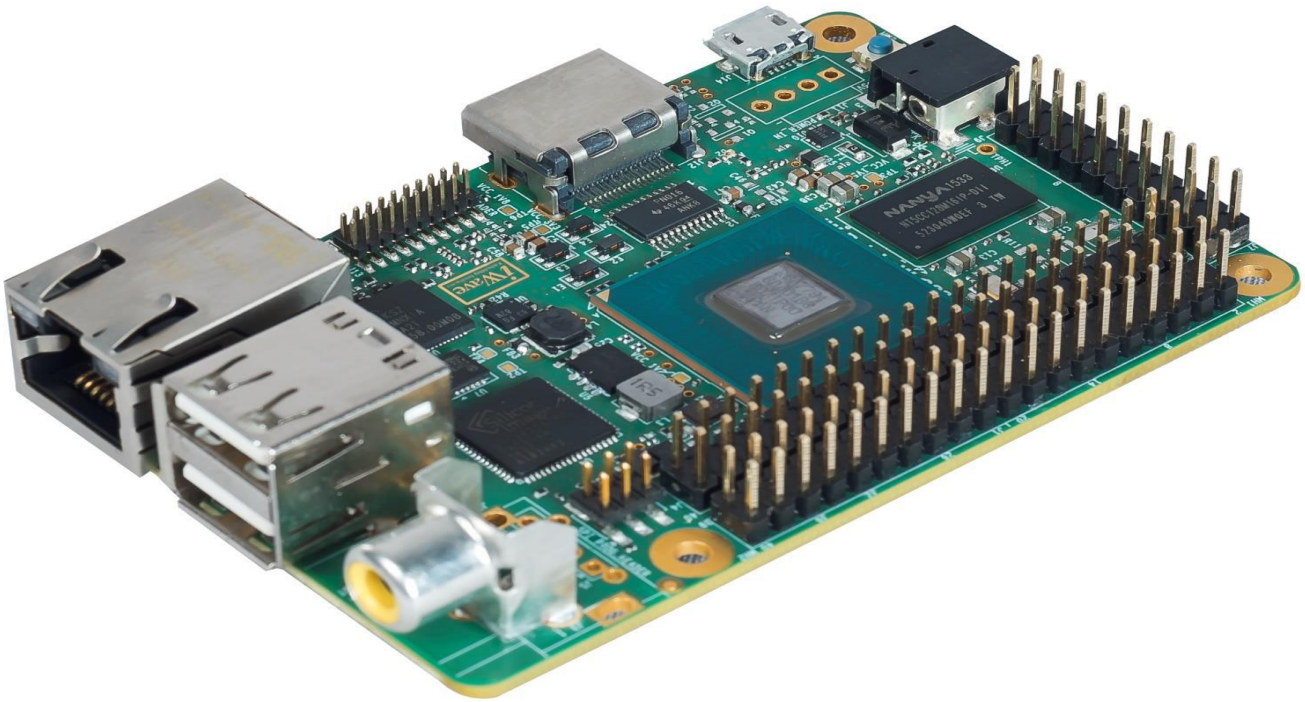
iW-RainboW-G23S

RZ/G1C SBC Development Platform

Linux User Guide



iWave

Embedding Intelligence

Document Revision History

Document Number		iW-PRFCC-UM-01-R3.0-REL1.1-Linux3.10.31
Revision	**Date**	**Description**
1.0	11th Aug 2017	Initial release
1.1	22nd Aug 2017	• Removed Ethernet section in U-Boot • Removed TFTP and NFS boot sections

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.

CPU and other major components used in this product may have several silicon errata associated with it. Under no circumstances, iWave Systems shall be liable for the silicon errata and associated issues.

Trademarks

All registered trademarks, product names mentioned in this publication are the property of their respective owners and used for identification purposes only.

Certification

iWave Systems Technologies Pvt. Ltd. is an ISO 9001:2015 Certified Company.

Warranty & RMA

Warranty support for Hardware: 1 Year from iWave or iWave's EMS partner.
For warranty terms, go through the below web link,
http://www.iwavesystems.com/support/warranty.html

For Return Merchandise Authorization (RMA), go through the below web link,
http://www.iwavesystems.com/support/rma.html

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,

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

Table of Contents

List of Figures

List of Tables

1. INTRODUCTION

1.1 Purpose

The purpose of this document is to help the software engineer to program and test the RZ/G1C SBC Linux development platform 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 U-Boot, Linux Operating systems and related software installed on the RZ/G1C SBC. The Linux BSP is the collection of binary, source code and support files that can be used to create a Linux kernel image and a root file system for RZ/G1C SBC.

1.3 List of Acronyms

The following acronyms will be used throughout this document.

Table 1: Acronyms & Abbreviations

Acronyms	Abbreviations
BSP	Board Support Package
CAN	Controller Area Network
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
DDR3	Double Data Rate 3
eMMC	Enhanced Multi Media Card
HDMI	High-Definition Multimedia Interface
I2C	Inter-Integrated Circuit
Kbps	Kilobits per second
LCD	Liquid Crystal Display
LVDS	Low Voltage Differential Signal
MAC	Media Access Controller
MB	Mega Byte
Mbps	Megabits per sec
MHz	Mega Hertz
MIPI	Mobile Industry CPU Interface
MMC	Multi Media Card
PWM	Pulse Width Modulation
RTC	Real Time Clock
SATA	Serial Advanced Technology Attachment
SD	Secure Digital
SOM	System On Module
SPI	Serial Peripheral Interface

Acronyms	Abbreviations
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
USB OTG	USB On The Go

2. BOARD SUPPORT PACKAGE

2.1 BSP Driver details

This section explains about the features supported, the device driver details and path of the device drivers and device tree details in the RZ/G1C SBC BSP.

2.1.1 BSP Supported Features

Refer the Software release note for the supported features in the BSP.

2.1.2 Driver Source description

This section explains about the device driver details and the corresponding path in the Linux BSP of RZ/G1C SOM.

Table 2: Driver Source Path

File Path	Description
eMMC	
include/linux/mmc/sh_mmcif.h	MMCIF driver header file
drivers/mmc/host/sh_mmcif.c	MMCIF driver source file
SD	
drivers/mmc/host/sh_mobile_sdhi.c	SDHI driver source file
drivers/mmc/host/tmio_mmc.h	SD/SDIO driver header file
drivers/mmc/host/tmio_mmc_pio.c	SD/SDIO driver source file
drivers/mmc/host/tmio_mmc_dma.c	
include/linux/mmc/sh_mobile_sdhi.h	SDHI driver header file
include/linux/mmc/host.h	MMC Subsystem host header file
include/linux/mmc/tmio.h	SD/SDIO driver header file
include/linux/mfd/tmio.h	SDHI driver, SD/SDIO driver header file
UART	
drivers/tty/serial/sh-sci.c	Serial source file
drivers/tty/serial/sh-sci.h	Serial header file
USB (USB OTG as Device)	
drivers/usb/renesas_usbhs/common.c	USB driver function/host driver source/header files
drivers/usb/renesas_usbhs/common.h	
drivers/usb/renesas_usbh/sfifo.c	
drivers/usb/renesas_usbhs/fifo.h	
drivers/usb/renesas_usbhs/mod.c	
drivers/usb/renesas_usbhs/mod.h	
drivers/usb/renesas_usbhs/mod_gadget.c	USB function driver source file
drivers/usb/renesas_usbhs/pipe.c	USB driver function/host driver source/header files
drivers/usb/renesas_usbhs/pipe.h	
drivers/usb/gadget/mass_storage.c	USB gadget mass storage driver source file
include/linux/usb/renesas_usbhs.h	USB function driver header file
include/linux/usbgadget.h	USB gadget driver header file

