Shikhara Board Bring up and Validation

Contents

[1. References 4](#_Toc475962569)

[2. Shikhara – Initial Board Testing 5](#_Toc475962570)

[2.1. Test Setup / Connecting the DSTREAM unit 5](#_Toc475962571)

[3. ARM DS-5 Development Studio 7](#_Toc475962572)

[3.1. Creating a new platform configuration 7](#_Toc475962573)

[3.2. Core Connection Establishment 9](#_Toc475962574)

[3.3. Creating Project in Eclipse 10](#_Toc475962575)

[4. Bare Metal Code 14](#_Toc475962576)

[4.1. Linker Script 14](#_Toc475962577)

[4.2. Bare Metal – Sample Examples 14](#_Toc475962578)

[4.3. Compilation 15](#_Toc475962579)

[5. Bare Metal Code –LEDS Blink Test 16](#_Toc475962580)

[5.1. Test Setup / Board Connections 16](#_Toc475962581)

[5.2. Test Procedure 16](#_Toc475962582)

[6. Bare Metal Code – UART Hello World Test 17](#_Toc475962583)

[6.1. Test Setup / Board Connections 17](#_Toc475962584)

[6.2. Test Procedure 17](#_Toc475962585)

[7. Bare Metal Code – SMC NOR Test 18](#_Toc475962586)

[7.1. Test Procedure 18](#_Toc475962587)

[8. Flash / Program / fuse BL1 into NOR 19](#_Toc475962588)

[8.1. Test Procedure 19](#_Toc475962589)

[8.2. File Transfer from Host PC 19](#_Toc475962590)

[9. Flash / Program / fuse BL1 into SPI Flash 22](#_Toc475962591)

[9.1. Test Procedure 22](#_Toc475962592)

[9.2. File Transfer from Host PC 22](#_Toc475962593)

[10. Flash / Program / fuse BL1 into SD 25](#_Toc475962594)

[10.1. Steps to format SD card 25](#_Toc475962595)

[10.2 Copying Images to SD 27](#_Toc475962596)

[11. Flash / Program / fuse BL1 into NAND 28](#_Toc475962597)

[11.1 Test Procedure 28](#_Toc475962598)

# 1. References

**Shikhara\_SoC\_Architecture.doc** [Contain the details about memory map of all the controllers]

smb://192.168.200.242/kdocs/Knowledge\_Docs/Projects\_Based/Shikhara\_ANUSoC/Project\_docs/ANUSOC\_HW\_ARCH/APRIL\_30\_2015/doc/arch\_doc/Shikhara\_SoC\_Architecture.doc

**shikhara\_soc\_22022016.pdf** [Contain the details of Board Schematic details]

smb://192.168.200.242/kdocs/Knowledge\_Docs/Projects\_Based/Shikhara\_ANUSoC/BareCodeTest\_Shikhara/shikhara\_soc\_22022016.pdf

**Shikhara\_Tapeout\_Signoff.docx** [Contain the details of Boot strap modes]

smb://192.168.200.242/kdocs/Knowledge\_Docs/Projects\_Based/Shikhara\_ANUSoC/BareCodeTest\_Shikhara/Shikhara\_Tapeout\_Signoff.docx

**ShikharaSoC\_FW\_Micro\_Architecture.pdf [**Contain the details about boot sequence and Procedure]

smb://192.168.200.242/kdocs/Knowledge\_Docs/Projects\_Based/Shikhara\_ANUSoC/Project\_docs/ANUS0C\_FW\_ARCH/ShikharaSoC\_FW\_Micro\_Architecture.pdf

**IP Controller Datasheets** [Data sheet of all the IP Controllers]

smb://192.168.200.242/kdocs/Knowledge\_Docs/Projects\_Based/Shikhara\_ANUSoC/Project\_docs/ANUSOC\_HW\_ARCH/June\_2014/fpga/DATASHEETS/

**CoreSight.pdf [**CoreSight Document]

smb://192.168.200.242/kdocs/Knowledge\_Docs/Projects\_Based/Shikhara\_ANUSoC/BareCodeTest\_Shikhara/CoreSight.pdf

**Debug Using D-Stream and DS-5.pdf** [Debug with DS-5]

smb://192.168.200.242/kdocs/Knowledge\_Docs/Projects\_Based/Shikhara\_ANUSoC/BareCodeTest\_Shikhara/Debug Using D-Stream and DS-5.pdf

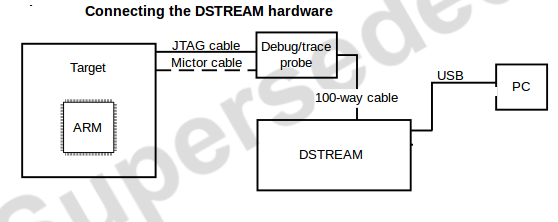
# 2. Shikhara – Initial Board Testing

## 2.1. Test Setup / Connecting the DSTREAM unit

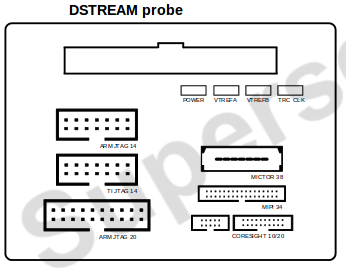
To connect the DSTREAM unit to host computer and to the target hardware, carry out the following:

1. Have the Host PC running Ubuntu 14.04 LTS [64-bit], with DS-5 installed, having the trail 60 day license
2. Connect the host computer to the DSTREAM unit as shown in the following figure, using the USB port
   * If you are connecting using the USB port, connect one end of the supplied USB cable to a USB port on the host computer, and the other end of the cable to the USB port on the DSTREAM unit.

**Note:** The USB drivers are installed with the debug host software.

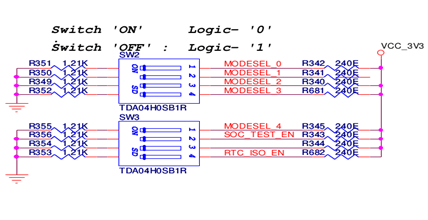


1. Connect the DSTREAM unit to the target hardware, using the appropriate debug or trace cables:
   * Connect one end of the supplied 100-way cable to the DSTREAM unit, and connect the other end of the cable to the probe unit.
   * Connect the target hardware to the probe using the appropriate cables and connector.
   * ARM JTAG 20 ,This is the most commonly-used debug connector standard for ARM-based target boards



1. Select Mode select pins to point “**TRACE MODE**” to Enter JTAG Mode

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Mode Selection** | SOC\_TEST\_EN | MODE\_SEL\_4 | MODE\_SEL\_3 | MODE\_SEL\_2 | MODE\_SEL\_1 | MODE\_SEL\_0 |
| NORMAL\_MODE\_0 | 0 | 0 | x | x | 0 | 0 |
| NORMAL\_MODE\_1 | 0 | 0 | x | x | 0 | 1 |
| NORMAL\_MODE\_2 | 0 | 0 | x | x | 1 | 0 |
| NORMAL\_MODE\_3 | 0 | 0 | x | x | 1 | 1 |
| TB\_MODE | 0 | 1 | x | 0 | 0 | 0 |
| TRACE\_MODE | 0 | 1 | x | 0 | 0 | 1 |



1. Power-up the target hardware [Connect 12V DC Input] and power-up the DSTREAM unit.
2. Typical Test Setup looks like this

