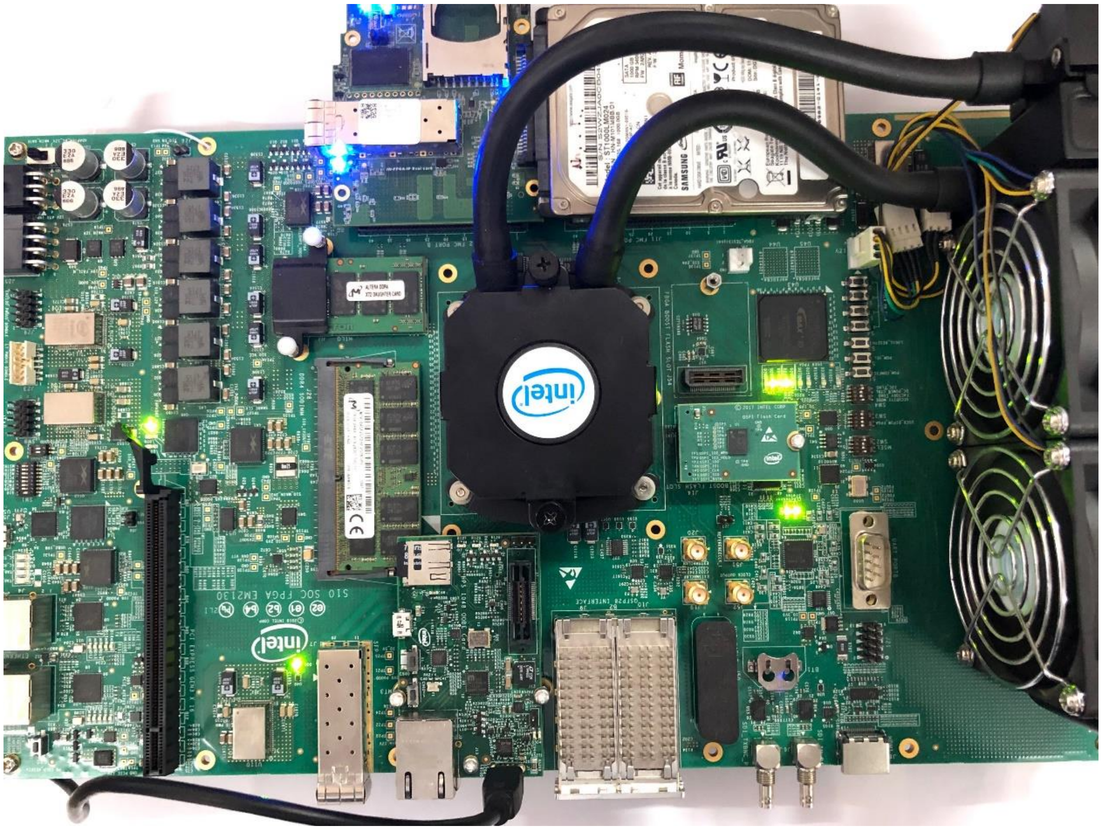
iWave SATA Host Controller
Linux User Guide



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 : emevx@iwavesystems.com
Website : www.iwavesystems.com

Table of Contents

List of Figures

1 Introduction

1.1 Purpose

The purpose of this document is help 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 Stratix10 Board. The Linux BSP is a collection of binaries, source code, and support files that can be used to create a Linux kernel image for iWave SATA Host controller support on Stratix10 Board.

2 Hardware Environment Setup

2.1 Stratix10 Hardware Setup

On Stratix10 Development kit, Connect the **iWave SATA FMC Daughter Card** to the FMC connector **FMC PORT B (J12)** as shown in the **Figure 1: Startix10 Hardware Setup**.