File Path	Description
USB2.0 Host	
drivers/usb/host/ehci-hcd.c	EHCI host driver source file
drivers/usb/host/ehci-hub.c	
drivers/usb/host/ehci-lpm.c	
drivers/usb/host/ehci-mem.c	
drivers/usb/host/ehci-q.c	
drivers/usb/host/ehci-sched.c	
drivers/usb/host/ehci-sysfs.c	
drivers/usb/host/ehci-pci.c	EHCI driver source file
drivers/usb/host/ehci.h	EHCI header file
drivers/usb/host/ohci-hcd.c	OHCI host driver source file
drivers/usb/host/ohci-hub.c	
drivers/usb/host/ohci-mem.c	
drivers/usb/host/ohci-q.c	
drivers/usb/host/ohci-pci.c	
drivers/usb/host/ohci.h	OHCI header file
drivers/usb/host/xhci.c	xHCI host driver source file
drivers/usb/host/xhci-hub.c	
drivers/usb/host/xhci-mem.c	
drivers/usb/host/xhci-ring.c	
drivers/usb/host/xhci-plat.c	xHCI driver source file
drivers/usb/host/xhci-rcar.c	xHCI driver source file for R-Car H2/M2
drivers/usb/host/xhci.h	xHCI header file
drivers/usb/host/Xhci-rcar.h	xHCI header file for R-Car H2/M2
drivers/usb/phy/phy.c	USB phy driver source file
drivers/usb/phy/phy-rcar-gen2-usb.c	R-Car Gen2 USB phy driver for EHCI/OHCI source file
drivers/pci/host/pci-rcar-gen2.c	R-Car Gen2 Internal PCI driver source file
drivers/phy/phy-core.c	Generic Phy framework
drivers/phy/phy-rcar-gen2.c	R-Car Gen2 USB phy driver for xHCI source file
firmware/UU3DRD1FW_2005L.dlmem	xHCI firmware
include/linux/usb.h	USB driver header file
include/linux/usb/hcd.h	HCD driver header file
Display	
drivers/gpu/drm/rcar-du/rcar_du_crtc.c	CRTC source file
drivers/gpu/drm/rcar-du/rcar_du_crtc.h	CRTC header file
drivers/gpu/drm/rcar-du/rcar_du_drv.c	DRM driver source file
drivers/gpu/drm/rcar-du/rcar_du_drv.h	DRM driver header file
drivers/gpu/drm/rcar-du/rcar_du_encoder.c	DRM encoder source file
drivers/gpu/drm/rcar-du/rcar_du_encoder.h	DRM encoder header file
drivers/gpu/drm/rcar-du/rcar_du_group.c	DRM group source file
drivers/gpu/drm/rcar-du/rcar_du_group.h	DRM group header file
drivers/gpu/drm/rcar-du/rcar_du_hdmicon.c	HDMI connector source file

File Path	Description
drivers/gpu/drm/rcar-du/rcar_du_hdmicon.h	HDMI connector header file
drivers/gpu/drm/rcar-du/rcar_du_kms.c	KMS driver source file
drivers/gpu/drm/rcar-du/rcar_du_kms.h	KMS driver header file
drivers/gpu/drm/rcar-du/rcar_du_plane.c	Plane operation source file
drivers/gpu/drm/rcar-du/rcar_du_plane.h	Plane operation header file
drivers/gpu/drm/rcar-du/rcar_du_regs.h	DU register header file
drivers/gpu/drm/rcar-du/rcar_du_vgacon.c	VGA connector source file
drivers/gpu/drm/rcar-du/rcar_du_vgacon.h	VGA connector header file
drivers/gpu/drm/rcar-du/vsp_du_if.c	vsp-du interface file
drivers/gpu/drm/rcar-du/vsp_du_if.h	vsp-du interface header file
drivers/gpu/drm/rcar-du/vspd_dl.c	vspd display list file
drivers/gpu/drm/rcar-du/vspd_drv.h	vsp driver header file
drivers/gpu/drm/rcar-du/vspd_drv_main.c	vsp driver main file
drivers/gpu/drm/rcar-du/vspd_drv_private.h	vsp driver private header file
include/linux/platform_data/rcar-du.h	DRM driver header file
drivers/gpu/drm/i2c/sii902x.c	SII902x source file
SPI	
drivers/spi/spi.c	SPI core interface source file
drivers/spi/spi-bitbang.c	
drivers/spi/spi-sh-msiof.c	MSIOF driver source file
drivers/mtd/spi-nor/spi-nor.c	SPI NOR Flash driver source file
include/linux/spi/spi.h	SPI core interface header file
include/linux/spi/spi_bitbang.h	
include/linux/spi/sh_msiof.h	MSIOF driver header file
include/linux/mtd/spi-nor.h	SPI NOR Flash driver header file
include/linux/spi/flash.h	
drivers/mtd/devices/m25p80.c	SPI NOR Flash driver source file
drivers/spi/spi-rspi.c	QSPI driver source file
EtherAVB	
drivers/net/ethernet/renesas/ravb_main.c	Gigabit Ethernet driver source file
drivers/net/ethernet/renesas/ravb.h	Gigabit Ethernet driver header file
drivers/net/phy/micrel.c	Phy driver source file
include/linux/micrel_phy.h	Phy driver header file
GPIO	
drivers/gpio/gpio-rcar.c	Source file (device dependence)
drivers/gpio/gpiolib.c	GPIO library source file
arch/arm/include/asm/gpio.h	Header file
include/linux/gpio.h	
VIN TVIN	
drivers/media/platform/soc_camera/rcar_vin.c	Video capture driver file
drivers/media/platform/rcar-dvdec.c	CVBS driver file
include/linux/platform_data/camera_rcar.h	Video capture driver header file

2.1.3 Device tree source description

This section explains about the device tree source code configuration and organization for RZ/G1C Pi SBC. The device tree source codes will be available in below path of the Linux kernel.

~/arch/arm/boot/dts/

Table 3: Device tree source

File Name	Description
r8a7747x_iwg23s.dtsi	This device tree source include file defines the common CPU controllers configuration of RZ/G1C (Dual) processors used in iW-RainboW-G23S platform.
r8a7747x-iwg23s_Pi.dts	This device tree source file is for Renesas R8A77470 (Dual) processors used in iW-RainboW-G23S platforms.

2.2 BSP Yocto Compilation

This section explains procedure and detailed information about compiling the YOCTO for RZ/G1C SBC.

2.2.1 Host Requirements

- A Linux host PC with latest version (ex. Ubuntu version 12.04).
- Root permission on the Development Host.
- Cross compiler package for RZ/G1C SBC.

2.2.2 Host setup

- This document assumes that Ubuntu PC is used. Not a requirement, but the packages may be named differently and the method of installing them may be different.
- The recommended minimum Ubuntu version is 12.04 or later.
- Minimum hard disk space required is about 50 GB and 4GB of RAM.

Note: Earlier version Ubuntu may cause the Yocto Project build setup to fail, because it requires python versions which are available only starting with Ubuntu 12.04.

2.2.3 Host package installation

To get the Yocto Project expected behaviour in a Linux Host Machine, the packages and utilities described below must be installed. An important consideration is the hard disk space required in the host machine. For example, when building on a machine running Ubuntu, the minimum hard disk space required is about 50 GB. It is recommended that at least 120 GB be provided, which is enough to compile any backend.

- Open a terminal window and install the below packages in host PC

```
$ sudo apt-get install sed wget cvs subversion git-core coreutils \
unzip texi2html texinfo libsdl1.2-dev docbook-utils gawk \
python-pysqlite2 diffstat help2man make gcc build-essential \
g++ desktop-file-utils chrpath libgl1-mesa-dev libglu1-mesa-dev \
mercurial autoconf automake groff bc libtool xterm
```

2.2.4 Yocto project setup

To setup the yocto, follow the procedure given below.

- Required files (poky, meta-openembedded, meta-linaro) are downloaded by git clone.

  $ mkdir iwg23s-release-bsp;cd iwg23s-release-bsp

  $ git clone git://git.yoctoproject.org/poky

  $ git clone git://git.openembedded.org/meta-openembedded

  $ git clone git://git.linaro.org/openembedded/meta-linaro.git

- Checkout available version of each git clone.

  $ cd iwg23s-release-bsp/poky

  $ git checkout -b tmp yocto-1.6.1

  $ cd iwg23s-release-bsp/meta-openembedded

  $ git checkout -b tmp dca466c074c9a35bc0133e7e0d65cca0731e2acf

  $ cd iwg23s-release-bsp/meta-linaro

  $ git checkout -b tmp 8a0601723c06fdb75e62aa0f0cf15fc9d7d90167

- The Yocto tar file from deliverables is located in the below path.

  iW-RainboW-G23S-Pi-RX.X-RELX.X-Linux3.10.31-YoctoDaisy_Deliverables/Source-Code/Linux/Yocto/meta-renesas-rzg1c.tar.gz

- Extract the Yocto tar file.

