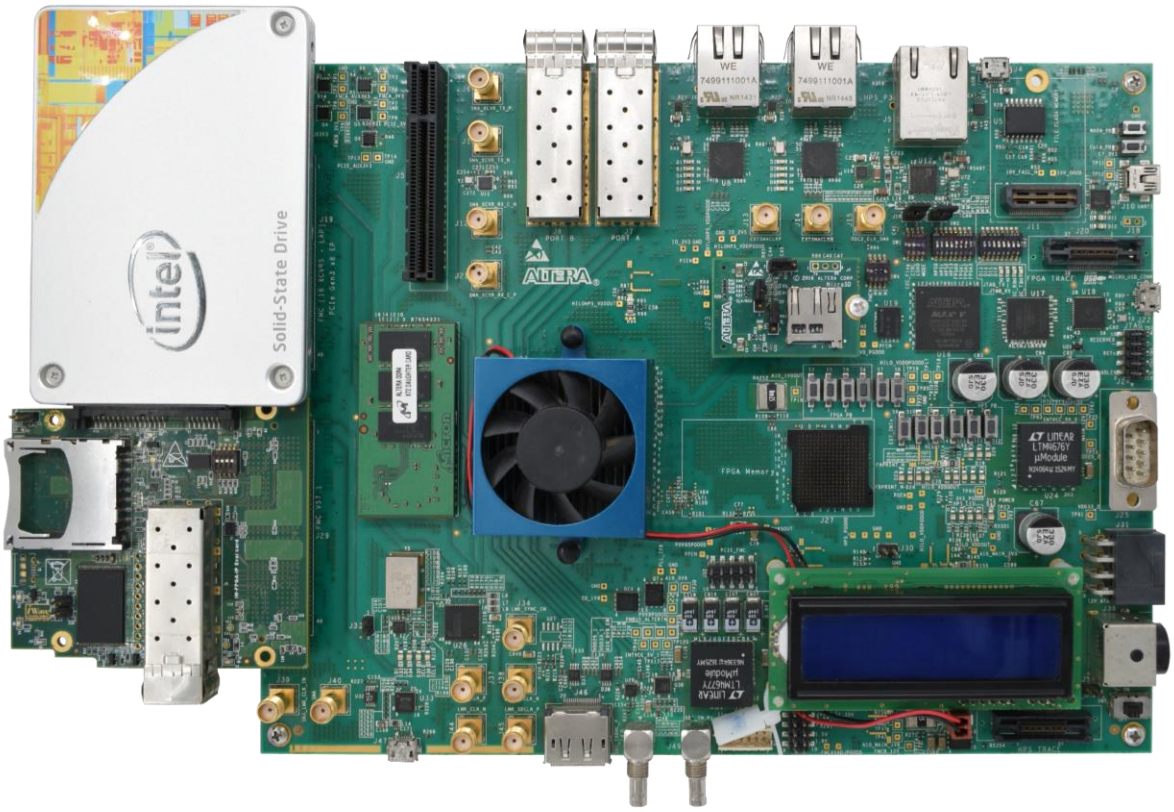
iWave SATA Host Controller Linux User Guide



Disclaimer

iWave Systems reserves the right to change details in this publication including but not limited to any Product specification without notice.

No warranty of accuracy is given concerning the contents of the information contained in this publication. To the extent permitted by law no liability (including liability to any person by reason of negligence) will be accepted by iWave Systems, its subsidiaries or employees for any direct or indirect loss or damage caused by omissions from or inaccuracies in this document.

Trademarks

All registered trademarks and product names mentioned in this publication are used for identification purposes only.

Technical Support

iWave Systems technical support team is committed to provide the best possible support for our customers so that our Hardware and Software can be easily migrated and used.