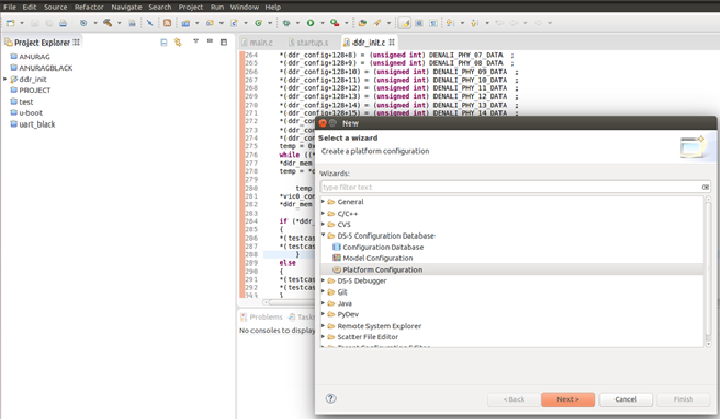


# 3. ARM DS-5 Development Studio

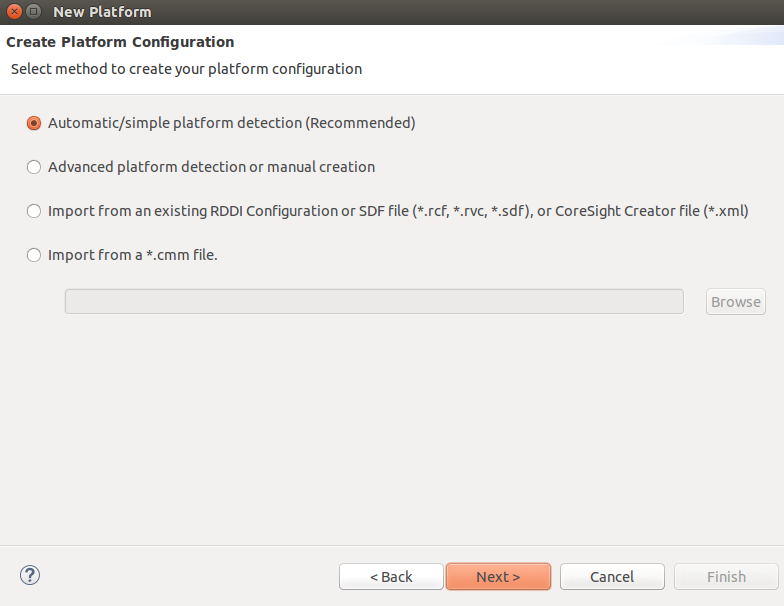
## 3.1. Creating a new platform configuration

You can use the Platform Configuration Editor (PCE) within DS-5 to create debug configurations for new platforms. Creating a new platform configuration in DS-5 requires a new Platform Configuration project in Eclipse

1. From the main menu in DS-5, select **File > New > Other** to open the new project dialog.
2. Select **DS-5 Configuration Database > Platform Configuration** and then click **Next**.

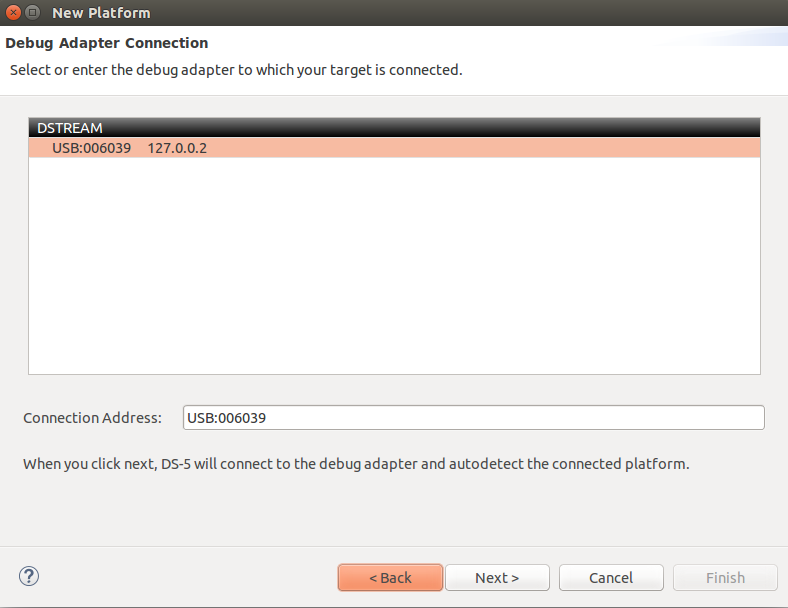


1. This shows the **Create Platform Configuration dialog box**. Select the required method to create the configuration for your platform and click **Next**.
2. Select **DS-5 Configuration Database > Platform Configuration** and then click **Next**



1. Automatic/simple platform detection

This is the recommended option. DS-5 automatically detects the devices that are present on your platform. It then provides you the opportunity to add more devices if needed and to specify how the devices are interconnected.



1. Once we have selected the DSTREAM unit, DS-5 will connect to the SoC and try to read all the information it needs for debug and trace

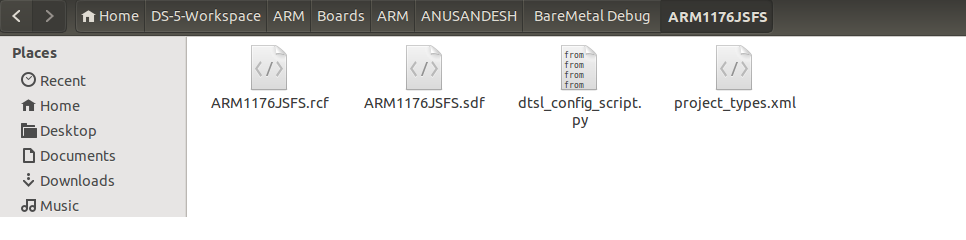
Note: **Manual Platform Configuration**

We can manually create the platform configuration if DS-5 does not automatically detect the platform configuration. DS-5 uses all the information that it reads from the platform to create a custom platform configuration

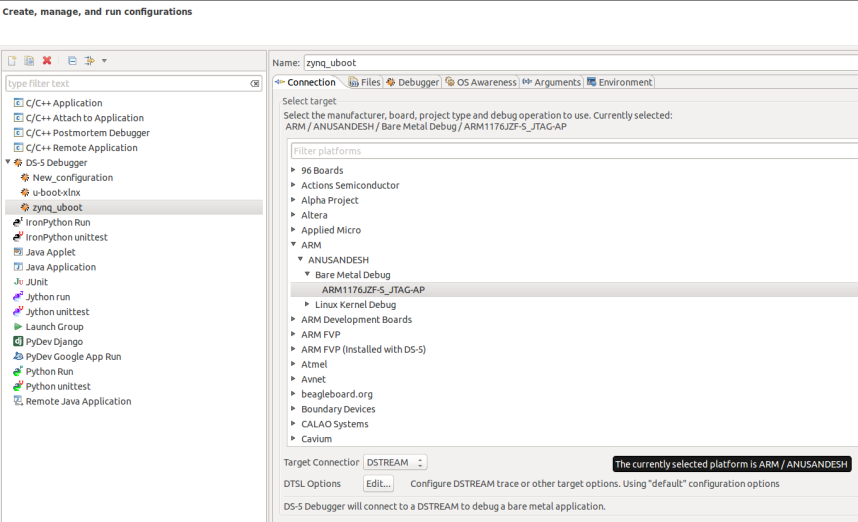
## 3.2. Core Connection Establishment

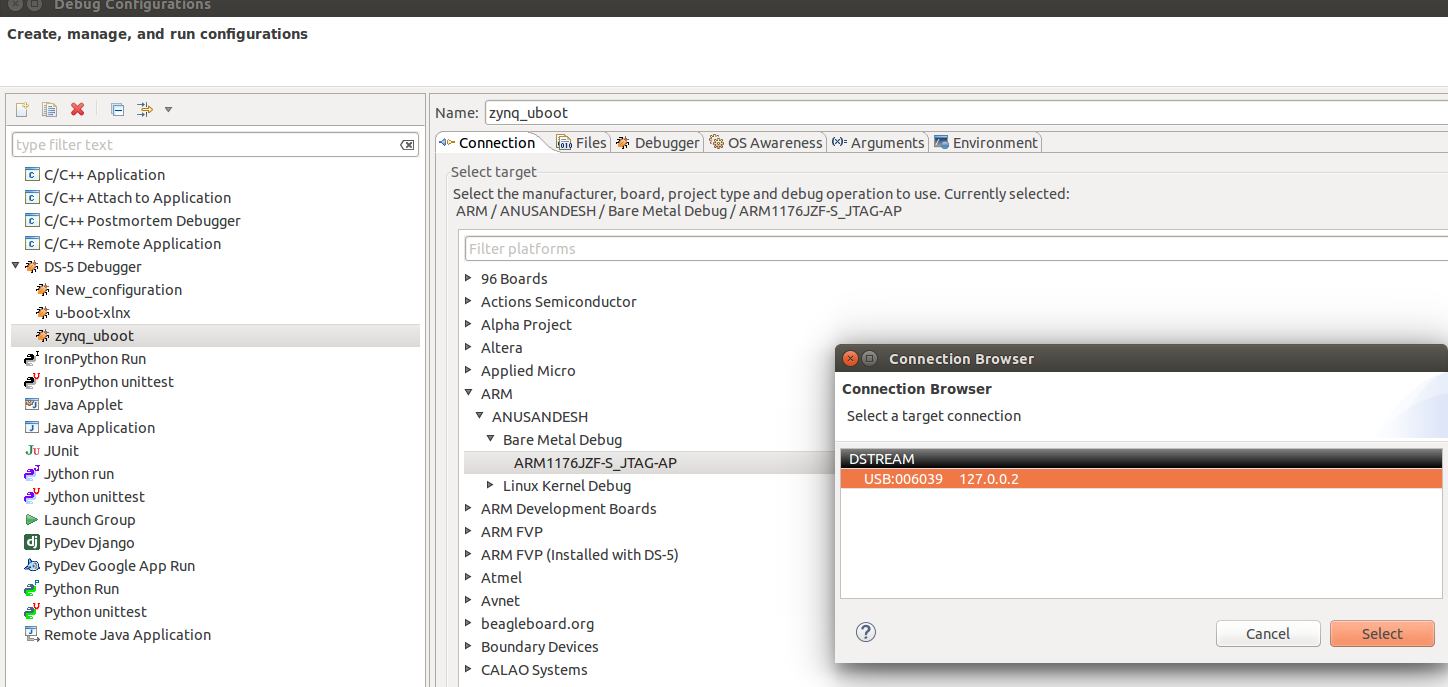
* **Procedure:**
  + From the main menu, select **Window--> Open Perspective--> DS-5 Debug**.
  + From the main menu, select **Run--> Debug Configurations** to open the Debug Configuration dialog.
  + In the configuration tree, **select DS-5 Debugger and then click New** to create a new configuration.
  + Enter a **suitable name for the new configuration**, in the Name field. For example, zynq\_uboot.
  + Use the **Connection tab to specify the target and connection settings**.
  + In the Select target area of the dialog, browse and select the platform you require. For example, to connect to the ARM1176JZF-S , Browse and select **home--> DS-5-Workspace-->Boards-->ARM --> ANUSANDESH--> BareMetal Debug--> ARM1176JZF-S\_JTAG-AP**.
  + Select your debug hardware unit in the Target Connection list. For example, DSTREAM.