  $ cd ~/<path to iwg23s-release-bsp>/iwg23s-release-bsp/

  $ tar zxf ~/<path to Yocto tar file>/meta-renesas-rzg1c.tar.gz

- The Yocto patch file from deliverables is located in the below path.

  iW-RainboW-G23S-Pi-RX.X-RELX.X-Linux3.10.31-YoctoDaisy_Deliverables/Source-Code/Linux/Yocto/PATCH003-iW-PRFCC-SC-01-RX.X-RELX.X-YoctoDaisy_basic_customization.patch

- To apply the Yocto patch file, execute the below command.

  $ patch -Np1 < <path to Yocto patch file>/PATCH003-iW-PRFCC-SC-01-RX.X-RELX.X-YoctoDaisy_basic_customization.patch

- The copy proprietary script file and Multimedia package directory from deliverables is located in the below path respectively.

  iW-RainboW-G23S-Pi-RX.X-RELX.X-Linux3.10.31-YoctoDaisy_Deliverables/Source-Code/Linux/Yocto/copy_proprietary_softwares.sh

  iW-RainboW-G23S-Pi-RX.X-RELX.X-Linux3.10.31-YoctoDaisy_Deliverables/Source-Code/Linux/Yocto/PKG-DIR

- Copy proprietary software into recipe directory structure.

  $ sh <path to script file>/copy_proprietary_softwares.sh <path to Multimedia package directory>/PKG-DIR

- Execute source command with with oe-init-build-env for setting environment

  $ source poky/oe-init-build-env

- Copy configuration files, bblayers.conf and local.conf for iwg23s platform.

  $ cp -rf ../meta-renesas/meta-rzg1/templates/iwg23s/mmp/bblayers.conf ./conf

- To build Wayland for iWG23S platform, overwrite the local.conf file with wayland configuration files by executing the below command

  $ cp -rf ../meta-renesas/meta-rzg1/templates/iwg23s/mmp/local-wayland.conf ./conf/local.conf

2.2.5 Yocto build

This section provides the detailed information and process for building the Yocto binaries.

- Open a terminal window and change the directory as follows.

  $ cd iwg23s-release-bsp

  $ source poky/oe-init-build-env

- To compile the yocto file system for Wayland, execute the below command.

  $ bitbake core-image-weston

- After the successful compilation the binaries will be placed in below path.

  ~/<path to iwg23s-release-bsp>/iwg23s-release-bsp/build/tmp/deploy/images/iwg23s/

- The binary files are listed below

  core-image-weston-iwg23s-<date & time>.rootfs.tar.bz2

  uImage--<revision>-iwg23s-<date & time>.bin

  uImage--<revision>-r8a7747x-iwg23s_Pi -<date & time>.dtb

  u-boot-iwg23s-<revision>.bin

- Rename the binary files as mentioned below

  core-image-weston-iwg23s-<date & time>.rootfs.tar.bz2 *- -> rootfs.tar.bz2*

  uImage--<revision>-iwg23s-<date & time>.bin *- -> uImage*

  uImage--<revision>-r8a7747x-iwg23s_Pi-<date & time>.dtb *- - > r8a7747x-iwg23s_Pi.dtb*

  u-boot-iwg23s-<revision>.bin *- -> u-boot.bin*

- Refer the **BINARY PROGRAMMING** section to update the Linux kernel binary.

2.2.6 U-boot

This section provides the detailed information and process for building the u-boot binary image.

- Open a terminal window and change the directory to yocto setup path.

  $ cd ~/<path to iwg23s-release-bsp>/iwg23s-release-bsp/

- To compile the u-boot, execute the below command.

  $ bitbake u-boot-iwg23s

- To re-build u-boot, execute the below commands.

  $ bitbake -f -c compile u-boot-iwg23s

- After the successful compilation the binaries will be placed in below path.

  ~/<path to iwg23s-release-bsp>/iwg23-release-bsp/build/tmp/work/cortexa7hf-vfp-neon-poky-linux-gnueabi/u-boot-iwg23s/v2013.01.01+gitAUTOINC+aa28b1d7a4-r0/git/

- The binary files are listed below

  u-boot.bin

- Refer the **BINARY PROGRAMMING** section to update the u-boot binary.

2.2.7 Linux kernel

This section provides the detailed information and process for building the kernel images.

- Open a terminal window and change the directory to yocto setup path.

  $ cd ~/<path to iwg23s-release-bsp>/iwg23s-release-bsp/

- To compile the Linux kernel, execute the below command.

  $ bitbake linux-iwg23s

- To change the Linux kernel configuration, execute the below command.

  $ bitbake -c menuconfig linux-iwg23s

- To re-build Linux kernel from scratch, execute the below command.

  $ bitbake -f -c compile linux-iwg23s

- After the successful compilation the binaries will be placed in below path.

  ~/<path to iwg23s-release-bsp>/iwg23s-release-bsp/build/tmp/work/iwg23s-poky-linux-gnueabi/linux-iwg23s/3.10+git9c8ffcd274f74c4c4efecb307157ee4c6df54fa4-r0/git/arch/arm/boot/S

- The binary files are listed below

  uImage

  dts/r8a7747x-iwg23s_Pi.dtb

- Refer the **BINARY PROGRAMMING** section to update the Linux kernel binary.

2.2.8 Cross-compiler build

This section provides the detailed information and process for building the cross-compiler and this cross-compiler can be used for standalone compilation of U-boot and Linux kernel. Make sure to do Yocto project setup before cross compiler build. Refer the **Yocto project setup** section to setup the yocto.

- Open a terminal window and change the directory as follows.

  $ cd iwg23s-release-bsp

- Execute source command with oe-init-build-env for setting environment.

  $ source poky/oe-init-build-env

- Modify **SDK_MACHINE** description on iwg23s-release-bsp/build/conf/local.conf accordingly with the host PC architecture.

  SDKMACHINE ?= "i686" (or " x86_64")

- To generate the package, execute the below command.

  $ bitbake meta-toolchain

- To install toolchain on Host PC, execute the below command.

  $ sudo tmp/deploy/sdk/poky-eglibc-x86_64(or i686)- cortexa7hf-vfp-neon-toolchain-1.6.1.sh

- During the toolchain installation, confirmation is required while below message is displayed to proceed

  [sudo] password for (INSTALL person): *(password of your account)*

  Enter target directory for SDK (default: /opt/poky/poky1.6.1): *(just a return)*

  Extracting SDK...done

  Setting it up...done

  SDK has been successfully set up and is ready to be used

- After successful installation of cross-compiler, the cross compiler will be available in below path.

  /opt/poky/1.6.1

2.3 BSP Standalone compilation

This section explains procedure and detailed information about compiling the U-boot and Kernel for RZ/G1C PI SBC without Yocto. The Loader should be compiled without Yocto. The following steps will help to build the Loader and to build the standalone u-boot and linux kernel for RZ/G1C PI SBC without Yocto.

- Before compiling the source code, cross compiler should be installed to "/opt" directory of host machine if it is not present.
- Refer the **Cross-compiler build** section to install the cross-compiler.

