Xinu on Intel Galileo

User Manual

Table of Contents **Page**

1.0 Introduction

Important note: this document uses the first generation of the Galileo. Intel has introduced a second generation Galileo; we will update the document once Gen2 boards are available.

To boot the Xinu operating system on a Galileo board requires upgrading the firmware, connecting a serial cable that is used as the system console, compiling Xinu, and downloading the Xinu image onto the Galileo. There are two ways to download an image: place the image on an SD card and plug the SD card into the Galileo, or connect the Galileo to a network, configure a server, and arrange for the Galileo to download and image over the network. Using an SD card means that no network configuration is needed, but a network connection means a programmer can download-test-modify-and-download quickly.

The following describes each step of the process, including instructions for the use of and SD card as well as instructions on how to set up bootstrap over a network.

Firmware Update for the Intel Galileo board

The Galileo firmware must be upgraded to the latest update in order to be able to run Xinu. More details on how to upgrade the firmware can be found on: <intel galileo upgrade url>. It is a good idea to run the latest firmware, even if one is not running Xinu.

2.0 Console connection on the Intel Galileo

2.1 Background

Every computer has an interface that allows interaction with an operating system. The Galileo provides a serial console that allows a user to keystrokes to the computer and display output from the board. Typically a System-on-Chip uses a serial device known as a Universal Asynchronous Receiver Transmitter (UART). UART is used to connect a SoC to an RS-232 serial interface. Originally, RS-232 was used to communicate with a terminal (a device with a screen and keyboard).

Modern computing systems do not use terminals. Instead, we use software running on a conventional computer (e.g., a laptop) to emulate a terminal. We will assume a laptop throughout the description, but a desktop system can be used instead. The serial interface from a Galileo is connected to the laptop, and a terminal emulation program runs on laptop. The terminal emulation program displays all the characters the board emits (characters that arrive from the board), and sends all keystrokes the user enters to the board.

2.2 Serial cable setup for the Intel Galileo

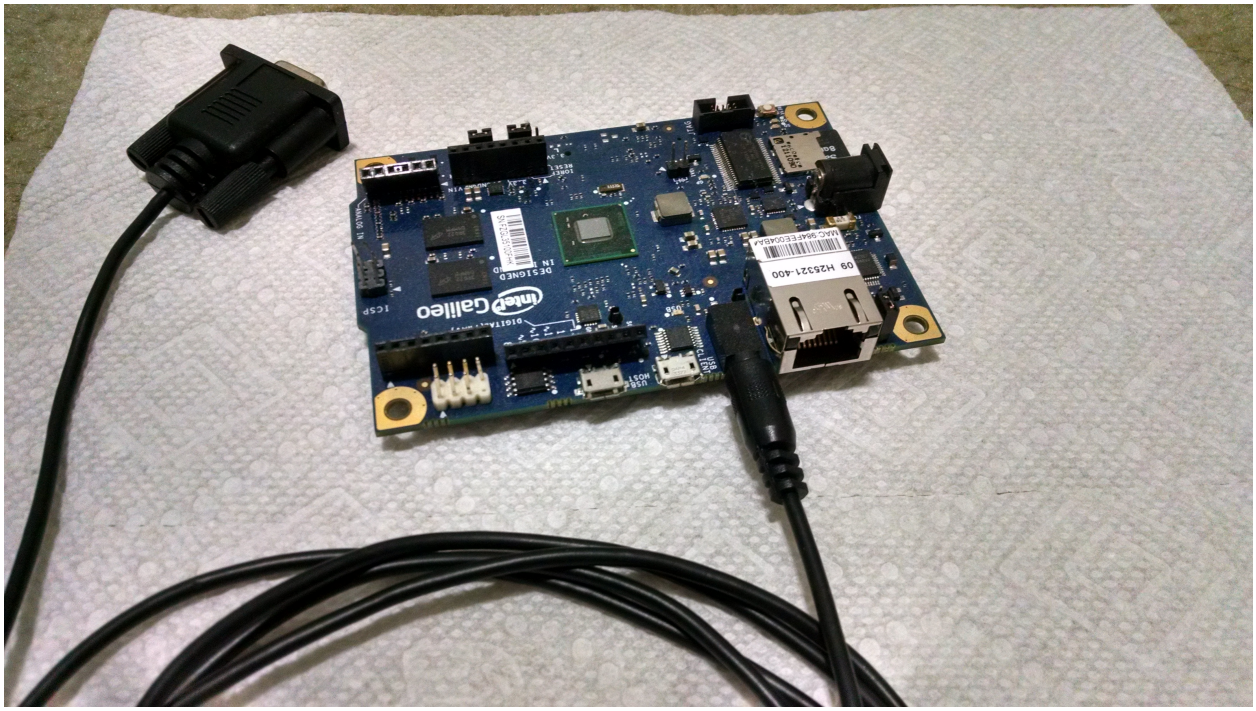
The Galileo board has a 3.5 mm connector that connects to a serial cable. Because laptops do not have serial ports, two cables are used to connect the Galileo to a laptop — one that connects from the Galileo to a standard DB-9 serial connector, and a second that connects a DB-9 connector to a USB connector.