The location of the folder and the files are shown in below picture



* To start debugging using D-Stream, we need to make sure that D-Stream connection has been made with the DS-5. To do this there is a Debug Configuration tab click on that one and select Connection from that. After that a dialogue box will appear. In that select the bare metal debug. Then we can see the USB id of the D-Stream connected. Now the DS-5 is connected to the D-Stream. It’s shown in below diagrams.



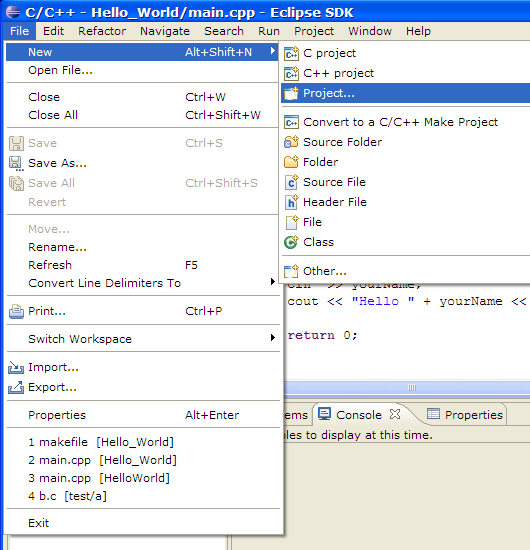


## 3.3. Creating Project in Eclipse

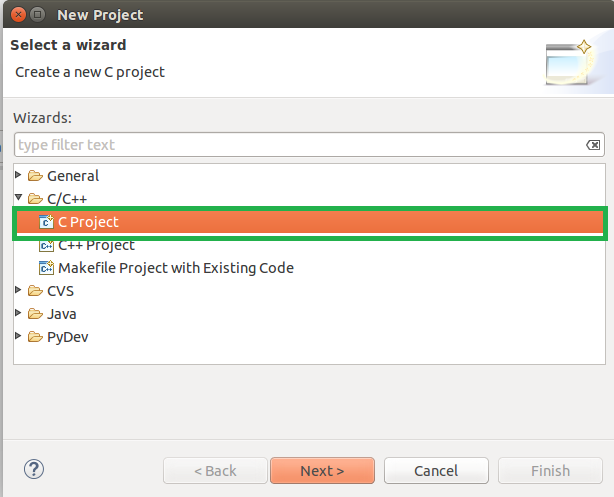
You can create a standard make or managed make C or C++ project.

To create a project:

1. Click **File > New > Project**.



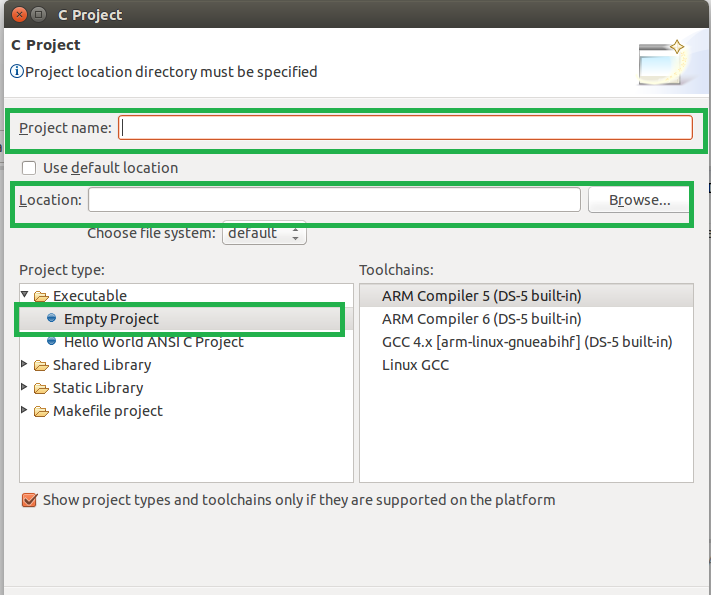
1. Select **C Project [**Select the type of project to create. Expand the **C/C++** folder and select **C Project]**



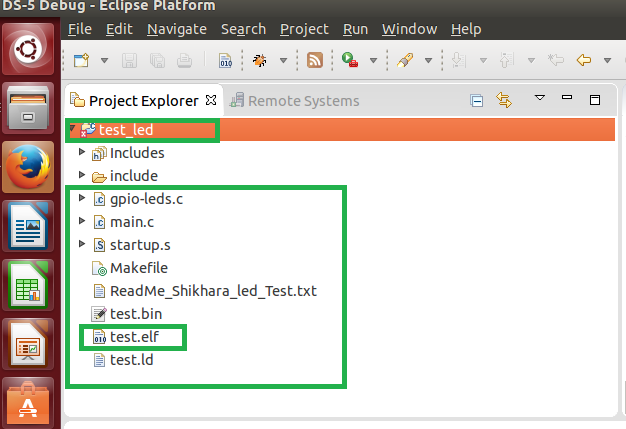
1. In the **Project name** field, type test\_led

Leave the **Use Default Location** option selected.

**Empty Project** provides a single source project folder that contains files

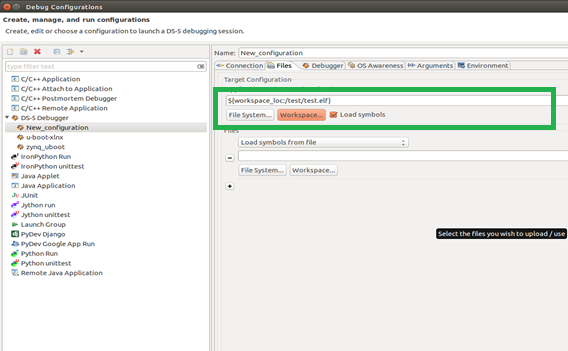


1. Project Explorer, will show the **project name, with source and executable [elf] file**



1. To debug a project:

* Click the **Run > Debug Configurations...** menu option. The **Debug** **Configurations** dialog opens.
* From the new configuration , workspace , select the elf file location
* From the new configuration ,Debugger, select entry point



# 4. Bare Metal Code

Compiling programs that run bare metal is different than compiling programs that run under an operating system and use standard libraries. gcc is configured by default to assume you are using an operating system and the standard libraries, which is how most programs are developed. Therefore, in order to compile bare metal programs we need to provide the appropriate options.