2.3.1 Loader

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

  host@host~$ mkdir <directory_name>

  host@host~$ cd <directory_name>

- Un-tar the downloaded loader-iwg23s.tar.gz file in to newly created directory.

  host@host/<Directory>~$ tar -xvzf /<path_to_iW-RainboW-G23S-Pi-RX.X-RELX.X-Linux3.10.31-YoctoDaisy_Deliverables>/iW-RainboW-G23S-Pi-RX.X-RELX.X-Linux3.10.31-YoctoDaisy_Deliverables/Source-Code/Loader/loader-iwg23s.tar.gz

  host@host/<Directory>~$ sync

- Copy the Loader patch file to current directory.

  host@host/<Directory>~$ cp /<path_to_iW-RainboW-G23S-Pi-RX.X-RELX.X-Linux3.10.31-YoctoDaisy_Deliverables>/iW-RainboW-G23S-Pi-RX.X-RELX.X-Linux3.10.31-YoctoDaisy_Deliverables/Source-Code/ Loader/PATCH001-iW-PRFCC-SC-01-RX.X-RELX.X-Linux3.10.31_Loader_basic_customization.patch .

  host@host/<Directory>~$ sync

- Change the directory to loader source code directory.

  host@host/<Directory>~$ cd <path_to_loader-iwg23s>/loader-iwg23s

- To apply the patch file, execute the below command.

  host@host/<Directory>/loader-iwg23s~$ patch -Np1 < ../ PATCH001-iW-PRFCC-SC-01-RX.X-RELX.X-Linux3.10.31_Loader_basic_customization.patch

- Export the architecture, cross compiler and tool chain path.

  host@host/<Directory>/loader-iwg23s~$ export ARCH=arm

  host@host/<Directory>/loader-iwg23s~$ export PATH=$PATH:/opt/poky/1.6.1/sysroots/<processor>-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi

  host@host/<Directory>/loader-iwg23s~$ export CROSS_COMPILE=arm-poky-linux-gnueabi-

  host@host/<Directory>/loader-iwg23s~$ export DDR=DDR3

- Change the directory to 'src'.

  host@host/<Directory>/loader-iwg23s~$ cd src

- To compile the Loader image, execute the below commands.

  host@host/<Directory>/loader-iwg23s/src~$ make

- After successful compilation, Loader (iw_rainbow_G23S_SPI_loader_v020_DDR3_E6300000_V100.bin) will be created in the below path respectively.

  ~/output/iW_Rainbow_G23S_SPI_LOADER_V020_DDR3_E6300000_V100.bin

- Refer the **BINARY PROGRAMMING** section to update the Loader binary.

2.3.2 U-Boot

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

  host@host~$ mkdir <directory_name>

  host@host~$ cd <directory_name>

- Un-tar the downloaded u-boot-iwg23s.tar.gz file in to newly created directory.

  host@host/<directory>~$ tar -xvzf /<path to iW-RainboW-G23S-Pi-RX.X-RELX.X-Linux3.10.31-YoctoDaisy_Deliverables>/iW-RainboW-G23S-Pi-RX.X-RELX.X-Linux3.10.31-YoctoDaisy_Deliverables/Source-Code/U-Boot/u-boot-iwg23s.tar.gz

- Copy the u-boot patch file to current directory.

  host@host/<directory>~$ cp /<path to iW-RainboW-G23S-Pi-RX.X-RELX.X-Linux3.10.31-YoctoDaisy_Deliverables>/iW-RainboW-G23S-Pi-RX.X-RELX.X-Linux3.10.31-YoctoDaisy_Deliverables/Source-Code/U-Boot/PATCH002-iW-PRFCC-SC-01-RX.X-RELX.X-Linux3.10.31_UBoot_basic_customization.patch .

- Change the directory to u-boot source code directory.

  host@host/<directory>~$ cd /<path to u-boot source>/u-boot-iwg23s

- To apply the patch file, execute the below command.

  host@host/<directory>/u-boot-iwg23s~$ patch -Np1 < ../PATCH002-iW-PRFCC-SC-01-RX.X-RELX.X-Linux3.10.31_UBoot_basic_customization.patch

- To export the Cross Compiler and tool chain path, execute the below command.

  host@host/<directory>/u-boot-iwg23s~$ export ARCH=arm

  host@host/<directory>/uboot-iwg23s~$ export PATH=$PATH:/opt/poky/1.6.1/sysroots/<processor>-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi

  host@host/<Directory>/u-boot-iwg23s~$ export CROSS_COMPILE=arm-poky-linux-gnueabi-

- To configure for iWave-G23S-Q7 platform, execute the below command.

  host@host/<directory>/u-boot-iwg23s~$ make iwg23s_config

- To compile the u-boot source code, execute the below command.

  host@host/<directory>/u-boot-iwg23s~$ make

- After successful compilation, boot loader image (u-boot.bin) will be created in below path.

  ~/u-boot-iwg23s/u-boot.bin

- Refer the **BINARY PROGRAMMING** section to update the u-boot binary.

2.3.3 Linux kernel

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

  host@host~$ mkdir <directory_name>

  host@host~$ cd <directory_name>

- Un-tar the downloaded linux-iwg23s.tar.gz file in to newly created directory.

  host@host/<Directory>~$ tar -xvzf /<path_to_ iW-RainboW-G23S-Pi-RX.X-RELX.X-Linux3.10.31-YoctoDaisy_Deliverables>/iW-RainboW-G23S-Pi-RX.X-RELX.X-Linux3.10.31-YoctoDaisy_Deliverables/Source-Code/Linux/Kernel/linux-iwg23s.tar.gz

  host@host/<Directory>~$ sync

- Copy the kernel patch file to current directory.

  host@host/<Directory>~$ cp /<path_to_ iW-RainboW-G23S-Pi-RX.X-RELX.X-Linux3.10.31-YoctoDaisy_Deliverables>/iW-RainboW-G23S-Pi-RX.X-RELX.X-Linux3.10.31-YoctoDaisy_Deliverables/Source-Code/Linux/Kernel/PATCH003-iW-PRFCC-SC-01-RX.X-RELX.X-Linux3.10.31_Kernel_basic_customization.patch .

- Change the directory to Linux source code directory.

  host@host/<Directory>~$ cd <path_to_linux-iwg23s>/linux-iwg23s

- To apply the patch file, execute the below command.

  host@host/<Directory>/linux-iwg23s~$ patch -Np1 < ../PATCH003-iW-PRFCC-SC-01-RX.X-RELX.X-Linux3.10.31_Kernel_basic_customization.patch

- Export the architecture, cross compiler and tool chain path.

  host@host/<Directory>/linux-iwg23s~$ export ARCH=arm

  host@host/<Directory>/linux-iwg23s~$ export PATH=$PATH:/opt/poky/1.6.1/sysroots/<processor>-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi

  host@host/<Directory>/linux-iwg23s~$ export CROSS_COMPILE=arm-poky-linux-gnueabi-

- To configure the kernel for RZ/G1C PI SBC, execute the below command.

  host@host/<Directory>/linux-iwg23s~$ make iw_rainbowg23s_defconfig

- To compile the kernel module drivers and kernel image, execute the below commands.

  host@host/<Directory>/linux-iwg23s~$ make;make uImage LOADADDR=0x40008000

- After successful compilation, kernel image (uImage) and device tree (.dtb) will be created in the below path respectively.

  ~/linux-iwg23s/arch/arm/boot/uImage

  ~/linux-iwg23s/arch/arm/boot/dts/r8a7747x-iwg23s_Pi.dtb

- Refer the **BINARY PROGRAMMING** section to update the Linux kernel binary.

Note: While applying the kernel patch, rejects to file include/dt-bindings/clock/r8a7747x-clock.h can be ignored.

2.4 BSP Customization

This section explains the information about steps to customize the platform devices information for customer specific devices and to customize the filesystems for other user configurations.

2.4.1 I2C device

This section describes how to set up the I2C device in the device tree. Here, the touch is taken as reference and it is connected in I2C2 bus.

- Add the pinctrl for the I2C2 bus as mentioned below.

```
i2c2 {
        pinctrl-0 = <&i2c2_pins>;
        pinctrl-names = "default";
        status = “okay”;
        ft5x06@38 {
         compatible = "focal,ft5x06";
         reg = <0x38>;
         interrupt-parent = <&gpio2>;
         interrupts = <12 GPIO_ACTIVE_HIGH>;
         int-gpio = <&gpio2 12 GPIO_ACTIVE_HIGH>;
      };
};
```

- Add the pfc,I2C2 pins as mentioned below

```
&pfc {
    pinctrl-0 = <&du_pins &du0_pins &usb0_pins &usb1_pins >;
    pinctrl-names = "default";

    i2c2_pins: i2c2 {
        renesas,groups = "i2c2";
        renesas,function = "i2c2";
};
```

2.4.2 UART

This section describes how to set up the flow control for UART in the device tree. Here, the UART4 is taken as reference and it is connected to hscif1 controller.

