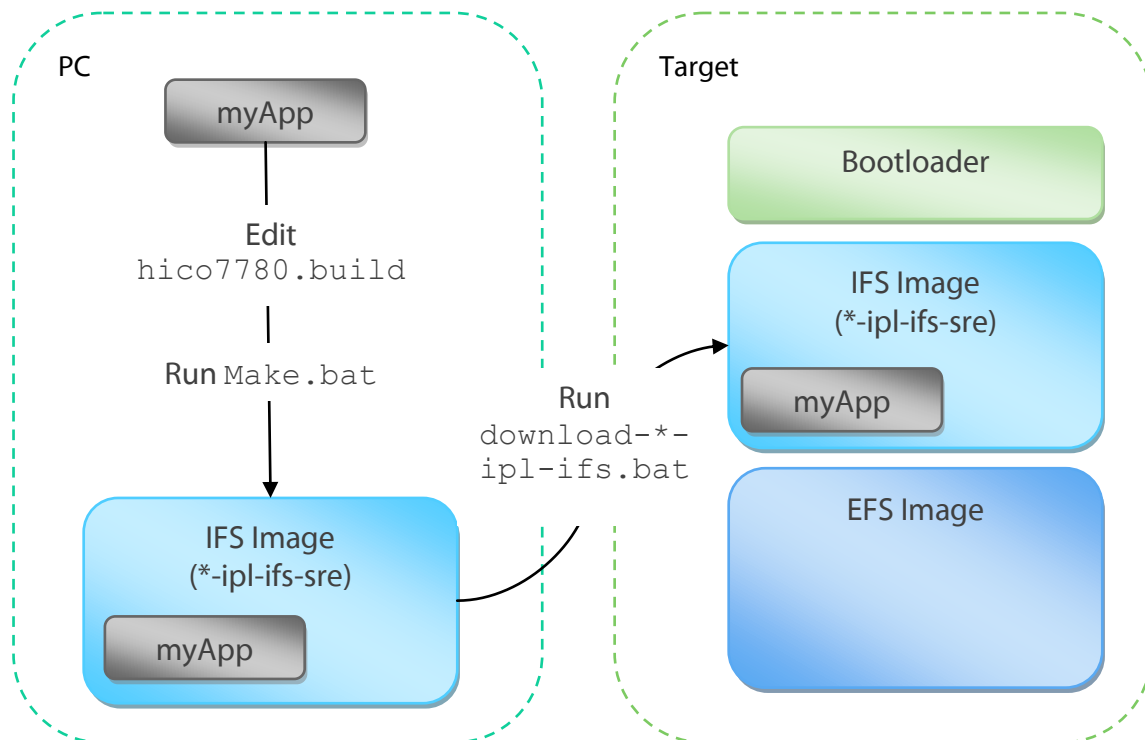
Note that the initial content in `hico7780-ipl-ifs-efs.srec` is *not* a file system composed with the `mkefs` utility, but an embedded `tar` archive. Content of this archive is copied to the EFS filesystem after running `/etc/system/efsinit`.

## 5 Getting your applications to the target

There are several ways how you can include you applications to the target system:

### 5.1 Add your application to the IFS image

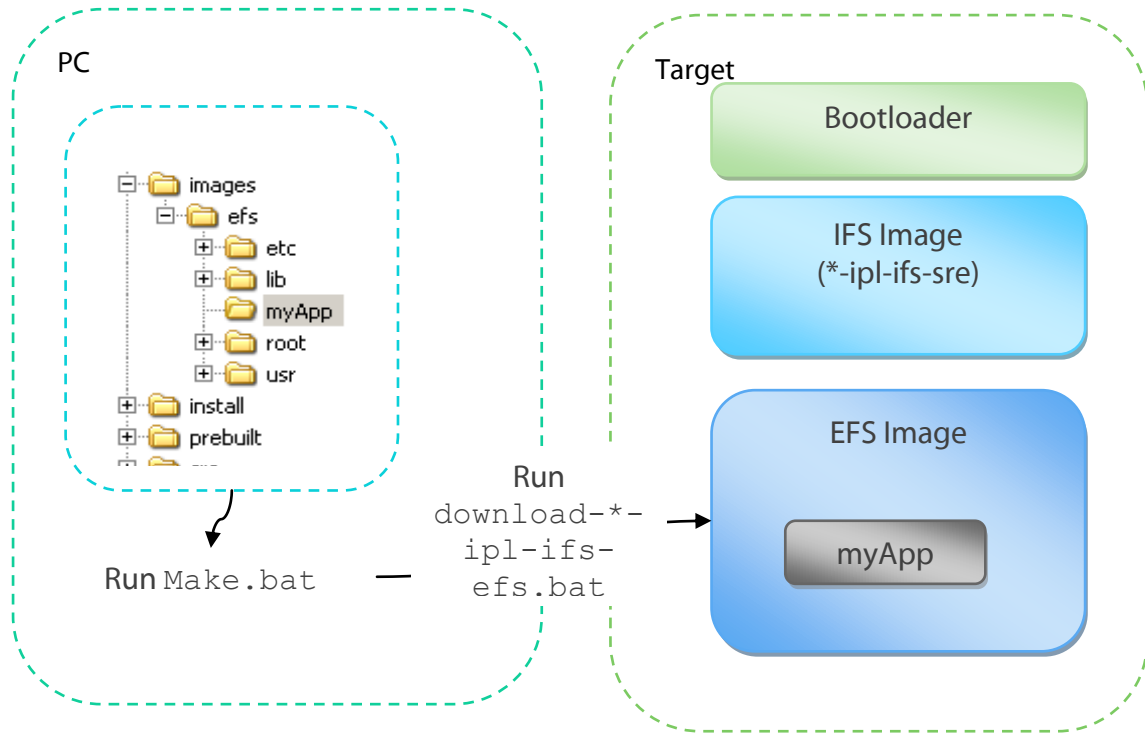
Configure the IFS image to include your applications by editing `hico7780.build`. See help page for `mkifs`, for how to edit the file.



Note, that if you use the IFS image for your applications, you always need to update the whole IFS image if you want to update your application. Recommended way is to use the EFS area for your applications.

## 5.2 Add your applications to the initial EFS image content

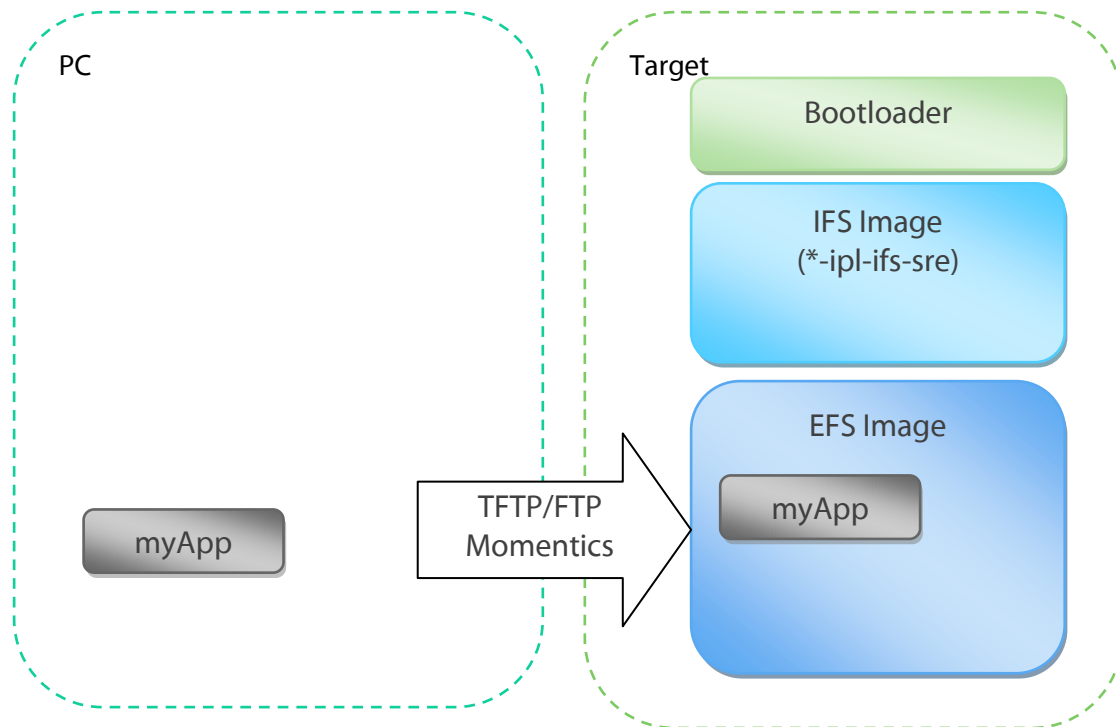
Simply copy your files to the directory pointed by `EFS_HOSTPATH` in `hico7780.build`. By default this directory is `images/efs`.