## 4.1. Linker Script

Relocation is the process of changing addresses already assigned to labels. This will also involve patching up all label references to reflect the newly assigned address. Primarily, relocation is performed for the following two reasons:

Section Merging

Section Placement

Section merging and placement is done by the linker. The programmer can control how the sections are merged, and at what locations they are placed in memory through a linker script file

## 4.2. Bare Metal – Sample Examples

|  |  |  |
| --- | --- | --- |
| **Description** | **Source Directory** | **Path / Location** |
| BareMetal GPIO[LEDS] Blink Test | Sample\_test\_leds | smb://192.168.200.242/kdocs/Knowledge\_Docs/Projects\_Based/Shikhara\_ANUSoC/BareCodeTest\_Shikhara/BareMetal-Code/Sample\_test\_leds |
| BareMetal UART Hello World Test | Sample\_test\_uart | smb://192.168.200.242/kdocs/Knowledge\_Docs/Projects\_Based/Shikhara\_ANUSoC/BareCodeTest\_Shikhara/BareMetal-Code/Sample\_test\_uart |
| BareMetal SMC NOR Fixed Pattern write and Read Test | Sample\_test\_nor | smb://192.168.200.242/kdocs/Knowledge\_Docs/Projects\_Based/Shikhara\_ANUSoC/BareCodeTest\_Shikhara/BareMetal-Code/Sample\_test\_nor |
| BareMetal SMC SRAM Fixed Pattern write and Read Test | Sample\_test\_sram | smb://192.168.200.242/kdocs/Knowledge\_Docs/Projects\_Based/Shikhara\_ANUSoC/BareCodeTest\_Shikhara/BareMetal-Code/Sample\_test\_sram |
| BareMetal NOR Flashing using xmodem | Sample\_xmodem\_nor\_flash | smb://192.168.200.242/kdocs/Knowledge\_Docs/Projects\_Based/Shikhara\_ANUSoC/BareCodeTest\_Shikhara/BareMetal-Code/Sample\_xmodem\_nor\_flash |
| BareMetal SPI Flashing using xmodem | Sample\_xmodem\_spi\_flash | smb://192.168.200.242/kdocs/Knowledge\_Docs/Projects\_Based/Shikhara\_ANUSoC/BareCodeTest\_Shikhara/BareMetal-Code/Sample\_xmodem\_spi\_flash |
| BL1 and BL2 Binaries | BL1-BL2\_Binaries | smb://192.168.200.242/kdocs/Knowledge\_Docs/Projects\_Based/Shikhara\_ANUSoC/BL1-BL2\_Binaries |

## 4.3. Compilation

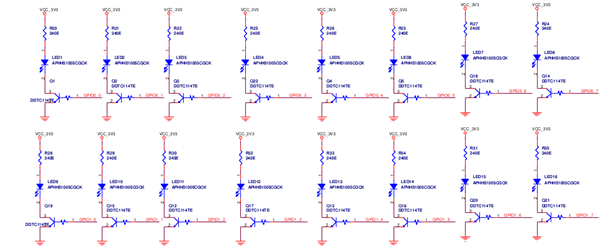
* Compile the code , before compilation , should have the cross-compiler installed on "Host PC running Ubuntu 14.04 LTS[64-bit]"
* Every sample test , will have the make file , run make to build elf file
* Typical Make file looks

arm-none-eabi-gcc -nostdlib -mcpu=cortex-a9 -o test.elf -T test.ld main.c gpio-leds.c startup.s -I ./include

arm-none-eabi-objcopy -O binary test.elf test.bin

# 5. Bare Metal Code –LEDS Blink Test

## 5.1. Test Setup / Board Connections



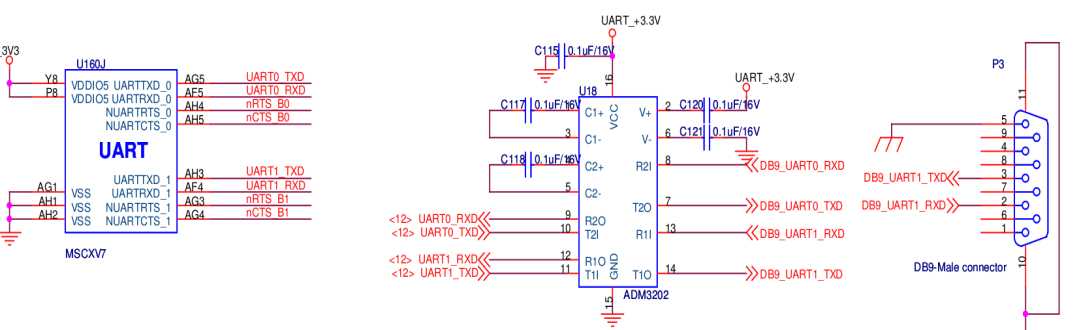
## 5.2. Test Procedure

Open the code with Eclipse:

1. Create work space , copy the "LED Blink Test" code from the Repository to work space
2. Check the linked script file especially "On-Chip RAM Memory Address" , Stack size (4KB) **0xD45B0000** ; On-Chip Memory Address 64KB
3. Check the memory-map is correct or not
   1. **SHIKHARA\_GPIO\_BANK\_A 0xD4576000**
   2. **SHIKHARA\_GPIO\_BANK\_B 0xD4577000**
4. Compile the code , before should have the cross-compiler installed on "Host PC running Ubuntu 14.04 LTS[64-bit]"
   1. arm-none-eabi-gcc -nostdlib -mcpu=cortex-a9 -o test.elf -T test.ld main.c gpio-leds.c startup.s -I ./include
   2. arm-none-eabi-objcopy -O binary test.elf test.bin
5. After creating the elf file, when we connect the target we can see in the command console which elf file is loading
6. Put the Breakpoints, see the expected output

# 6. Bare Metal Code – UART Hello World Test

## 6.1. Test Setup / Board Connections



## 6.2. Test Procedure

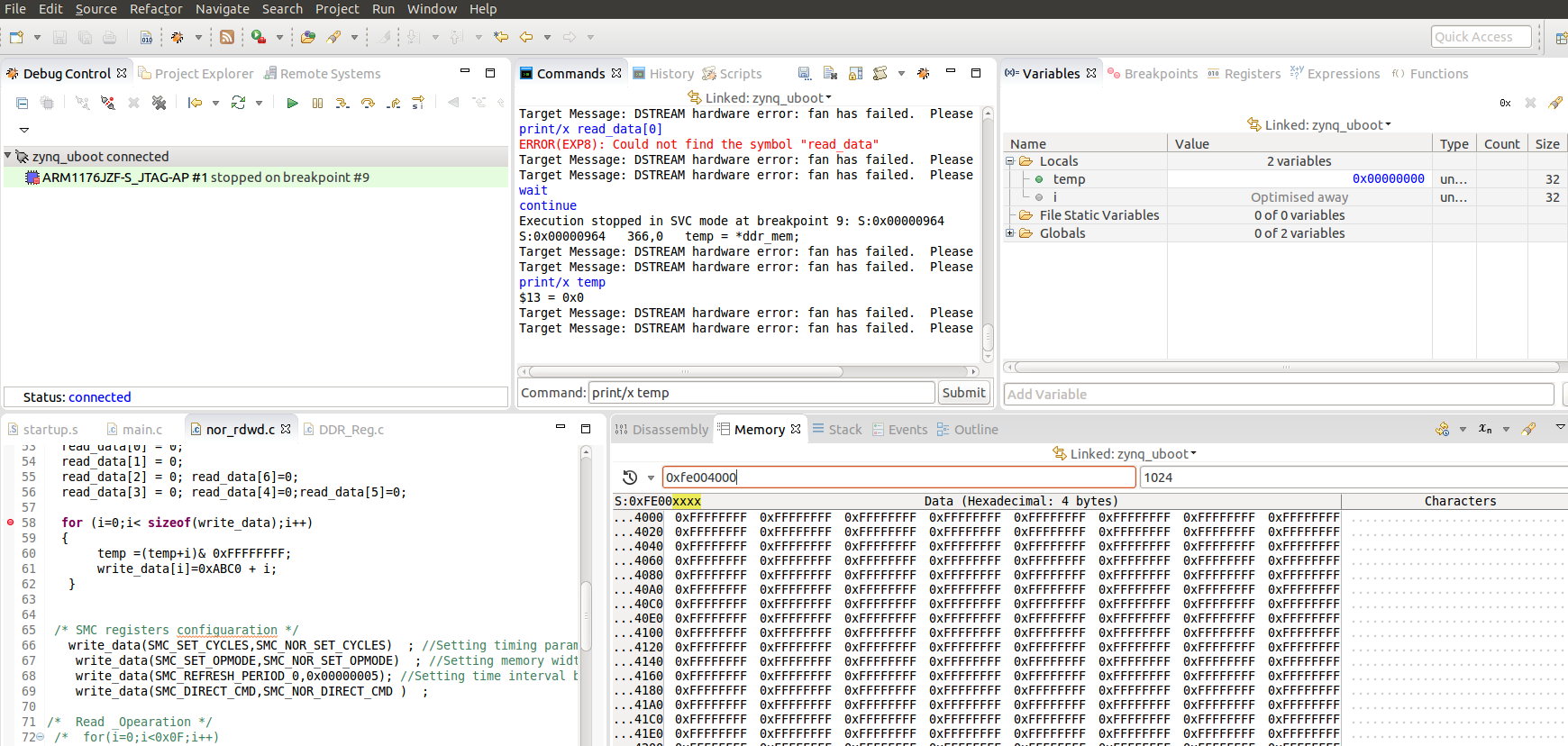
Open the code with Eclipse:

1. Check the linked script file especially "On-Chip RAM Memory Address" , Stack size (4KB)
2. Create work space , copy the "UART Test" code from the Repository to here
3. Check the memory map of the following
   1. **SHIKHARA\_UART0\_BASE 0xD457B000**
   2. **SHIKHARA\_UART1\_BASE 0xD457C000**
   3. **SHIKHARA\_UART2\_BASE 0xD457D000**
   4. **SHIKHARA\_UART3\_BASE 0xD457E000**
   5. **CONFIG\_PL011\_CLOCK 240000000 //24MHZ**
   6. **CONFIGURE\_BAUDRATE 115200**
4. Compile the code , before should have the cross-compiler installed on "Host PC running Ubuntu 14.04 LTS[64-bit]"
   1. arm-none-eabi-gcc -nostdlib -mcpu=cortex-a9 -o test.elf -T test.ld main.c shikhara-uart.c startup.s -I ./include
   2. arm-none-eabi-objcopy -O binary test.elf test.bin
5. After creating the elf file, when we connect the target we can see in the command console which elf file is loading
6. Put the Breakpoints, see the expected output

# 7. Bare Metal Code – SMC NOR Test

## 7.1. Test Procedure

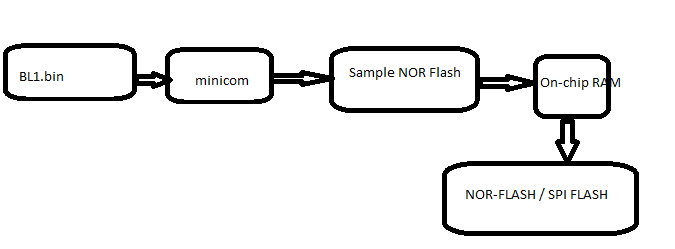
* Get the code Bare metal smc nor code from Repository
* Check the linked script file especially "On-Chip RAM Memory Address" , Stack size (4KB)
* Check for the SMC-NOR Flash timings [SMC\_NOR\_SET\_CYCLES , SMC\_NOR\_SET\_OPMODE, SMC\_NOR\_DIRECT\_CMD]
* Create work space , copy the "SM NOR Test" code from the Repository to here
* Check the memory map of the following NOR [0xC0000000 to 0XC3FFFFFF] & SMC Controller
* Open the memory tab, type the NOR base address 0XC0000000 ,See the blow fig
* Put the Breakpoints, see the expected output



# 8. Flash / Program / fuse BL1 into NOR

## 8.1. Test Procedure

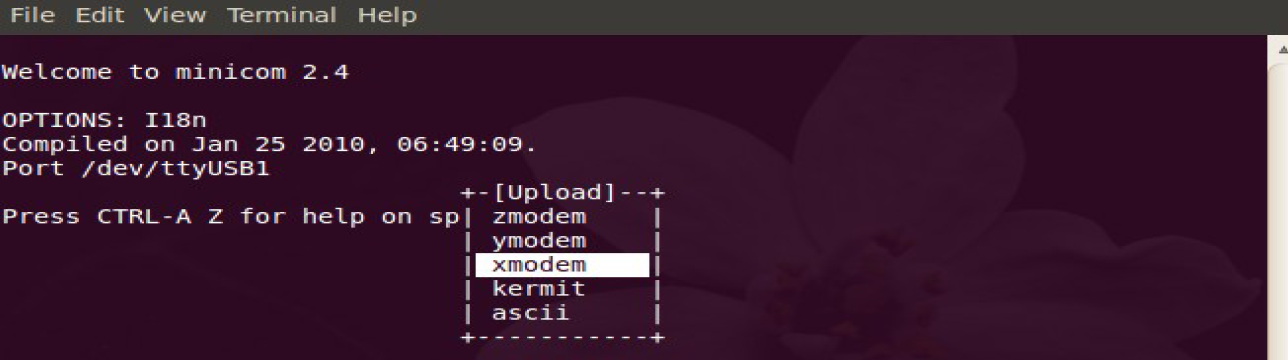
* Get the BL1 from the Pre build directory of repository
* Get the xmodem nor flash Bare metal code from repository
* Create work space , copy the "XMODEM NOR Flash" code from the Repository to here
* Using the ds-5, run the code from the on-chip ram , check the code is already compiled and generated elf file
* Open the minicom on Host PC , with 115200, 8N1 Configuration
* When “CCCC” characters appear on minicom window, File Transfer Procedure



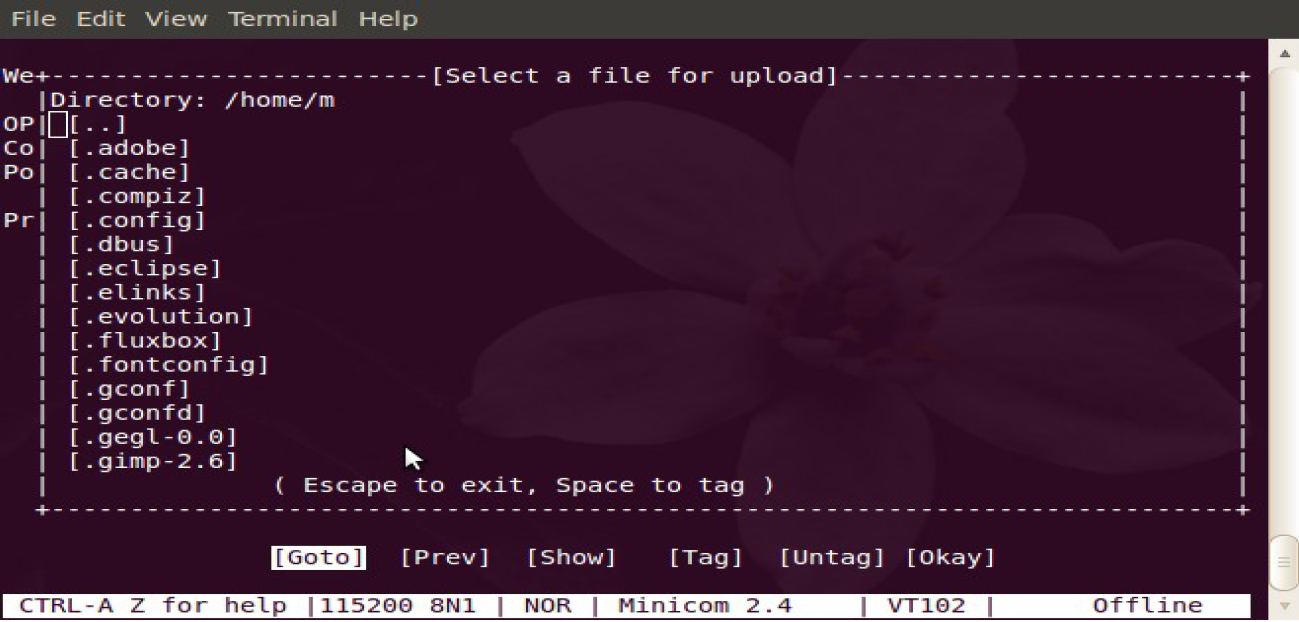
## 8.2. File Transfer from Host PC

Send a file through xmodem protocol using minicom. For this it is assumed the target device connected at the other end of the port uses the same protocol and configured with same settings like Bps/Par/bits.