1. 3.5 mm to DB-9 serial cable (Amazon link)
2. DB-9 to USB cable (Amazon link). Note: this cable is not necessary if the laptop has a DB-9 port on it, in which case cable #1 can be plugged into the DB-9 port on the laptop directly.

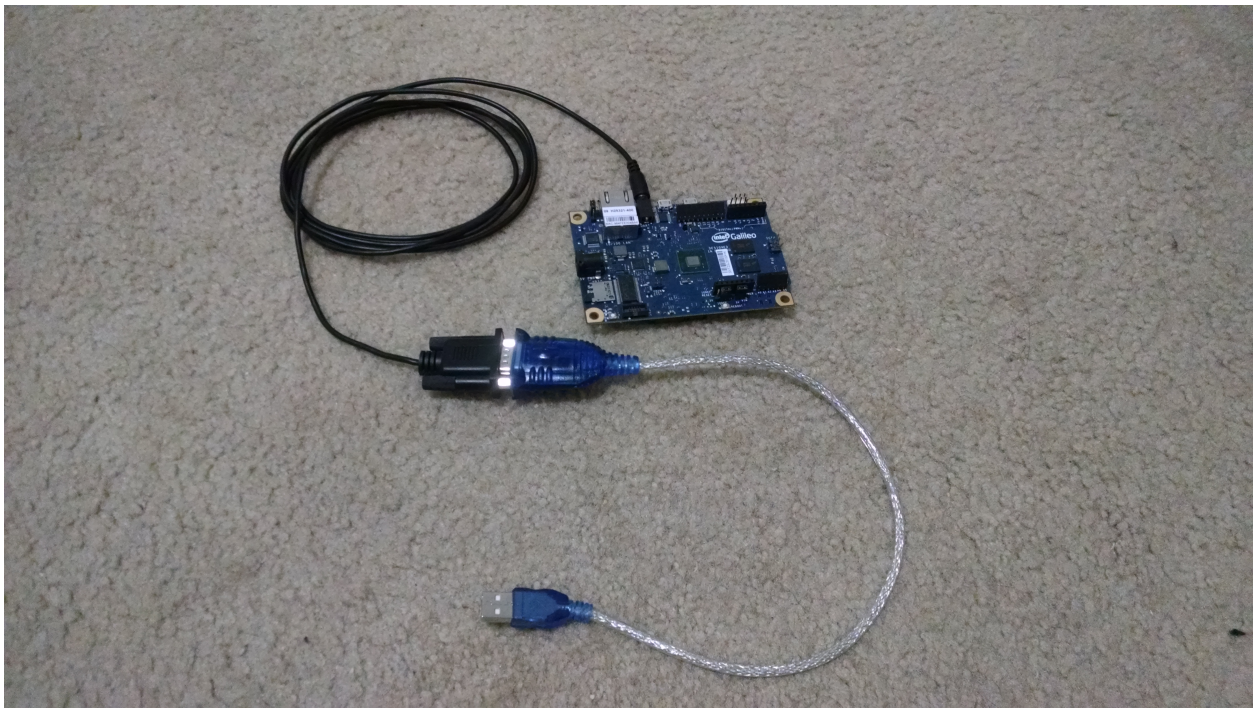
Serial Cable (3.5 mm to DB-9)

Serial cable (DB-9 to USB)

Connecting cable 1 to the Galileo



Complete Serial cable connection to the Galileo



2.3 Terminal emulator software configuration

Various terminal emulation programs are available, and any of them will suffice. We will use PuTTY in our examples because it runs on Linux and Windows and has been tested with the Intel Galileo board. The software can be downloaded from http://www.putty.org/