For assistance, contact our Technical Support team at,

Address : iWave Systems Technologies Pvt. Ltd.
7/B, 29th Main,
BTM Layout 2nd Stage,
Bangalore,
India – 560076
Email : ascdo@iwavesystems.com
Website : www.iwavesystems.com

Table of Contents

List of Figures

1 Introduction

1.1 Purpose

The purpose of this document is to help the software engineer to program and test the iWave SATA Host controller and this will also guide to configure the Linux development environment in the Host PC and build the board support package.

1.2 Scope

The document describes the Linux Operating System and related software installed on the Arria10 Development Board. The Linux BSP is a collection of binary, source code, and support files that can be used to create a Linux kernel image for iWave SATA Host controller support on Arria10 Development Board.

2 Software Development environment setup

2.1 Host Requirements

- A Linux host PC with latest version (ex. Ubuntu version 14.04)
- Root permission on the Development Host.
- Cross compiler package for Arria10 development board.

2.2 Host package installation

- Open a terminal window and install the below packages in host PC.
  $ sudo apt-get update
  $ sudo apt-get install -y gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath libsdl1.2-dev xterm
  $ sudo apt-get install -y openjdk-7-jre tzdata-java libcups2 libjpeg8 icedtea-7-jre-jamvm openjdk-7-jre-headless openjdk-7-jdk git gawk wget g++ chrpath diffstat texinfo make

2.3 Standalone Compilation

The following steps will help to build the linux kernel for Arria 10 Evaluation Board.

2.3.1 Linux kernel

- Extract toolchain.tar.gz in to the /opt directory.
  host@host/<Directory>~$tar -xvzf <Release_folder>/Software/Binaries/toolchain.tar.gz -C /opt

- Create a directory and open the directory in host to build the Linux.
  host@host~$ mkdir <directory_name>
  host@host~$ cd <directory_name>

- Extract linux-socfpga.tar.gz file in to newly created directory.
  host@host/<Directory>~$tar -xvzf <Release_folder>/Software/Source_Code/linux-socfpga.tar.gz

- Copy the kernel patch file to current directory.
  host@host/<Directory>~$cp <Release_folder>/Source_Code/PATCH001-iW-ASCDO-SC-01-R1.0-REL1.0-Linux4.1.33-Ltsi-iWave-SATA-support.patch .

- Change the directory to Linux source code directory.
  host@host/<Directory>~$cd <path_to linux-socfpga >/linux-socfpga

- To apply the iWave SATA Host controller patch file, execute the below command.

host@host/<Directory>~$ patch -Np1 < ../PATCH001-iW-ASCDO-SC-01-R1.0-REL1.0-Linux4.1.33-Ltsi-iWave-SATA-support.patch

- To configure the kernel for Arria10 Development Kit, execute the below command.
  host@host/<Directory>~$make ARCH=arm CROSS_COMPILE=/opt/gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux/bin/arm-linux-gnueabihf- socfpga_defconfig

- To compile the kernel module drivers and kernel image, execute the below commands.
  host@host/<Directory>~$make ARCH=arm CROSS_COMPILE=/opt/gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux/bin/arm-linux-gnueabihf- zImage

- To compile the dts, execute the below commands.
  host@host/<Directory>~$make ARCH=arm CROSS_COMPILE=/opt/gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux/bin/arm-linux-gnueabihf- dtbs

- After successful compilation, kernel image (zImage) and device tree (.dtb) image will be generated and to generate the SD card image rename *socfpga_arria10_socdk_sdmmc.dtb* to *socfpga_arria10_socdk.dtb*
  host@host/<Directory>~$mv arch/arm/boot/dts/socfpga_arria10_socdk_sdmmc.dtb arch/arm/boot/dts/socfpga_arria10_socdk.dtb

- Use the below binary files for the SD card image generation as specified in the section Binary Programming
  ~/linux-socfpga/arch/arm/boot/zImage
  ~/linux-socfpga/arch/arm/boot/dts/socfpga_arria10_socdk.dtb