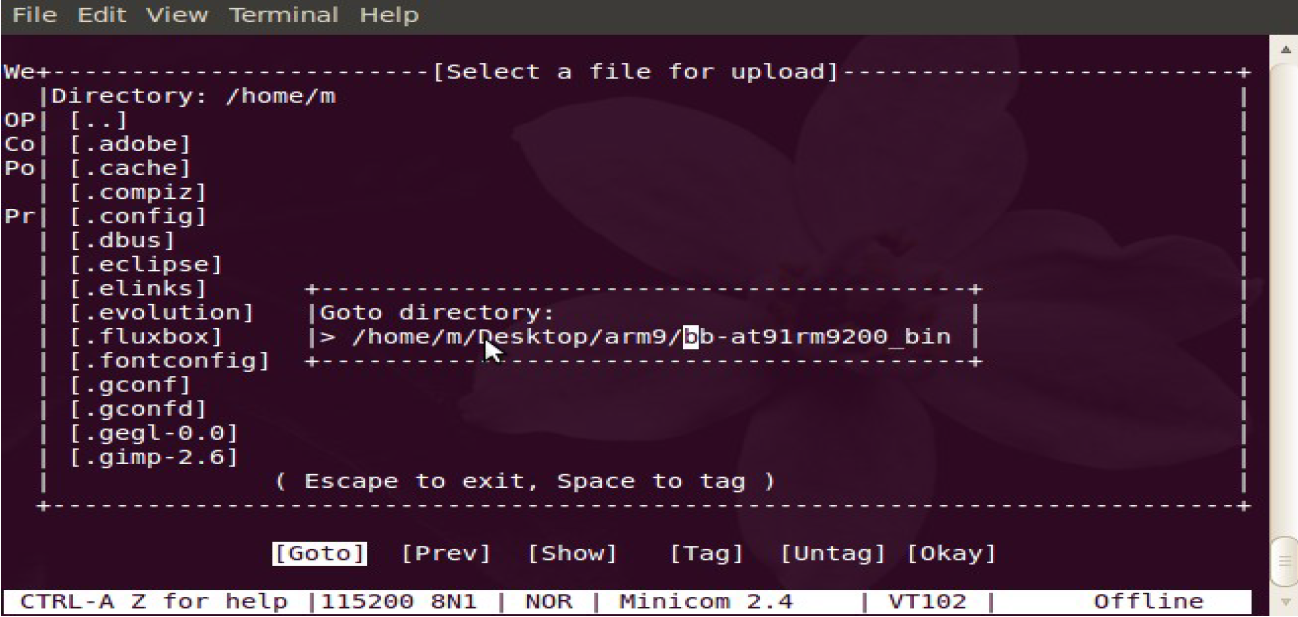
* To send a file press CTRL – A S , this selects the option Send file , Highlight xmodem protocol and press enter



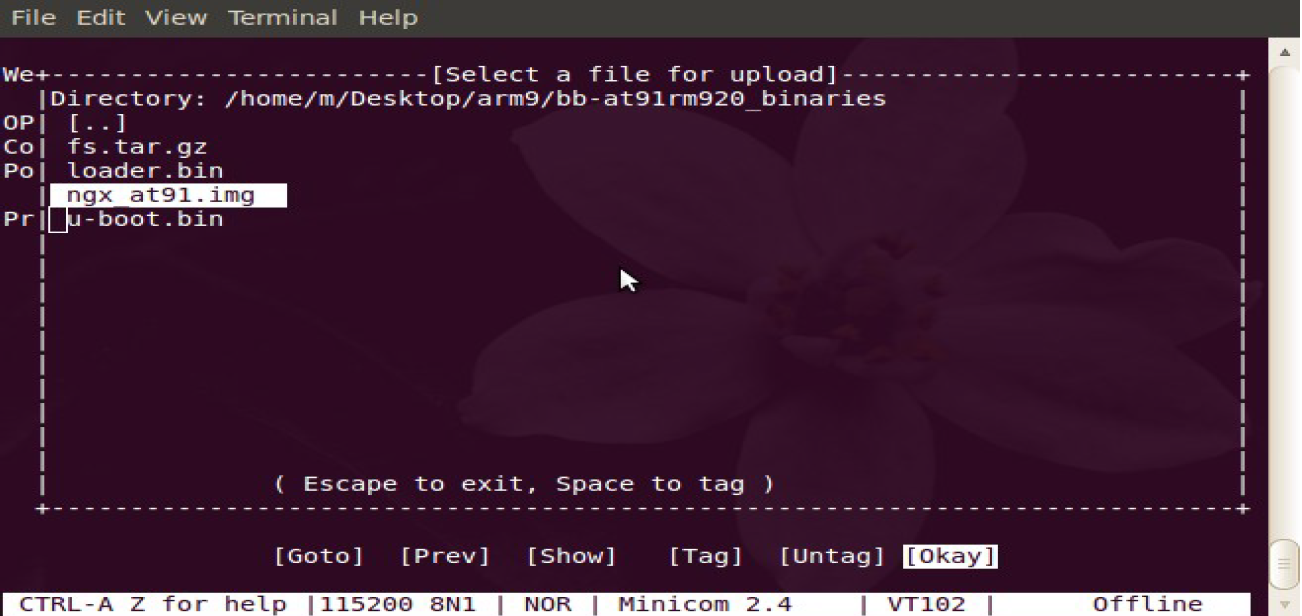
* Use arrow keys to highlight [Goto] and press enter



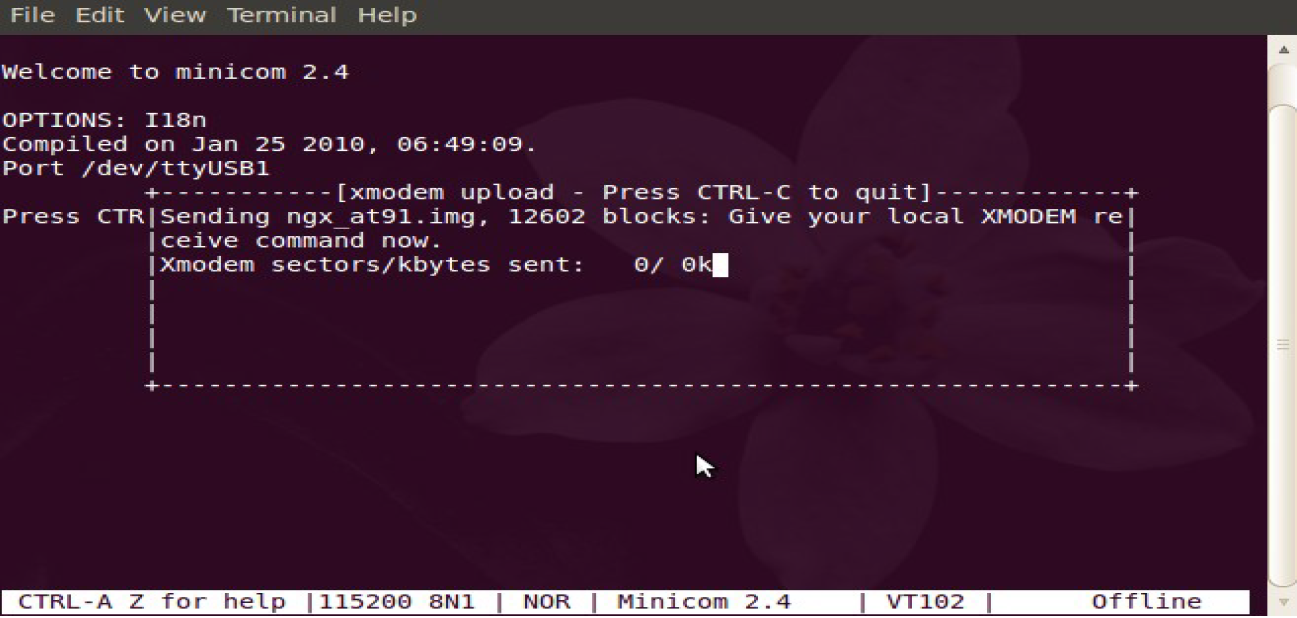
* Enter the path of the directory to send the file. To get the path of the files you you can do as follows...