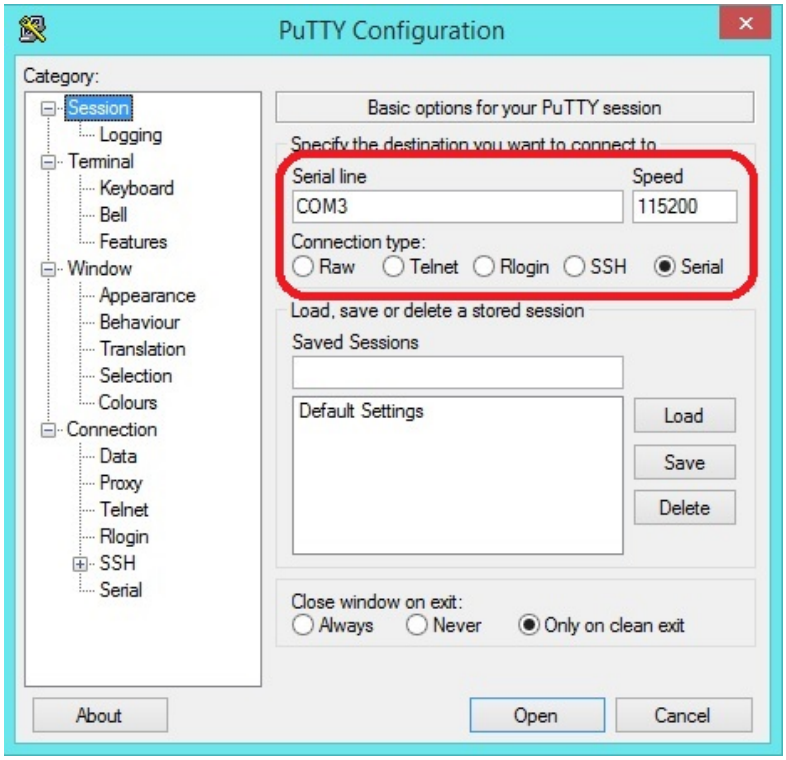
When the Galileo board is connected to a laptop using the two cables described above, a device is created (e.g. /dev/ttyUSB0 in Linux or COM3 in Windows, but the actual devices may differ on your laptop). PuTTY uses the device (port in case of Windows) to communicate with the Galileo.

PuTTY configuration:

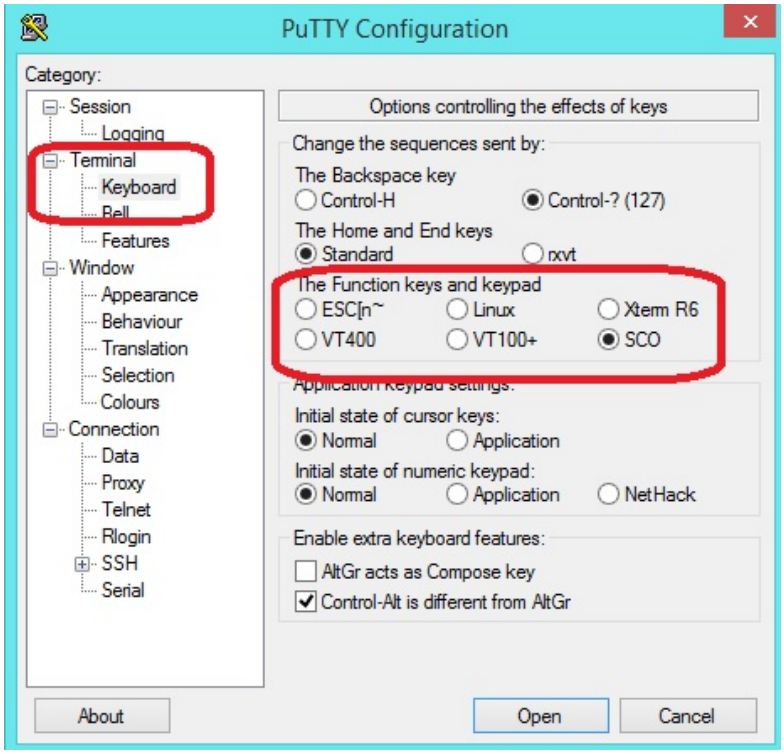
1. Launch PuTTY. In the main "Session" pane, select *Serial*
2. In the *Serial line* field enter the device name (e.g. /dev/ttyUSB0 in Linux or COM3 in Windows)
3. In the *Speed* field enter 115200
4. In the left pane, under *Terminal*, click on *Keyboard*
5. Under the heading *The Function keys and keypad*, select *SCO*
6. Click *Open*
7. A window for I/O will appear
8. Reboot the Galileo board

After the above steps are performed, you will see the debug output that is printed on the screen when the Galileo board reboots.

Putty Serial line connection and Baud rate



Putty keyboard settings



3.0 Downloading and Compiling Xinu

3.1 Background

Xinu is a small and elegant Operating System designed by Prof. Douglas Comer at the Purdue University. Except for a few low-level pieces, such as a context switch and interrupt dispatching code, Xinu is written in C and implements all the fundamental Operating System features, including concurrent process execution, process synchronization, inter-process communication, and an IPv4 network stack with UDP, ICMP and TCP. To get information about the history of Xinu visit http://www.xinu.cs.purdue.edu,

Tools needed to compile Xinu

1. A Linux machine (e.g., a laptop)
2. GCC compiler that is capable of compiling for the Intel x86 architecture (If the Linux machine that is used for compiling is an x86 or x86-64 machine, the default gcc will be able to compile for x86 architecture)
3. Flex lexical analyzer
4. Bison parser
5. Make utility

3.2 Downloading Xinu source code and compiling Xinu

1. Download the tarball http://www.xinu.cs.purdue.edu/files/Xinu-code-Galileo.tar.gz
2. Untar the tarball on a Linux machine
3. Change to the compile/ directory
4. From the shell run the following:

```
$ make clean
$ make
```

After following the steps above, in the compile directory two files will be generated: *xinu.elf* and *xinu*. The file *xinu.elf* must be used when running Xinu from the SD card and the file *xinu* must be used when using Network boot to run Xinu.