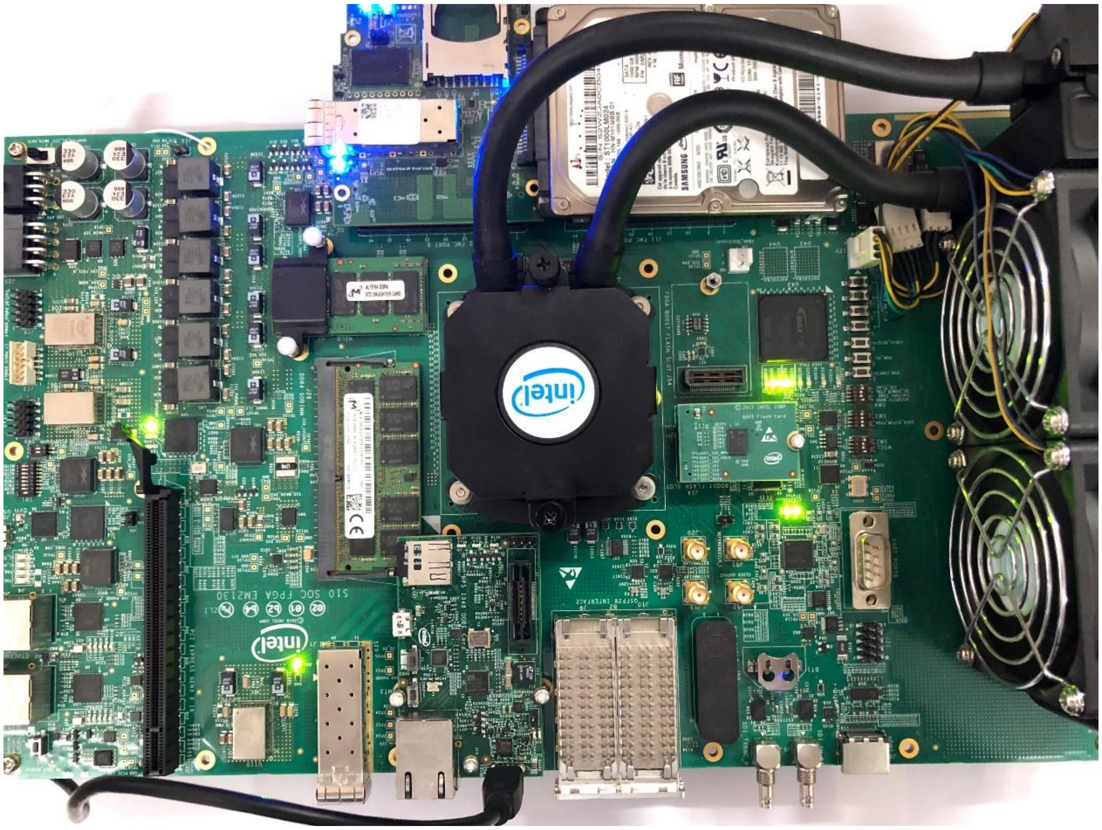


Figure 1: Startix10 Hardware Setup

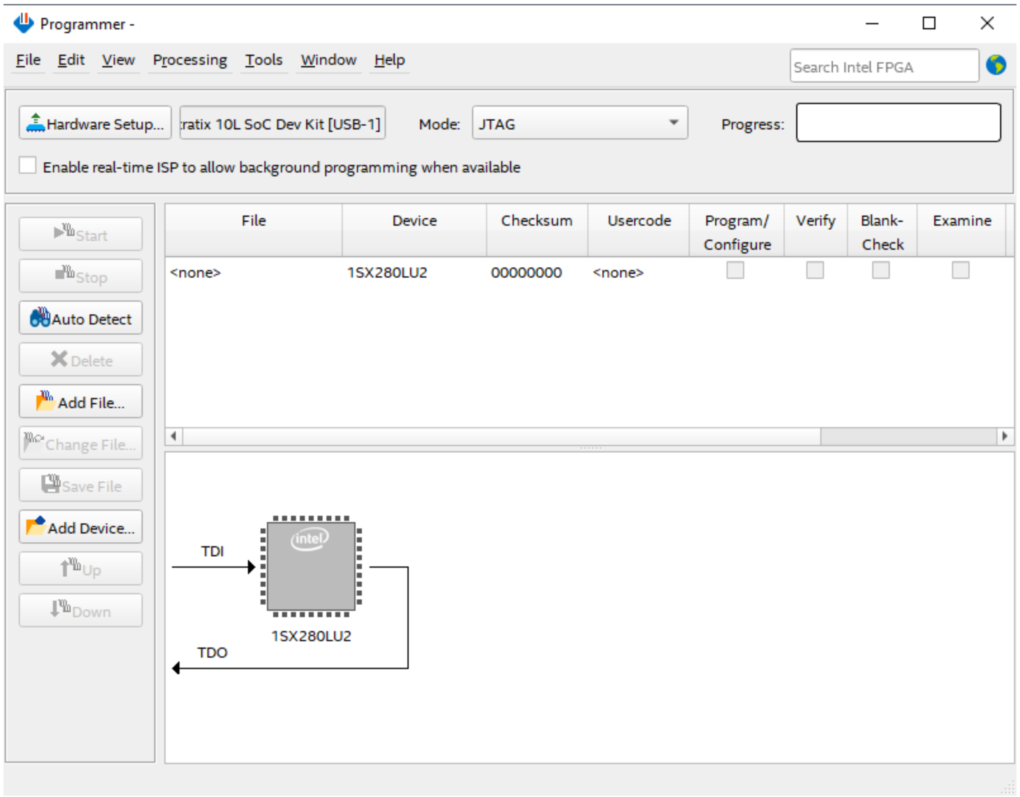
2.2 Programming the Binary using JTAG.