- To enable Hardware Flow Control, add "ctsrts" property to the uart node in device tree as mentioned below.

```
&hscif1 {
    pinctrl-0 = <&hscif1_pins>;
    pinctrl-names = "default";
    ctsrts;
    status = "okay";
};
```

2.4.3 GPIO

This section describes how to set up the GPIO functionality for an IOMUX. Here, the GPIO Pin 5-29 is taken as reference.

- All the iomux pins are configured as GPIO functionality by default. In case if other iomux functionality is added for the pin, then remove mux configuration from pfc node in device tree.
- To acquire the gpio pin from device tree, include the code as mentioned below.

```
#include <linux/gpio.h>
#include <linux/of_gpio.h>
{
struct device_node *np;
int gpio;
np = of_find_node_by_path("/gpio@e6055000");    // base address of GPIO5 Port.
gpio = of_get_gpio(np, 29); // GPIO Pin 29 is acquired in a gpio variable.
}
```

- To get the base address of GPIO port to mention in of_find_node_by_path(), refer the arch/arm/boot/dts/r8a7747x_iwg23s.dtsi file.

2.4.4 USB OTG Host Configuration

- To enable the USB OTG as Host, deselect “Renesas USBHS Controller” in make menuconfig and compile again.

Device Drivers --->

USB support --->

USB Gadget Support --->

USB Peripheral Controller -->

<> Renesas USBHS controller

2.4.5 Yocto Package

This section describes how to add an extra package in Yocto filesystem.

- To add an package in Yocto filesystem, add a line to conf/local.conf file in Yocto.

IMAGE_INSTALL_append = " package"

- Make sure to include a space before the package name.
- Refer **BSP Yocto Compilation** to compile the Yocto with package added.

3. BINARY PROGRAMMING

This section explains procedure and detailed information about programming the binaries into boot device of RZ/G1C SBC. The below figure shows the minimum memory requirement for partitioning the boot device. The loader and u-boot binaries are programmed to SPI device and kernel image and filesystem are programmed to eMMC by default.

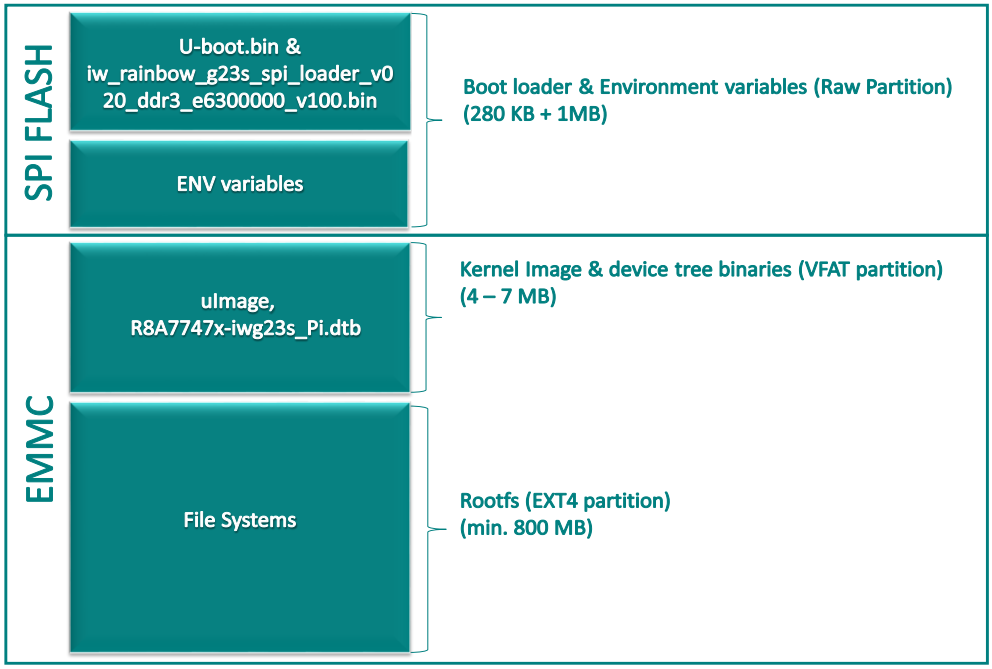


Figure 1: Boot device memory layout

3.1 JTAG Programming

This section explains the step by step procedure to flash the binaries into RZ/G1C SBC through JTAG Debugger.

3.1.1 Requirements

To program the binaries into RZ/G1C SBC through JTAG Debugger, following Items are required:

- JTAG Debugger
- Power Supply
- RZ/G1C development platform.

Note: Here PEEDI JTAG debugger is taken as example. (http://www.ronetix.at/peedi.html) and the contents of Configuration file required for PEEDI JTAG debugger to boot RZ/G1C is mentioned in APPENDIX.

3.1.2 Booting Via JTAG

- Copy the JTAG debugger configuration file and the binary files uboot (uboot.bin), loader (iw_rainbow_g23s_spi_loader_v020_ddr3_e6300000_v100.bin) required for booting RZ/G1C SBC into the SD card, Insert this SD card into JTAG Debugger.
- The prebuilt binaries are available in the deliverables in the below path.

  iW-RainboW-G23S-SBC-RX.X-RELX.X-Linux3.10.31-YoctoDaisy_Deliverables/Binaries/
- Rename the loader binary iw_rainbow_g23s_spi_loader_v020_ddr3_e6300000_v100.bin to loader.bin.
- Open minicom for JTAG Debugger and Target board separately in host PC.
- Power on the JTAG Debugger and then target board.
- Once the JTAG Debugger is booted, set the following parameters in JTAG terminal to boot the target board.
- To set the supervisor mode interrupts, execute the below command.

  RZ/G1x> set cpsr 0x1D3
- To set the stack pointer for loader, execute the below command.

  RZ/G1x> set sp 0xE6303D00
- To set the program counter for loader, execute the below command

  RZ/G1x> set pc 0xE6300000
- To load loader binary image into memory from SD Card, execute the below command

  RZ/G1x > memory load card://loader.bin bin 0xE6300000

  RZ/G1x > go

  RZ/G1x > halt
- To set the supervisor mode interrupts, execute the below command.

  RZ/G1x > set cpsr 0x1D3
- To set the stack pointer for u-boot, execute the below command.

  RZ/G1x > set sp 0x7003FFFC
- To set the program counter for u-boot, execute the below command

  RZ/G1x > set pc 0x70000000
- To load loader.bin image into memory from SD Card, execute the below command

  RZ/G1x > memory load card://u-boot.bin bin 0x70000000

  RZ/G1x > go
- Now, target board will boot and boot prints will appear in the target board console.
- Stop in u-boot command prompt by pressing any key.
- Once command prompt appears, execute the below command in JTAG board to halt the target board core.

  RZ/G1x > halt

3.1.3 U-Boot Programming

- Boot the target through JTAG debugger as mentioned in **Booting Via JTAG** section
- To load U-boot and Loader into external RAM of target board for further programming to SPI, execute the below commands.

  RZ/G1x> memory load card://u-boot.bin bin 0x50008000

  RZ/G1x> memory load card://loader.bin bin 0x50000000

  RZ/G1x> go

- Target board will still be alive with command prompt.
- To erase the SPI flash in target board, execute the below command.

  iWave-G23S> sf probe

  iWave-G23S> sf erase 0x0 0x200000

- To copy loader and U-boot to SPI flash in target board console, execute the below command.

  iWave-G23S> sf write 0x50000000 0x0 <loader size in hex>

  iWave-G23S> sf write 0x50008000 0x20000 <Uboot size in hex>

  Example:

  iWave-G23S> sf write 0x50000000 0x0 401c

  iWave-G23S> sf write 0x50008000 0x20000 3bcdc

- Hence, the u-boot and loader is programmed to SPI flash through JTAG Debugger.

3.2 Manual Programming

This section explains the step by step procedure to program the Linux binaries into RZ/G1C PI SBC manually through a SD Card. Refer this section only if any of the loader and u-boot binaries are already programmed and bootable from SPI flash.