This directory is added to the `hico7780-ipl-ifs-efs.srec`, which can be downloaded to the target with the emtrion's KernelDnld download utility. This is of course not the preferred way in the development phase, since downloading the whole `efs+ifs` image takes a long time. This method is mainly meant to be used for production purposes.

### 5.3 Download your application directly on the EFS file-system

After downloading the first IFS image (`hico7780-ipl-ifs.srec`), you can use `/etc/system/efsinit` command to format and initialize the EFS file-system. The EFS file-system 'overlays' the IFS file-system so you can just copy your application files and binaries somewhere (for example `/user/myapp/`), and they will land on the EFS file-system.



NOTE: You don't necessarily need to have anything in the initial EFS file-system. You can always use `/etc/system/efsinit` on the target to create an empty EFS file-system and copy all your files there via ftp/ftp Momentics IDE, etc.

### 5.4 Formatting EFS area on the target

It is not necessary to download the `hico7780-ipl-ifs-efs.srec` in order to have a functioning EFS file-system. You can download only the IFS image and then initialize, format and mount an empty EFS partition.

In order to initialize (erase and format it) EFS area you can run command `/etc/system/efsinit`. The script erases and formats the EFS are.

## 6 System startup

Following list outlines the most important events after the board is powered:

1. Emtrion bootloader starts and after initializing the basic hardware, jumps to address `0xc0000` where the QNX `ipl` (Initial Program Loader) is located.
2. QNX `ipl` copies the IFS image to RAM and then jumps to the QNX start-up code.
3. QNX start-up code does further hardware initialisation and fills the system page with hardware information.
4. The kernel (`procnto`) invokes the script section in `hico7780.build`. Normally some basic services and drivers are started here, but `hico7780.bsp`, the script executes immediately shell script `/etc/system/sysinit`.
5. `/etc/system/sysinit` is the main configuration script. It starts the hardware drivers, mounts the EFS filesystem, initializes networking, and starts photon (if found on the system). For a more detailed overview – please refer to the script itself.
6. As a last task `/etc/system/sysinit` starts a login console. At this point everything should be up and running.

### 6.1 Startup/boot configuration

#### 6.1.1 Target Network configuration

You can assign the target a static IP address or configure the network via DHCP. You set this in the `hico7780.build`:

```
# Network configuration IPADDR=none|dhcp|<ipaddr>
IPADDR=dhcp
NETMASK=255.255.255.0
ETHADDR=001122334455
```

For any more complicated network configuration, you need set `IPADDR` to `none` and configure manually – or, change the corresponding parts in `/etc/system/sysinit`.

Default value for `IPADDR` is `dhcp`.

## 6.2 Default network services

### 6.2.1 Telnet

Telnet service on the target is enabled by default and you can connect with any telnet client. Username is 'root' and password is 'root'. Telnet can be disabled in `/etc/inetd.conf` by commenting out the corresponding line.