The Programming file(ghrd_1sx280lu2f50e2vg_hps.sof) available in the path "EMEVX_Release1.5_SATA_3.0_Host_AHCI\FPGA\iW-EMEVX-PF-01-R1.0-REL1.5\iW-EMEVX-FF-01-R1.0-REL1.5\iW-EMEVX-SY-01-R1.0-REL1.5\iW-EMEVX-SY-01-R1.0-REL1.5\output_files" can be used to program the Stratix10 Development kit for testing purpose.

To program the device,

- Open Quartus Prime Pro 18.1 tool.

- Go to Tools ➡ Programmer, the below window will appear.



Hardware Setup... ratix 10L SoC Dev Kit [USB-1] Mode: JTAG Progress:

□ Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine
<none>	1SX280LU2	00000000	<none>	□	□	□	□

Figure 2: Quartus Programmer Window

- Click on **Auto Detect** and select the correct part number (**1SX280LU2**).

- Double click **<none>** under file option to load the programming binary(ghrd_1sx280lu2f50e2vg_hps.sof).

- After the file is successfully loaded in the programmer tool, select the check box for **Program/Configure**.

- Click on **Start** to program the FPGA.

Note: For generating (ghrd_1sx280lu2f50e2vg_hps.sof) file from the binary(.sof) generated by the tool, refer the link below:

- https://rocketboards.org/foswiki/Projects/Stratix10SoCConfiguringFPGAfromHPSDesignExample
-
- This generated file then can be used to program the Stratix10 FPGA using JTAG.

2.3 DIP Switch Settings for QSPI Flash.

The Programming file(ghrd_1sx280lu2f50e2vg_hps.sof) for the SATA Project released is flashed to the QSPI Device available on the Stratix10 Development kit.

Default DIP switch settings for programming through JTAG:

- **SW1: - 1: OFF, rest: ON**
- **SW2: - 1: ON, 2: ON, 3: ON, 4: OFF**
- **SW3: - all OFF**
- **SW4: - 1: ON, 2: OFF, 3: OFF, 4: ON**

For accessing the QSPI Flash, DIP switch settings need to be changed to:

- **SW1: - 1: OFF, rest: ON**
- **SW2: - 1: ON, 2: OFF, 3: OFF, 4: OFF**
- **SW3: - all OFF**
- **SW4: - 1: ON, 2: OFF, 3: OFF, 4: ON**

3 Software Development environment setup

3.1 Host Requirements

- A Linux host PC with latest version (ex. Ubuntu version 16.04)
- Root permission on the Development Host.
- Cross compiler package for Stratix 10 development board.

3.2 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
```

3.3 Standalone Compilation

The following steps will help to build the Linux kernel and uboot for Stratix 10 Evaluation Board.

3.3.1 Uboot

- Extract toolchain.tar.gz in to the /opt directory.