3.2.1 Requirements

To program the binaries into RZ/G1C SBC, following Items are required:

- SD card (Micro SD)
- Card reader for manual binary programming
- Host PC (Linux) for manual binary programming

3.2.2 SD Card Preparation

- Prepare a bootable SD card by referring the **Bootable Micro SD** Card section.
- Create a folder “Binaries” in the windows partition of the SD card.
- Copy the binaries mentioned below to the Binaries folder in the windows partition of the SD card.

  iW_Rainbow_G23S_SPI_LOADER_V020_DDR3_E6300000_V100.bin

  u-boot.bin

  uImage

  r8a7747x-iwg23s_Pi.dtb

  rootfs.tar.bz2
- The prebuilt binaries are available in the deliverables in the below path.

  iW-RainboW-G23S-Pi-RX.X-RELX.X-Linux3.10.31-YoctoDaisy_Deliverables/Binaries/
- Insert the SD card into the board and boot the board.
- Change the boot arguments to SD boot and boot from SD card. Refer the **Environment variables settings** section to change the boot arguments to SD card.
- Boot Linux from the SD card.

3.2.3 eMMC Partition

This section describes the steps to partition an eMMC to program the binaries. Refer this section only if the eMMC is not partitioned.

- Unmount if /dev/mmcblk1 is mounted in any mount point. eMMC should not be mounted while partitioning.

  *$ umount /media/mmcblk1**

- Start partitioning using fdisk command.

  $ fdisk /dev/mmcblk1

- After running fdisk, it will change shell prompt to

  Command (m for help):

- Press 'p' to view already existing partitions. Delete all existing partitions using command ’d’. Enter individual partitions like 1, 2, 3, etc until all the partitions are deleted. Once all the partitions are deleted, the below message gets displayed.

  Command (m for help): d

  No partition is defined yet!

- Press 'u' to change the unit to cylinder.

  Command (m for help): u

- Press 'n' to create new partition (going to create first partition).

  Command (m for help): n

  Command action

  e extended

  p primary partition (1-4)

- Press 'p' to create primary partition. Give 1 as partition number. Then give first cylinder as ‘7’(Based on the total cylinders we have to change the size) and Last cylinder as half of displayed size (e.g. below case 1038 is displayed, give approximate half size, 512)

  p

  Partition number (1-4): 1

  First cylinder (1-1038, default 1): 7

  Last cylinder, +cylinders or +size{K,M,G} (1-1038, default 1038): 512

- Press 'n' to create new partition (going to create second partition).

  Command (m for help): n

  Command action

  e extended

  p primary partition (1-4)

- Press 'p' to create primary partition. Give 2 as partition number. Just press enter without any arguments for First and Last cylinder. Because, First & Last cylinder locations will be displayed from end of 1st partition to end of disk.

  p

  Partition number (1-4): 2

  First cylinder (513-1038, default 513): 513

  Last cylinder, +cylinders or +size{K,M,G} (513-1038, default 1038): press ENTER

- Assign file system to created partitions. Partition 1 as FAT16 and partition 2 as LINUX.

  Command (m for help): t

  Partition number (1-4): 1

  Hex code (type L to list codes): 6

  Changed system type of partition 1 to 6 (FAT16)

  Command (m for help): t

  Partition number (1-4): 2

  Hex code (type L to list codes): 83

- List out partition types in eMMC. Press 'p' to view. Below message will be displayed

  Command (m for help): p

  Disk /dev/mmcblk2: 1021 MB, 1021837312 bytes

  31 heads, 62 sectors/track, 1038 cylinders

  *Units = cylinders of 1922 * 512 = 984064 bytes*

  Disk identifier: 0x00000000

Device Boot	*Start*	*End*	*Blocks*	*Id*	*System*
/dev/mmcblk1p1	*7*	*512*	*492001*	*6*	*FAT16*
/dev/mmcblk1p2	*513*	*1038*	*505486*	*83*	*Linux*

- Now the partitions are created as above. Save these changes by pressing 'w'.

  Command (m for help): w

- Again make sure both the partitions are unmounted.

  $ umount /dev/mmcblk1p1

  $ umount /dev/mmcblk1p2

- Now format both the partitions. Partition 1 as VFAT i.e windows partition and 2nd partition as EXT4.

  $ mkfs.vfat /dev/mmcblk1p1

  $ mkfs.ext4 /dev/mmcblk1p2

- Reboot the board and now eMMC is ready to be programmed.

3.2.4 U-Boot Programming

- Boot Linux from the SD card.
- Make sure that SD card is mounted and all the binaries to be programmed are present in the SD card, by listing the contents of Binaries folder in SD card.
- To list the contents of Binaries folder in SD card, execute the below command.

  root@iWave-G23S~$ ls <mount_directory>/Binaries/

  Example:

  root@iWave-G23S~$ ls /media/mmcblk0p1/Binaries/ (For Micro SD)

- Change the directory to the binaries folder

  root@iWave-G23S~$ cd /<mount_directory>/Binaries/

- Erase the SPI flash

  root@iWave-G23S/<mount_directory>~$ flash_eraseall /dev/mtd0

- Program the loader to SPI Flash

  root@iWave-G23S/<mount_directory>~$ dd if=iW_Rainbow_G23S_SPI_LOADER_V020_DDR3_E6300000_V100.bin of=/dev/mtdblock0 bs=1

  root@iWave-G23S/<mount_directory>~$ sync

- Program the u-boot binary to SPI flash

  root@iWave-G23S/<mount_directory>~$ dd if=u-boot.bin of=/dev/mtdblock0 bs=1k seek=128

  root@iWave-G23S/<mount_directory>~$ sync

3.2.5 Kernel Programming

- Boot Linux from the SD card.
- Make sure that SD card is mounted and all the binaries to be programmed are present in the SD card, by listing the contents of Binaries folder in SD card.
- To list the contents of Binaries folder in SD card, execute the below command.

  root@iWave-G23S~$ ls <mount_directory>/Binaries/

  Example:

  root@iWave-G23S~$ ls /media/mmcblk0p1/Binaries/ (For Micro SD)

- Change the directory to the binaries folder

  root@iWave-G23S~$ cd /<mount_directory>/Binaries/

- Copy the uImage and dtb files to windows partition of eMMC.

  root@iWave-G23S/<mount_directory>~$ cp -rf uImage /media/mmcblk1p1

  root@iWave-G23S/<mount_directory>~$ cp -rf r8a7747x-iwg23s_Pi.dtb /media/mmcblk1p1

- Remove the contents of the linux partition and untar the rootfs into linux partition of eMMC.

  *root@iWave-G23S/<mount_directory>~$ rm -rf /media/mmcblk1p2/**

  root@iWave-G23S/<mount_directory>~$ tar -xvf rootfs.tar.bz2 -C /media/mmcblk1p2/

  root@iWave-G23S/<mount_directory>~$ sync

- Reboot the board

  root@iWave-G23S/<mount_directory>~$ reboot

3.3 Bootable Micro SD Card

This section explains the step by step procedure to flash the binaries into micro SD card manually.

3.3.1 Requirements

To program the binaries in SD card for RZ/G1C SBC, following Items are required:

- SD card (Micro SD)
- Card reader for manual binary programming
- Host PC (Linux) for manual binary programming

3.3.2 SD Card Partition

This section describes the steps to partition an SD card to program the binaries and boot the RZ/G1C SBC. Refer this section only if a new Micro SD card or SD card is used.

- Insert SD card using SD card reader to the PC. Execute mount command to see the attached nodes and mount points.

  $ mount
- SD card may attach to the dev nodes either sdb/sdc/sdd/sde in Host PC. Assume the SD card is attached to /dev/sdb node.
- Unmount if /dev/sdb is mounted in any mount point. SD card should not be mounted while partitioning.

  $ umount /dev/sdb
- Start partitioning using fdisk command.

  $ sudo fdisk /dev/sdb
- After running fdisk, it will change shell prompt to

  Command (m for help):
- Press 'p' to view already existing partitions. Delete all existing partitions using command ’d’. Enter individual partitions like 1, 2, 3, etc until all the partitions are deleted. Once all the partitions are deleted, the below message gets displayed.

  Command (m for help): d

  No partition is defined yet!
- Press 'u' to change the unit to cylinder.

  Command (m for help): u

- Press 'n' to create new partition (going to create first partition).