4.0 Running Xinu from the micro-SD card

4.1 Background

When the Galileo boots, the hardware runs BIOS code. After it does some preliminary initialization, the BIOS hands over control to a boot loader named GRUB. GRUB chooses an Operating System image to boot, and starts the Operating System boot process. GRUB can be configured to select an image automatically or allow a user to select an image.

The Intel Galileo boards come with a modified version of GRUB 0.99 already installed in their flash memory. The modified GRUB cannot perform all steps needed to boot Xinu. Instead, the modified GRUB only handles two possibilities: booting Linux or chain-loading another boot-loader. Xinu is built to be multiboot compliant, but unfortunately the modified GRUB 0.99 on the Galileo does not support booting a multiboot kernel. Therefore, we need a more powerful version of GRUB to boot XInu.

The idea is straightforward: add a more powerful GRUB to the SD card. Use the built-in version of GRUB to run the more powerful version of GRUB, and use the more powerful version of GRUB to boot Xinu. Fortunately, the more powerful version of GRUB can remain on the SD card, and the GRUB configurations can be permanently stored. As a result, GRUB only needs to be set up once and then it can be used to boot Xinu again and again.

4.2 Micro-SD card setup

1. Format the micro-SD card to a FAT-32 file system
2. Download the tarball http://www.xinu.cs.purdue.edu/files/xinu-galileo-sdcard.tar.gz and untar it onto the micro-SD card (doing so loads the more powerful version of GRUB and the configuration file)
3. Copy the compiled Xinu image (xinu.elf) to the micro-SD card
4. Insert the micro-SD card in the Galileo board
5. Connect the board to a computer running a terminal emulator (PuTTY) using the procedure given above
6. Reboot the Galileo board

Whenever Xinu is re-built, the image on the micro-SD card must be replaced by the new image. To avoid repeatedly removing the SD card from the Galileo, inserting the card in the laptop, copying a new image to the card, one can , set up a network boot as described in the next section.

5.0 Loading Xinu over the network using Network Boot

5.1 Background

Network boot is the process by which a computer downloads an Operating System image over the network and boots it. Typically the BIOS in a computer supports Network boot using the Preboot eXecution Environment (PXE).

How Network boot works:

1. The Network boot client sends a DHCP request to obtain an IP address and the IP address of the server that hosts the OS image
2. The DHCP server responds with an IP address for the Network boot client, the IP address of the TFTP server, and the name of a file containing an operating system image
3. The Network boot client downloads the OS image from the TFTP server and boots it.

Boot loaders, such as GRUB, support Network boot provided they have drivers for the on-board Ethernet device. Unfortunately, GRUB does not support the Ethernet device that is present on the Intel Galileo boards. So, the Xinu Galileo team designed a Xinu-based boot loader called Xboot. Xboot downloads a Xinu image over the network and starts running Xinu.

5.2 Micro-SD card setup for network bootstrap

1. Format the micro-SD card to a FAT-32 filesystem
2. Download the tarball http://www.xinu.cs.purdue.edu/files/xinu-galileo-sdcard-nb.tar.gz and untar it in the micro-SD card
3. Insert the micro-SD card in the Galileo board and reboot the board

5.3 DHCP and TFTP server setup

You will need to run a DHCP server and a TFTP server on a computer that is connected to the same network as a Galileo board .

The DHCP server must be configured to assign a client IP address to the Galileo board, and must also return the address of a TFTP server and the name of the file that will contain a Xinu image.

The DHCP server configuration might look like this:

subnet x.x.x.x netmask m.m.m.m {

option routers <Router IP address [optional]>

option subnet-mask <Subnet mask>

next-server <TFTP Server address>

filename "<XINU Image name>"

range <IP address range low> <IP address range high>

}

where x.x.x.x is the dotted decimal for the IP address and m.m.m.m is the dotted decimal for the address mask used on your local network. The TFTP server, the XINU image must be placed in the /tftpboot directory [or the corresponding root directory for your TFTP server]. The Xinu Network boot image is named "xinu".

Every time Xinu is modified, the new Xinu image (xinu) must be placed in the TFTP root directory and the Galileo board must be rebooted to fetch the new image.