* Highlight the file to send using arrow keys and press spacebar to select the file. Press enter, **select BL1.bin**



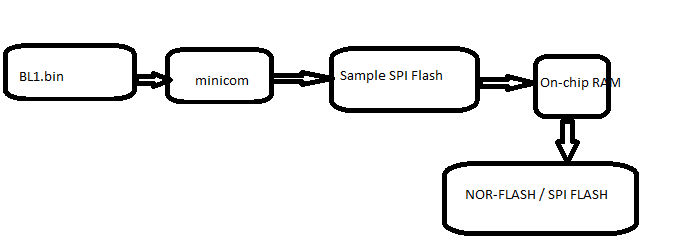
* File is sent and the progress is shown in xmodem sectors and Kbytes sent



# 9. Flash / Program / fuse BL1 into SPI Flash

## 9.1. Test Procedure

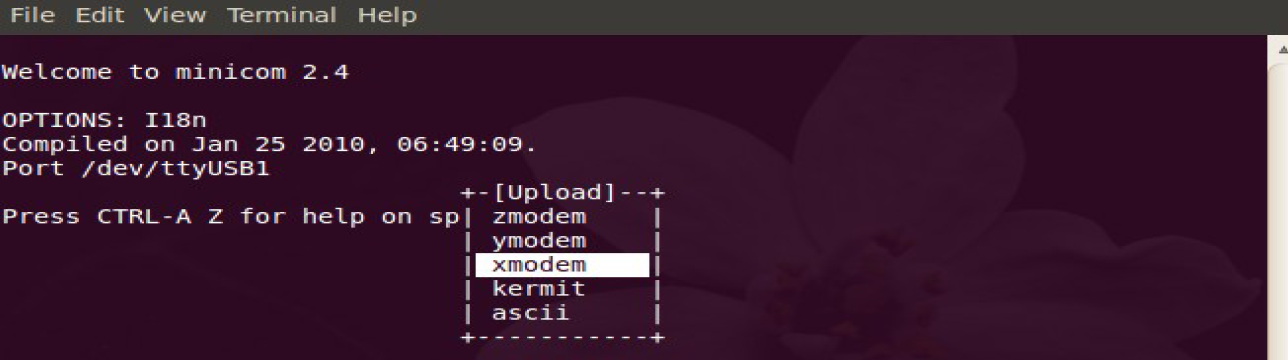
* Get the BL1 from the Pre build directory of repository
* Get the xmodem nor flash Bare metal code from repository
* Create work space , copy the "XMODEM SPI Falsh" code from the Repository to here
* Using the ds-5, run the code from the on-chip ram , check the code is already compiled and generated elf file
* Open the minicom on Host PC , with 115200, 8N1 Configuration
* When “CCCC” characters appear on minicom window, File Transfer Procedure



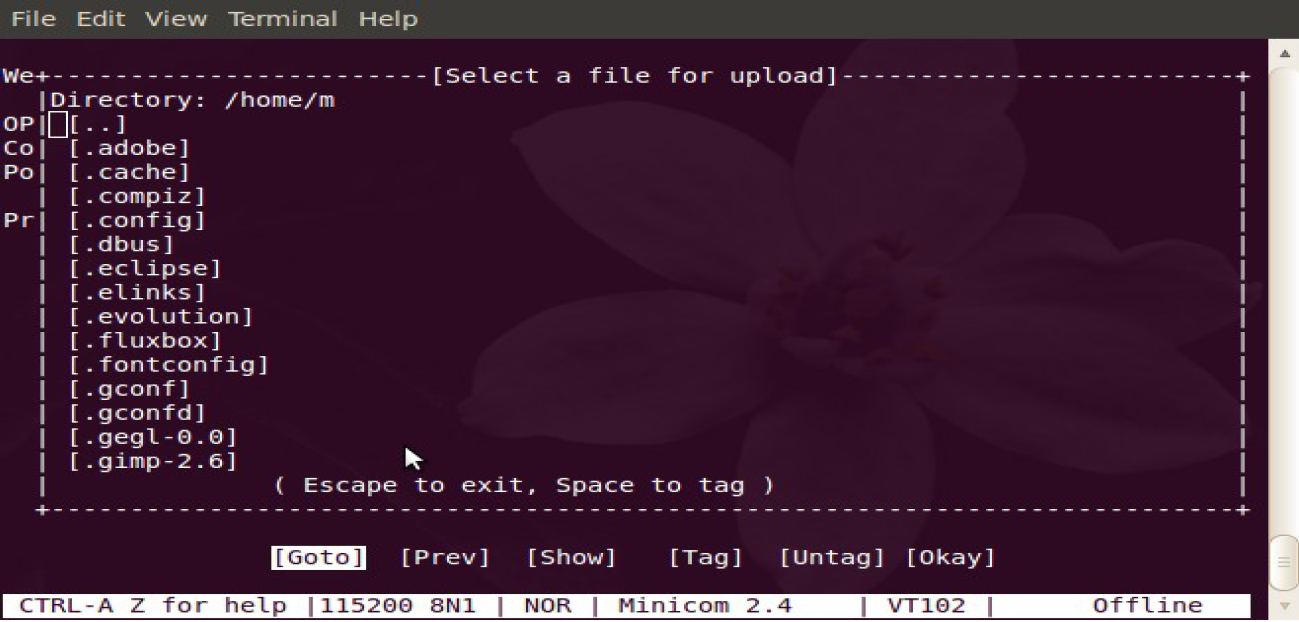
## 9.2. File Transfer from Host PC

Send a file through xmodem protocol using minicom. For this it is assumed the target device connected at the other end of the port uses the same protocol and configured with same settings like Bps/Par/bits.

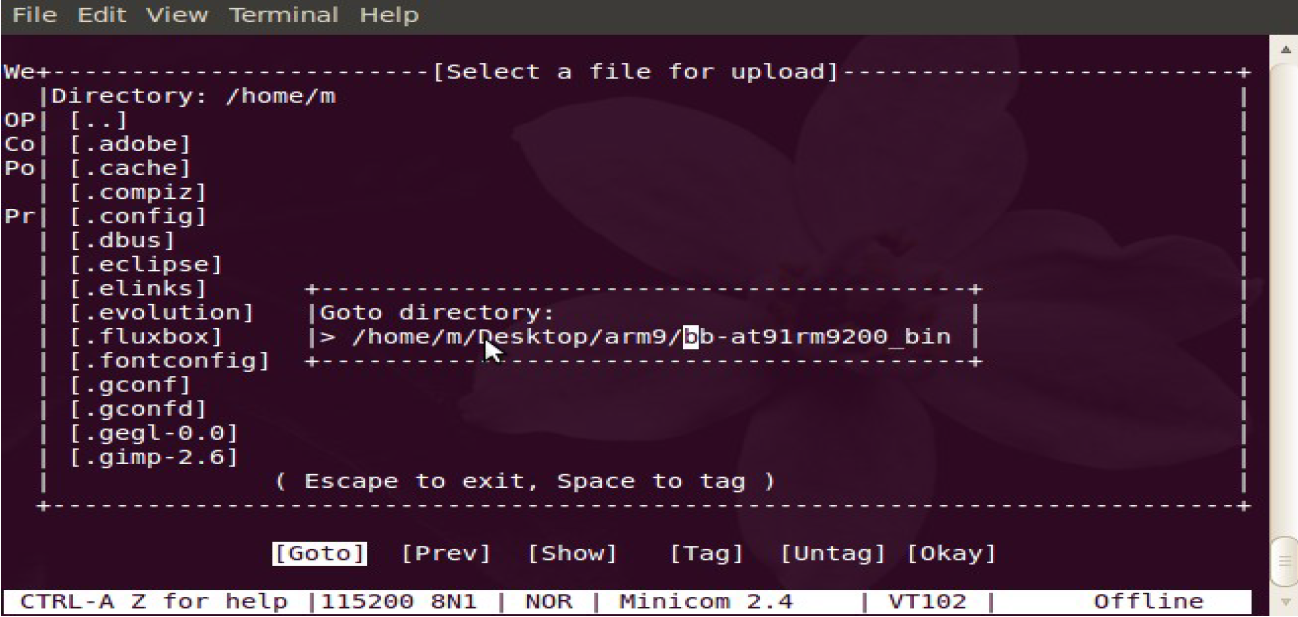
* To send a file press CTRL – A S , this selects the option Send file , Highlight xmodem protocol and press enter