  Command (m for help): n

  Command action

  e extended

  p primary partition (1-4)

- Press 'p' to create primary partition. Give 1 as partition number. Then give first cylinder as '7'(Based on the total cylinders we have to change the size) Because first 7 cylinders is for U-Boot.bin purpose and Last cylinder as half of displayed size (e.g. below case 1038 is displayed, give approximate half size, 512)

  p

  Partition number (1-4): 1

  First cylinder (1-1038, default 1): 7

  Last cylinder, +cylinders or +size{K,M,G} (1-1038, default 1038): 512

- Press 'n' to create new partition (going to create second partition).

  Command (m for help): n

  Command action

  e extended

  p primary partition (1-4)

- Press 'p' to create primary partition. Give 2 as partition number. Just press enter without any arguments for First and Last cylinder. Because, First & Last cylinder locations will be displayed from end of 1st partition to end of disk.

  p

  Partition number (1-4): 2

  First cylinder (513-1038, default 513): 513

  Last cylinder, +cylinders or +size{K,M,G} (513-1038, default 1038): press ENTER

- Assign file system to created partitions. Partition 1 as FAT16 and partition 2 as LINUX.

  Command (m for help): t

  Partition number (1-4): 1

  Hex code (type L to list codes): 6

  Changed system type of partition 1 to 6 (FAT16)

  Command (m for help): t

  Partition number (1-4): 2

  Hex code (type L to list codes): 83

- List out partition types in SD. Press 'p' to view. Below message will be displayed

```
Command (m for help): p
Disk /dev/sdb: 1021 MB, 1021837312 bytes
31 heads, 62 sectors/track, 1038 cylinders
Units = cylinders of 1922 * 512 = 984064 bytes
Disk identifier: 0x00000000
Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               7         512      492001    6  FAT16
/dev/sdb2             513        1038      505486   83  Linux
```

- Now the partitions are created as above. Save these changes by pressing 'w'.

```
Command (m for help): w
```

- Again make sure both the partitions are unmounted.

```
$ umount  /dev/sdb1
$ umount  /dev/sdb2
```

- Now format both the partitions. Partition 1 as VFAT (windows) partition and 2nd partition as EXT4 (Linux).

```
$ sudo mkfs.vfat /dev/sdb1
$ sudo mkfs.ext4 /dev/sdb2
```

- Now SD card is ready to use.
- Remove the SD card and insert again then the respective partitions can be viewed by the below command.

```
$ mount
```

3.3.3 SD Card Programming

- Insert SD card using SD card reader to the PC. Execute mount command to see the attached nodes and mount points.

  $ mount

- SD card may attach to the dev nodes either sdb/sdc/sdd/sde. Assume the SD card is attached to /dev/sdb node.
- Copy uImage and .dtb files into SD card windows partition.

  $ cp /<path_to_uImage>/uImage /media/<mount_point_of_sdcard VFAT partition>

  $ cp /<path_to_dtb files>/r8a7747x-iwg23s_Pi.dtb /media/<mount_point_of_sdcard VFAT partition>

- Untar the tar file rootfs.tar.bz2 inside the SD card Linux partition.

  $ sudo tar -xvjf /<path_to_rootfs.tar.gz>/rootfs.tar.bz2 -C /media/<mount_point_of_sdcard EXT4 partition>

  $ sync

- Unmount the SD card from the host PC.

  $ umount /media/<mount point sdcard windows partition>

  $ umount /media/<mount point sdcard Linux partition>

- Now the bootable SD Card is ready to be used in RZ/G1C SBC to boot the Linux.

4. U-BOOT TESTING AND BOOT CONFIGURATION

This section explains about testing the peripherals in u-boot level and also loading the Linux OS from different devices for RZ/G1C SBC. The RZ/G1C SBC boots u-boot and loader image from the SPI flash. In U-Boot, the supported devices are,

- RAM
- SD/MMC
- SPI NOR flash
- USB

Open the HyperTerminal on PC/Laptop with the following settings.

Baud rate : 115200Bps

Data bits : 8

Parity : None

Stop bits : 1

Flow control : None

4.1 Basic commands

- To display the platform information, execute the below command and the platform information will be displayed in command prompt as shown below.

```
iWave-G23S> bdinfo
arch_number = 0xDEF5BCB5
boot_params = 0x40000100
DRAM bank   = 0x00000000
-> start    = 0x40000000
-> size     = 0x40000000
ethaddr     = 00:01:02:03:04:05
ip_addr     = <NULL>
baudrate    = 115200 bps
TLB addr    = 0x5FFF0000
relocaddr   = 0x5FF77000
reloc off   = 0x79C73000
irq_sp      = 0x5FE74F60
sp start    = 0x5FE74F50
FB base     = 0x00000000
```

- To list the available commands and descriptions in U-Boot level, execute the below command and the available commands will be displayed in command prompt as shown below.

```
iWave-G23S> help
bdinfo      - print Board Info structure
bootm       - boot application image from memory
fdt         - flattened device tree utility commands
go          - start application at address 'addr'
help        - print command description/usage
md          - memory display
mmc         - MMC sub system
mmcinfo     - display MMC info
mw          - memory write (fill)
printenv    - print environment variables
reset       - Perform RESET of the CPU
run         - run commands in an environment variable
saveenv     - save environment variables to persistent storage
setenv      - set environment variables
version     - print monitor, compiler and linker version
```

4.2 RAM Test

- In RZ/G1C PI SBC, the RAM physical address is from 0x40000000 to 0x5FFFFFFF.
- To write the data into RAM location, execute the below command.

```
iWave-G23S> mw <RAM_addr> <DATA> <No_of_location_to_be_write>
```

- To display the data in the RAM location, execute the below command.

```
iWave-G23S> md <RAM_addr>  <No_of_location_to_be_display>
```

- To test the RAM read/write, execute the below command.

```
iWave-G23S> mtest <RAM_addr_start>  <RAM_addr_end> <DATA> <No_of_times>
```

Example:

```
iWave-G23S> mtest 0xb0000000 0xb0080000 0xAABBCCDD 0x1
Pattern AABBCCDD  Writing...  Reading...Tested 1 iteration(s) without errors.
```

Note: Accessing the restricted RAM area or other physical address may cause unpredictable behaviour. Make sure, you are not entering the restricted area RAM address. 0x5F800000-0x5FFFFFFF is the u-boot RAM location and this RAM area should not be accessed.

4.3 SD/MMC Test

- To initialize the particular SD/MMC device, execute the below command.

  iWave-G23S> mmc dev <SD slot No>

- The SD/MMC static slot numbers are below.

  Micro SD (SDHI2) - 0

  eMMC (MMC) -1

- To display the SD/eMMC device information, execute the below command.

  iWave-G23S> mmcinfo

  Device: sh-sdhi

  Manufacturer ID: 3

  OEM: 5344

  Name: SU04G

  Tran Speed: 25000000

  Rd Block Len: 512

  SD version 3.0

  Clock: 50000000

  High Capacity: Yes

  Capacity: 3965190144 Bytes

  Bus Width: 4-bit

4.4 SPI NOR Flash Test

Caution: Since SPI Flash is the default boot device, accessing the SPI Flash will corrupt the boot code.

- To enable the SPI flash, execute the below command.

  iWave-G23S> sf probe

  SF: Detected SST25VF016B with page size 256 Bytes, erase size 4 KiB, total 2 MiB

- To erase the contents in SPI flash, execute the below command.

  iWave-G23S> sf erase <offset_address> <size>

  Example:

  iWave-G23S> sf erase 0x000000 0x10000

  Erasing SPI NOR flash 0x0 [0x10000 bytes]

  .....SUCCESS

- To write any data to the SPI flash, first need to write that data into the RAM location then can be copied to SPI flash.
- To write the data into RAM, refer the RAM Test section
- To write the data from RAM into SPI flash, execute the below command.

  iWave-G23S> sf write <RAM_addr> <flash_offset><size>

  Example:

  iWave-G23S> sf write 0x40800000 0x000000 0x100

  Writing SPI NOR flash 0x0 [0x100 bytes] <- ram 0x40800000

  SUCCESS

4.5 USB Test

- To scan the USB controller, execute the below command.

  iWave-G23S> usb start

- To show the available USB devices, execute the below command.

  iWave-G23S> usb info

- To show the USB storage devices details, execute the below command.

  iWave-G23S> usb storage

- To set/ show the current USB storage device, execute the below command.

  iWave-G23S> usb dev <device id>

  Example:

  iWave-G23S> usb dev 0

4.6 Environment variables settings

By default the environment variables will be saved in SPI Flash device. In the default environment setting, Linux booting from eMMC, UART1 as debug port with 115200bps 8n1 baudrate and mac address 00:01:02:03:04:05 are supported.