2.3.2 Changes with respect to standard driver

- drivers/ata/ahci.h

The SATA port registers are different in our SATA IP. We added list of registers under IW_SATA_ENABLE.

```
#ifdef IW_SATA_ENABLE
/* registers for each SATA port */
PORT_LST_ADDR                = 0x28, /* command list DMA addr */
PORT_LST_ADDR_HI      = 0x2C, /* command list DMA addr hi */
PORT_FIS_ADDR                = 0x30, /* FIS rx buf addr */
PORT_FIS_ADDR_HI              = 0x34, /* FIS rx buf addr hi */
PORT_IRQ_STAT                = 0x38, /* interrupt status */
PORT_IRQ_MASK                = 0x3C, /* interrupt enable/disable mask */
PORT_CMD                     = 0x40, /* port command */
```

```
PORT_TFDATA             = 0x44, /* taskfile data */
PORT_SIG                = 0x48, /* device TF signature */
PORT_SCR_STAT           = 0x4C, /* command issue */
PORT_SCR_CTL            = 0x50, /* SATA phy register: SStatus */
PORT_SCR_ERR            = 0x54, /* SATA phy register: SControl */
PORT_SCR_ACT            = 0x58, /* SATA phy register: SError */
PORT_CMD_ISSUE          = 0x5C, /* SATA phy register: SActive */
/* Not implemented */
PORT_SCR_NTF            = 0x60, /* SATA phy register: SNotification */
PORT_FBS                = 0x64, /* FIS-based Switching */
PORT_DEVSLP             = 0x68, /* device sleep */
```

- drivers/ata/libata-eh.c

ata_read_log_page is not supported in our SATA IP. So the function definition and function call is disabled from driver.

- config

User has to enable AHCI SATA Support from menuconfig.

```
CONFIG_SATA_AHCI=y
CONFIG_SATA_AHCI_PLATFORM=y
```

- arch/arm/boot/dts/socfpga_arria10_socdk.dtsi

User has to add DTS entry for iWave sata Ip in socfpga_arria10_socdk.dtsi. The compatible entry should be generic-ahci. “reg” entry and “interrupt” entry should be there in device tree.

3 Binary Programming

The make_sdimage.py is a tool which can be used to create a bootable SD card image. The tool runs in a linux system and is used to partition the micro SD card and program the binaries in to the card.

3.1 Requirements

To program the binaries for Arria10 platform, following Items are required:

- Micro SD card reader.
- Host PC (linux).
- Micro SD
- Binary files *(uboot_w_dtb-mkpimage.bin, zImage, socfpga_arria10_socdk.dtb, rootfs.tar.gz, ghrd_10as066n2.rbf)*

The below figure shows the memory layout for the boot device.

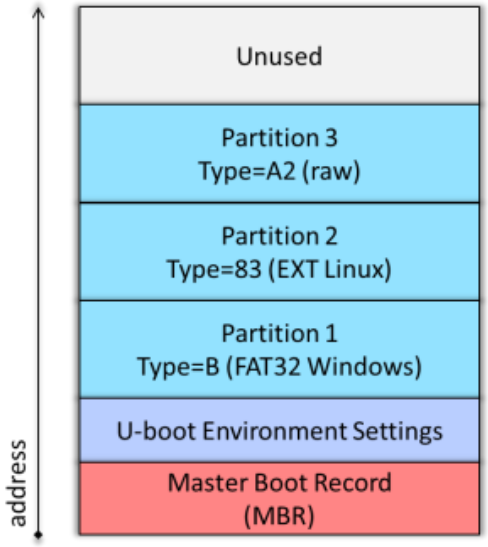


Figure 1: Boot device memory layout

3.2 SD card image creation procedure

- Create a directory and open the directory in host to generate the SD card Image.
  host@host~$ mkdir <directory_name>
  host@host~$ cd <directory_name>

- Copy Binary files (***uboot_w_dtb-mkpimage.bin, zImage, socfpga_arria10_socdk.dtb, rootfs.tar.xz, ghrd_10as066n2.rbf***) to the newly created directory

- Copy *make_sdimage.py* file to the current directory
  host@host/<Directory>~$ cp <Release_folder>/Software/Source_Code/Tools/make_sdimage.py .