*Note that telnet is a very insecure service – be sure to disable it in a production system!*

### 6.2.2 FTP

FTP is enabled by default, and you can copy files from and to the target with a ftp client. Username and password are both 'root' and FTP directory on the target is `/root`. FTP can be disabled in `/etc/inetd.conf` by commenting out the corresponding line.

*Note that FTP is a very insecure service – be sure to disable it in a production system!*

### 6.2.3 Graphics with photon

If file `/usr/photon/bin/photon` (photon startup script), photon is started automatically in the `/etc/system/sysinit`.

The default photon environment is simple desktop environment. The photon binaries are located in the `efs` filesystem.

## 7 Hardware interfaces

### 7.1 Real Time Clock

Value of the SH7780 internal RTC time is set as system time in the `/etc/system/sysinit` script. The `hico7780` utility has a command line options to set and read the time in the battery buffered RTC. When writing time to RTC the time string needs to be in format `YYYYMMDDhhmm.ss`. Here are some examples:

```
# Set system time/date from the battery buffered RTC
date `hico7780 rtc-read`

# Save current system time to the RTC
hico7780 rtc-write `date +%Y%M%d%H%M.%S`
```

### 7.2 Touch Interface, Temperature and ANI1 (tsc2000)

HiCO.SH7780 has a resistive 4 wire touch interface controlled by the onboard TSC2000 chip. The chip also provides a temperature value and an analogue input which is connected to the ANI1. Driver `tsc2000` provides an access to all of these interfaces.

The driver is started by default in the `/etc/system/sysinit` script. After starting the driver you can find following new device nodes:

- `/dev/touch` – Data from this device node is compatible to the `devi-em Photon` input driver. Normally you don't need to read any data directly from this node.
- `/dev/ani1` – from here you can read an integer value which represents the `ani1` voltage level:

```
int val;
fd=open("/dev/ani1",O_RDONLY);
read(fd,&val,sizeof(int);
```

- `/dev/temp` – From here you can read the current `tsc2000` chip temperature, which somewhat corresponds the board/environment temperature, as an integer value (see the `ani1` example above).

### 7.3 Graphics driver (devg-smi5xx.so)

The SM502 companion chip provides graphics with 2D acceleration. The driver is implemented as a photon DLL and should not require any direct commands from user. The graphics driver is loaded in the `photon` startup script `/usr/photon/bin/photon`.

Graphics resolution and other parameters can be set in configuration file `usr/photon/config/smi5xx.conf`. Note, however, that you will need to study the SM502 datasheet in order to do non-trivial changes in this file.

## 7.4 Ethernet (devn-smc9118.so)

The Ethernet driver `devn-smc9118.so` is implemented as a `io-pkt` compatible DLL and it gets loaded together with the `tcp/ip` stack. The driver is started in `/etc/sys/sysinit`.

## 7.5 Serial ports (devc-ser8250, devc-sersci)

HiCO.SH7780 has 3 serial ports which in the hardware documentation are referred as UART-[A:C]. Uarts A and B are located on the sm502 companion chip, while uart-C is the SCIF0 interface on the SH7780 processor.

Two drivers are needed to use the serial ports; `devc-ser8250` and `devc-sersci`. Both are started in the system initialisation file `/etc/sys/sysinit`. By QNX convention the serial port device nodes are named as `ser[0-9]`. Following symbolic links were added to ease the use:

```
# ls -l /dev/uart[A-C]
lrwxrwxrwx 1 root root 9 Dec 14 15:22 /dev/uartA -> /dev/ser3
lrwxrwxrwx 1 root root 9 Dec 14 15:22 /dev/uartB -> /dev/ser4
lrwxrwxrwx 1 root root 9 Dec 14 15:22 /dev/uartC -> /dev/ser1
```

## 7.6 Status LED

The three color on board LED, (red, green, combined yellow) status leds indicates CPU usage during normal operation. The LED blinks green with one second interval. The on-time between the interval indicates the CPU usage. Continuous green light means 100% CPU load. Currently, the red light is only being used to indicate the boot process.

The status LED driver is implemented in the `hico7780` utility and it is started in the early phases of `/etc/system/sysinit` startup script.