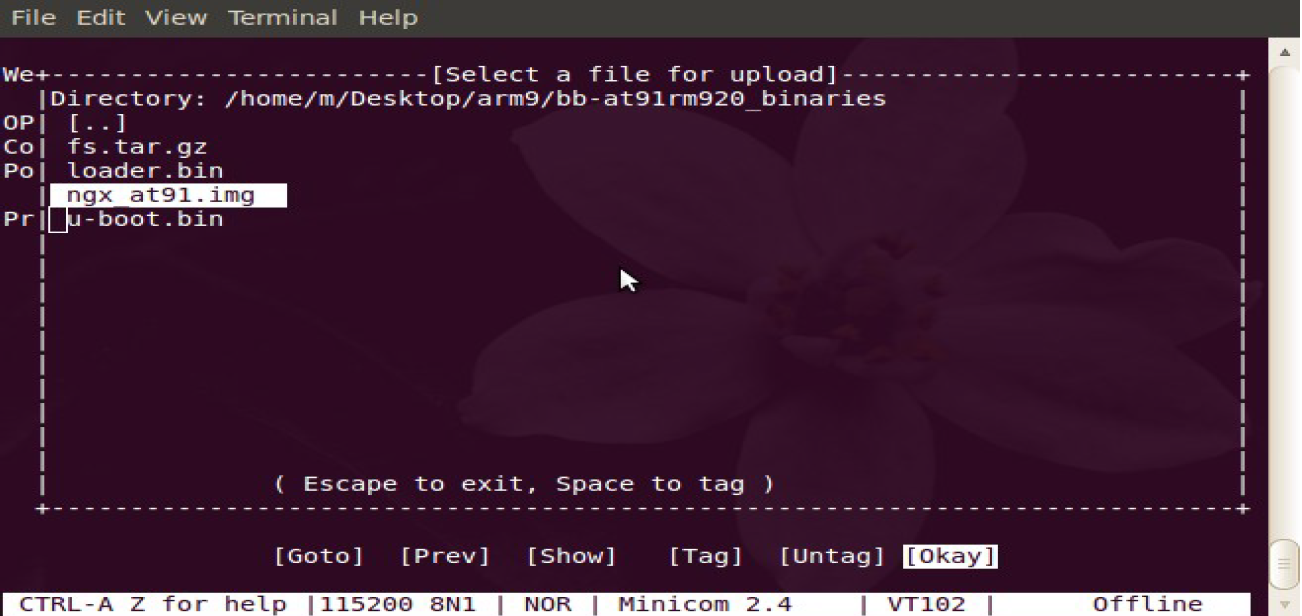
* Use arrow keys to highlight [Goto] and press enter



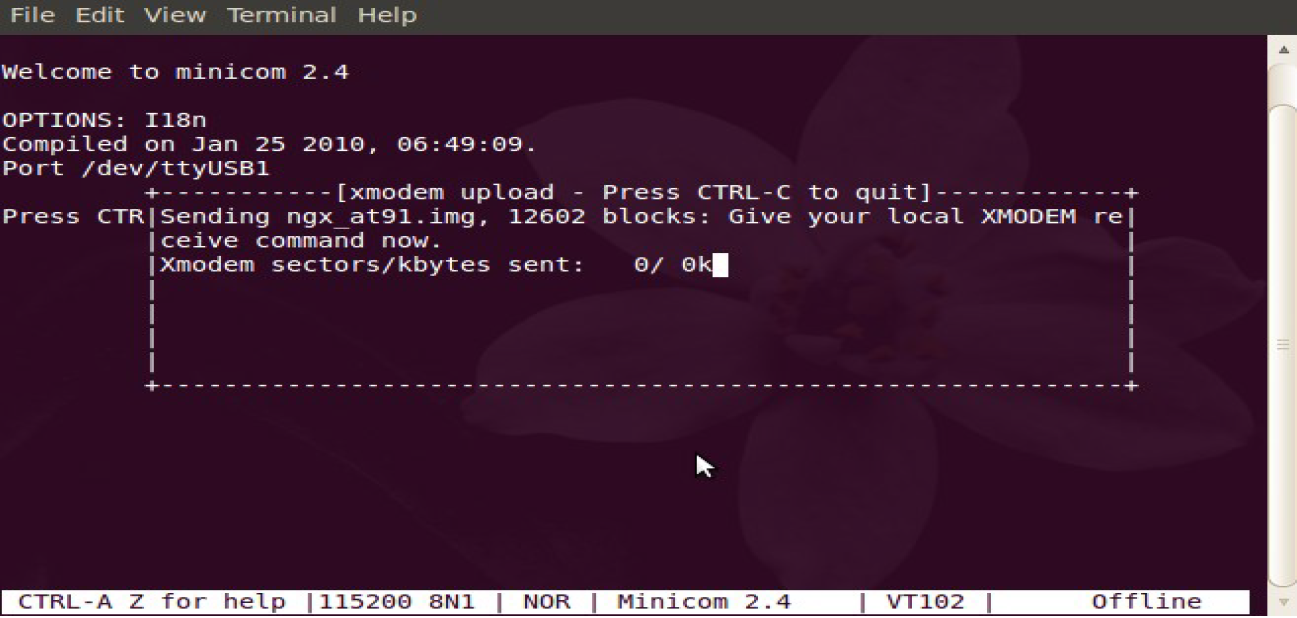
* Enter the path of the directory to send the file. To get the path of the files you you can do as follows...



* Highlight the file to send using arrow keys and press spacebar to select the file. Press enter ,**select BL1.bin**



* File is sent and the progress is shown in xmodem sectors and Kbytes sent



# 10. Flash / Program / fuse BL1 into SD

## 10.1. Steps to format SD card

(i) To know the disk’s available and know the name of the SD

#fdisk –l

(ii) The **fdisk** utility does not seem to erase the first few bytes of the first sector in the card when the partition table is saved. Use dd to erase the first sector.

#dd if=/dev/zero of=/dev/sdX bs=1024 count=1

(iii) Partition the SD card. We will create two partitions on the SD card. One 200 MB sized boot partition and a second partition taking the remaining space on the SD card.

#fdisk /dev/sdX

(iv) Now configure the sectors, heads and cylinders of the SD card.

Command (m for help): x

Expert command (m for help): h

Number of heads (1-256, default 30): 255

Expert command (m for help): s

Number of sectors (1-63, default 29): 63

Expert command (m for help): c

Number of cylinders (1-1048576, default 2286): <new\_cylinders calculated from above>

Command (m for help): r

(v) Now the actual partitions can be created

Command (m for help): n

Partition type:

P primary (0 primary, 0 extended, 4 free)

e extended

Select (default p): p

Partition number (1-4, default 1): 1

First sector (2048-15759359, default 2048):

Using default value 2048

Last sector, +sectors or +size {K, M, and G} (2048-15759359, default 15759359): +200M

Command (m for help): n

Partition type:

P primary (1 primary, 0 extended, 3 free)

E extended

Select (default p): p

Partition number (1-4, default 2): 2

First sector (411648-15759359, default 411648):

Using default value 411648

Last sector, +sectors or +size{K,M,G} (411648-15759359, default 15759359):

Using default value 15759359

(vi) Check the new partition table and write the changes

Command (m for help): p

Disk /dev/sdb: 8068 MB, 8068792320 bytes

249 heads, 62 sectors/track, 1020 cylinders, total 15759360 sectors

Units = sectors of 1 \* 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk identifier: 0x920c958b

Device Boot Start End Blocks Id System

**/dev/sdb1 2048 411647 204800 c W95 FAT32 (LBA)**

**/dev/sdb2 411648 15759359 7673856 83 Linux**

Command (m for help): w

The partition table has been altered!

Calling ioctl () to re-read partition table.

WARNING: If you have created or modified any DOS 6.x

Partitions please see the fdisk manual page for additional

Information.

Syncing disks.

## 10.2 Copying Images to SD

* SD Card Partition Table

|  |  |  |
| --- | --- | --- |
|  | **Block start** | **length** |
| **BL1** | 0x11 | 0x8C00/0X200=70 |
| **BL2(u-boot.bin)** | 0x80 | 0x33E(830) |

We need to make sure we leave some space for BL1 and BL2 (u-boot) starting from offset 8704B. Here is an example SD card layout:

+-------+------- +--------+------+------------------------

| MBR | ... | BL1 | BL2 | FAT partition...

+-------+-------+--------+---------+----------------------

0 512 8704 65536 1M

(Offsets in bytes)

* Here is an example SD card layout, as displayed by **fdisk**:

Device Boot Start End Blocks Id System

/dev/sdc1 2048 8054783 4026368 c W95 FAT32 (LBA)

* Run the following commands to install BL1 & BL2 (U-Boot).

dd if=BL1.bin of=/dev/sdc1 bs=512 seek=17 (0x11 blocks need to skip)

seek = BLOCKS [skip BLOCKS obs-sized blocks at start of output]

Make sure you replace /dev/sdx with the correct path for your SD Card. In my case it is /dev/sdb and the commands look like

dd if=u-boot.bin of=/dev/sdx bs=512 seek=128 (0x80 block need to skip)

# 11. Flash / Program / fuse BL1 into NAND

## 11.1 Test Procedure

* From the NOR/SD/SPI Booting , will get u-boot prompt
* We will at u-boot command prompt ,
* Need to read BL1 from SPI Flash

sf read address offset length [ read 'length' bytes starting at 'offset' to memory at 'address' ]

* Need to write BL1 to NAND Flash

nand write - address offset size [ read/write 'size' bytes starting at offset 'offset' from the address]