```
host@host/<Directory>~$tar -xvzf <Release_folder>/EMEVX_Release1.5_SATA_Host_Controller_IP/SOFTWARE/COMPILER/ gcc-linaro-7.4.1-2019.02-i686_aarch64-linux-gnu.tar.xz  -C  /opt
```

- Create a directory and open the directory in host to build the Linux.

```
host@host~$ mkdir Uboot
```

- Extract linux-socfpga.tar.gz file in to newly created directory.

```
host@host/<Directory>~$tar -xvzf <Release_folder>/EMEVX_Release1.5_SATA_Host_Controller_IP/SOFTWARE/SOURCE_CODE/UBOOT/u-boot-socfpga.tar.gz -C Uboot
```

- Change the working directory to Uboot folder

```
host@host$ cd Uboot
```

Export the environmental variables to cross compile the u-boot and linux for underlying platform.

```
host@host$ export CROSS_COMPILE=aarch64-linux-gnu-
host@host$ export ARCH=arm64
host@host$ export PATH=$PATH:/opt/gcc-linaro-7.4.1-2019.02-i686_aarch64-linux-gnu/bin/
```

- Run below commands to clean,configure and to create uboot binary respectively.

```
host@host$ make mrproper
host@host$ make socfpga_stratix10_defconfig
host@host$ make
```

- Use the below binary file for the SD card image generation as specified in the section Binary Programming

```
~/ u-boot-dtb.img
```

NOTE: In the above commands Replace *<Release folder>* ***with the path in the local PC where Release folder is Downloaded***

3.3.2 Linux kernel

- Create a directory and open the directory in host to build the Linux.

```
host@host~$ mkdir LINUX_KERNEL
```

- Extract linux-socfpga.tar.gz file in to newly created directory.

```
host@host/<Directory>~$tar -xvzf <Release_folder>/EMEVX_Release1.5_SATA_Host_Controller_IP/SOFTWARE/SOURCE_CODE/LINUX_KERNAL/linux-socfpga.tar.gz -C LINUX_KERNEL
```

- Copy the kernel patch file to current directory.

```
host@host/<Directory>~$cp <Release_folder>/EMEVX_Release1.5_SATA_Host_Controller_IP/SOFTWARE/SOURCE_CODE/LINUX_KERNAL/PATCH001-iW-EMEVX-SC-01-R1.0-REL1.0-Linux4.14-Ltsi-iWave-SATA-support.patch
```

- Change the directory to Linux source code directory.

```
host@host/<Directory>~$cd LINUX_KERNEL
```

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