- Create "rootfs" directory and extract the rootfs.tar.xz file to it.
  host@host/<Directory>~$ mkdir rootfs
  host@host/<Directory>~$ tar -xf rootfs.tar.xz -C rootfs

- Run the below command to create the SD card image.
  sudo ./make_sdimage.py -f -P uboot_w_dtb-mkpimage.bin,num=3,format=raw,size=10M,type=A2 -P rootfs/,num=2,format=ext3,size=1500M -P zImage,ghrd_10as066n2.rbf,socfpga_arria10_socdk.dtb,num=1,format=vfat,size=500M -s 2G -n sdimage_a10_sata.img*

- Compress the newly created .img file using below command (*Optional*)
  host@host/<Directory>~$ tar -zvcf sdimage.tar.gz sdimage_a10_sata.img

3.3 Programming the Binaries

3.3.1 Booting from SD Card.

- Make sure BSEL in Arria 10 Develpment board is in SD Boot mode.
- Copy the the SD card images from the release folder
  <Release_folder>/Software/Binaries/sdimage.tar.gz

- Refer **APPENDIX A : SD card programming** to prepare the SD card for boot

4 Linux SATA Sanity Testing

4.1 SATA read and write using the Linux file system.

- Format the partition using the below command
  mkfs.vfat -F 32 /dev/sda

- Mount the card using the **mount** command as shown below
  mount /dev/sda(1) /mnt/

- Use below commands for read/write a file from SATA Disk.
  mkdir /mnt/sata/
  cp /home/root/sata.txt /mnt/sata
  cp /mnt/sata/sata.txt /home/root/sata1.txt
  Note: sata.txt and sata1.txt are just example files any files can be copied using cp command.
- Check whether the file differs.
  diff /home/root/sata.txt /home/root/sata1.txt
  command should execute without any error messages.

- Un-mount the partition using the below command
  umount /mnt

4.2 File write and read by using the Linux dd command.

- Use **dd** command to read/write a file from card.
  dd if=/dev/urandom of=/tmp/data bs=1M count=10
  dd if=/tmp/data of=/dev/sda(1) bs=1M count=10
  dd if=/dev/sda(1) of=/tmp/data1 bs=1M count=10
- Use **md5sum** command to verify both files
  md5sum /tmp/data /tmp/data1
  sha values reported by md5sum should be equal for data and data1 files

5 APPENDIX A : SD card programming

This section explains how to create the SD card necessary to boot Linux, using the SD card image available with the precompiled Linux binaries package.

5.1 Creating SD Card on Linux

The required steps are:

- Copy SD card image file from the release package to PC:

  $ tar -xzf sdimage.tar.gz

  This will create the file sdimage.img which contains the SD card image file.
- Determine the device associated with the SD card on the host by running the following command before and after inserting the card in the reader:

  $ cat /proc/partitions

  Let's assume it is /dev/sdx.
- Use dd utility to write the SD image to the SD card:

  $ sudo dd if=sdimage.img of=/dev/sdx bs=1M
- Use sync utility to flush the changes to the SD card:

  $ sudo sync

5.2 Creating SD Card on Windows

This section explains how to write the SD image to the SD card on a Windows PC.

- Copy SD card image file (sdimage.tar.gz) from the release package to PC.
- Extract the SD card image from the compressed archive sdimage.tar.gz using WinZip or similar tools. This creates the file sdimage.img.
- UseWin32DiskImager to write the image to the SD card. The tool can be downloaded from here:

  https://sourceforge.net/projects/win32diskimager/files/latest/download
- Browse sdimage.img file and click on Write button to program the SD card.