```
host@host/<Directory>~$patch -Np1 < <path to patch file>/ PATCH001-iW-EMEVX-SC-01-R1.0-REL1.0-Linux4.14-Ltsi-iWave-SATA-support.patch
```

- To configure the kernel for Stratix 10 Board, execute the below command.
  host@host/<Directory>~$make s10_iw_defconfig

- To compile the kernel module drivers and kernel image and dts, execute the below commands.
  host@host/<Directory>~$make

- After successful compilation, kernel image (Image) and device tree (.dtb) image will be generated

- Use the below binary files for the SD card image generation as specified in the section Binary Programming
  ~/arch/arm64/boot/Image
  ~/arch/arm64/boot/dts/altera/socfpga_stratix10_socdk.dtb

NOTE: In the above commands Replace *<Release folder>* ***with the path in the local PC where Release folder is Downloaded***

4 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.

4.1 Requirements

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

- Micro SD card reader.
- Host PC (linux).
- Micro SD
- Binary files
  (u-boot.scr, u-boot-dtb.img, Image, socfpga_stratix10_socdk.dtb, rootfs.tar.gz)

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

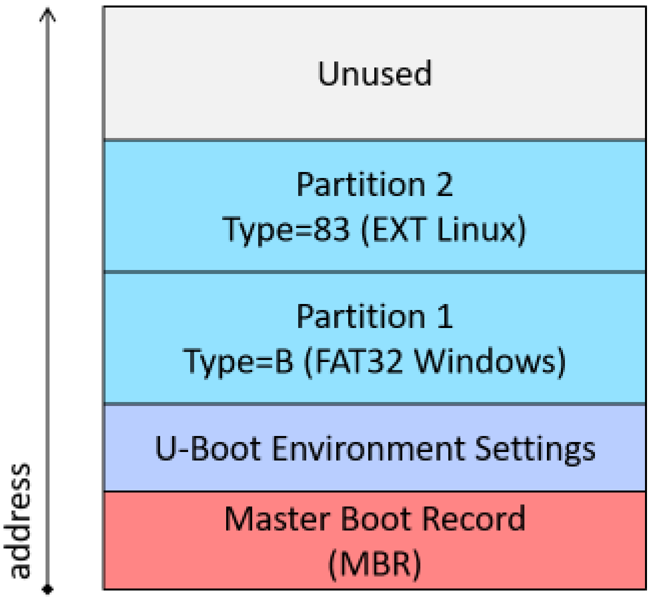


Figure 3: Boot device memory layout

4.2 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 to the newly created directory

- Copy *make_sdimage.py* file to the current directory
  ```
  host@host/<Directory>~$ cp <Release_folder>/EMEVX_Release1.0_SATA_Host_Controller_IP/SOFTWARE/BINARY/make_sdimage.py .
  ```

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

- Change permission of python script
  ```
   host@host/<Directory>~$sudo chmod 777 make_sdimage.py
  ```

- Run the below command to create the SD card image.
  ```
   host@host/<Directory>~$sudo ./make_sdimage.py -n sd_card.img -s 2G -P u-boot.scr,u-boot-dtb.img,socfpga_stratix10_socdk.dtb,Image,num=1,format=vfat,size=500M -P rootfs/*,num=2,format=ext3,size=1G
  ```

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

5 Linux SATA Read Write Testing

5.1 Formatting SSD

- unmount the SSD as shown

  umount /dev/sdx1

  umount /dev/sdx
- Run fdisk command to delete partitions and create new ones

  fdisk /dev/sdx

  Command (m for help):
- Enter m, command option table will be displayed
- Enter p to print partition table
- Enter d to delete the partition.
- Enter the Partition number you want to delete.
- Enter n to create new partition and enter p

  ```
  Partition type:
     p   primary (2 primary, 0 extended, 2 free)
    e   extended
  ```
- Enter 1 as the partition number
- Press enter to take default cylinder size.
- Press enter again to allocate complete memory to primary partition.
- Enter p to cross check Partition Table.
- Enter t to display available file system types to write to partition.
- Enter L to display the table and enter hex code (83) to select the required file system
- Enter w, changes done will be written to the Disk
- Format the partition using the below command

  mkfs.ext3 /dev/sdx1

NOTE: Formatting SSD is not Required if it already contains partitions and valid file system supported by linux.
Please verify the device file using fdisk -l here sdx is given just for refrence

5.2 SATA read and write using the Linux file system.

- Mount the card using the **mount** command as shown below

  mount /dev/sdx1 /mnt/

- Use below commands for read/write a file from SATA Disk.
  mkdir /mnt/sata/
  vim sata.txt (add some contents in the sata.txt file)
  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.

- Use below command for SATA to SATA Transfer read/write testing.
  dd if=/mnt/sata.txt of=/mnt/sata1.txt bs=1M
- Un-mount the partition using the below command
  umount /mnt

6 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.

6.1 Creating SD Card on Linux

The required steps are:

- Copy SD card image file created from Binaries folder.

  ```
  mkdir SDcard
  cd SDcard
  cp <path to SD card image file>/sd_card.img
  ```

- 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.

- Unmount the SD card Partitions before flashing SD with image file.

  ```
  umount  /dev/sdx1
  umount /dev /sdx2
  ```

  NOTE: unmount all the partitions.Here SD card is assumed to be detected as sdx .

  Replace x in sd(x) with the corresponding device file for SD in your Device

- Use dd utility to write the SD image to the SD card:

  ```
  $ sudo dd if=sd_card.img of=/dev/sdx bs=1M
  ```

- Use sync utility to flush the changes to the SD card:

  ```
  $ sudo sync
  ```