4.6.1 eMMC boot

- To load the kernel and file systems from the eMMC, set the environment variables as shown below.

  iWave-G23S> *setenv bootcmd_mmc 'run bootargs_mmc;run fdt_check;mmc dev 1;fatload mmc 1 ${loadaddr} ${kernel};fatload mmc 1 ${fdt_addr} ${fdt_file};bootm ${loadaddr} - ${fdt_addr}'*

  iWave-G23S> *setenv bootargs_mmc 'setenv bootargs ${bootargs_base} root=/dev/mmcblk1p2 rootwait rootfstype=ext4 rw'*

  iWave-G23S> *setenv bootcmd 'run bootcmd_mmc'*

  iWave-G23S> *saveenv*

- To boot the platform, execute the below command.

  iWave-G23S> *boot*

4.6.2 Micro SD boot

- To load the kernel and file systems from the Micro SD, set the environment variables as shown below.

  iWave-G23S> *setenv bootcmd_msd 'run bootargs_msd;mmc dev 0;fatload mmc 0 ${loadaddr} ${kernel};fatload mmc 0 ${fdt_addr} ${fdt_file};bootm ${loadaddr} - ${fdt_addr}'*

  iWave-G23S> *setenv bootargs_msd 'setenv bootargs ${bootargs_base} root=/dev/mmcblk0p2 rootwait rootfstype=ext4 rw'*

  iWave-G23S> *setenv bootcmd 'run bootcmd_msd'*

  iWave-G23S> *saveenv*

- To boot the platform, execute the below command.

  iWave-G23S> *boot*

4.6.3 HDMI settings

- By default, the HDMI is supported with 1080i resolution.
- To change the HDMI resolution, add the environment variables as shown below to bootargs.

  video=[name of connector]:[resolution][-<bpp>][@<refresh rate>][i]

  Example:

  iWave-G23S> setenv bootargs_mmc 'setenv bootargs ${bootargs_base} root=/dev/mmcblk1p2 rootwait rootfstype=ext4 rw video=HDMI-A-1:1280x720-32@60'
- To save the environment variables, execute the below command.

  iWave-G23S> saveenv
- To boot the platform, execute the below command.

  iWave-G23S> boot
- The HDMI supported resolutions are,

  1920x1080i, 1280x720p, 1024x768p, 720x480p, 640x480p

4.6.4 Optional features setting

- To set the MAC address for the platform and to save, execute the below commands.

  iWave-G23S> setenv ethaddr '<MAC addr>'

  iWave-G23S> saveenv
- By default dual core will be in active state. To limit the no of processors add the below string in the boot arguments.

  maxcpus=<maximum no of cpu to be active>

  Example

  iWave-G23S> setenv bootargs_mmc 'setenv bootargs ${bootargs_base} root=/dev/mmcblk1p2 rootwait rootfstype=ext4 rw maxcpus=1'

4.6.5 Default Environment Variable

- To restore the default environment variables, execute the below commands

  iWave-G23S> env default -f -a

  iWave-G23S> saveenv

  iWave-G23S> reset

Caution: Since MAC address will be stored in the SPI Flash by default, restoring the default environment variable will restore the default MAC address.

5. LINUX PERIPHERAL TESTING

This section explains about testing the peripherals in Linux OS level for RZ/G1C SBC. Connect debug UART with host PC and Power ON the RZ/G1C SBC and enter into the kernel console to test peripherals in Linux.

RZ/G1C SBC can boot Linux OS image from the following boot media devices.

- eMMC (default)
- Micro SD
- TFTP and NFS

5.1 Block devices Test

The RZ/G1C SBC will support the below block devices.

- eMMC, Micro SD
- USB host
- USB OTG (device)
- SPI NOR Flash

5.1.1 Testing device Requirements

To test the block devices supported by RZ/G1C SBC, following Items are required.

- USB memory stick
- Micro SD
- USB Type A to micro B cable.

5.1.2 Block Device Test

- The Micro SD / eMMC / USB (Host) / USB OTG (as Host) will mount in below mentioned directories.

  Micro SD - */media/mmcblk0p1, /media/mmcblk0p2 …etc*

  eMMC - */media/mmcblk1p1, /media/mmcblk1p2 …etc*

  USB - */media/sda1, /media/sdb1 …etc*

- To mount the device manually, if not mounted automatically, execute the below command

  root@iWave-G23S~$ mount -t <type> <device node> <Directory>

  Example:

  root@iWave-G23S~$ mount -t vfat /dev/mmcblk1p1 /mnt/mmcblk1p1

- To view the contents, execute the below command.

  root@iWave-G23S~$ cd /<mount_directory>

  root@iWave-G23S/<mount_directory>~$ ls

- To create a directory and remove a directory from the mounted partition, execute the below commands.

  root@iWave-G23S/<mount_directory>$ mkdir <directory_name>

  root@iWave-G23S/<mount_directory>$ rm –rf <target_directory>

- To copy a file to the mounted partition, execute the below command.

  root@iWave-G23S/<mount_directory>$ cp <source_file> <Destination>

- To exit from the mount folder, execute the below command.

  root@iWave-G23S/<mount_directory>$ cd /root

- To unmount the device, execute the below command.

  root@iWave-G23S~$ umount <Mount Directory>

5.1.3 USB OTG as device

- Connect OTG Cable to OTG Port
- To insert the file storage module.

  root@iWave-G23S~$ insmod /lib/modules/3.10.31-ltsi/kernel/fs/configfs/configfs.ko

  root@iWave-G23S~$ insmod /lib/modules/3.10.31-ltsi/kernel/drivers/usb/gadget/libcomposite.ko

  root@iWave-G23S~$ insmod /lib/modules/3.10.31-ltsi/kernel/drivers/usb/gadget/g_mass_storage.ko file=/dev/mmcblk1p1 removable=1
- After successful module registration, it shows the below debug message.

  g_mass_storage gadget: Mass Storage Function, version: 2009/09/11

  g_mass_storage gadget: Number of LUNs=1

  lun0: LUN: removable file: /dev/mmcblk1p1

  g_mass_storage gadget: Mass Storage Gadget, version: 2009/09/11

  g_mass_storage gadget: userspace failed to provide iSerialNumber

  g_mass_storage gadget: g_mass_storage ready

  g_mass_storage gadget: high-speed config #1: Linux File-Backed Storage
- Then the files and folders from RZ/G1C SBC's eMMC will be mounted in the Windows Host PC as removable disk.
- To unload the modules, execute the below command

  root@iWave-G23S~$ rmmod g_mass_storage.ko

  root@iWave-G23S~$ rmmod libcomposite.ko

  root@iWave-G23S~$ rmmod configfs.ko
- To change the directory to "root" folder, execute the below command

  root@iWave-G23S~$ cd /home/root

5.1.4 SPI NOR Flash Test

Caution: Since SPI flash is used as boot device, accessing the SPI Flash will corrupt the boot code.

- To display the SPI NOR flash information, execute the below command.

  root@iWave-G23S ~$ cat /proc/mtd

- To mount the SPI NOR flash partitions, execute the below command.

  root@iWave-G23S~$ mount –t jffs2 /dev/mtdblock0 /<mount_directory>

- To view the files and folders in mounted partitions, execute the below command.

  root@iWave-G23S~$ cd /<mount_directory>

  root@iWave-G23S/<mount_directory>$ ls

- To create a directory and remove a directory in mounted partition, execute the below commands respectively.

  root@iWave-G23S/<mount_directory>$ mkdir <directory_name>

  root@iWave-G23S/<mount_directory>$ rm –rf <target_directory>

- To copy a file to the mounted partition, execute the below command.

  root@iWave-G23S/<mount_folder>$ cp <soruce_file> <Destination>

- To exit from the mount partitions and to unmount, execute the below command.

  root@iWave-G23S/<mount_directory>$ cd /root

- To unmount the device, execute the below command.

  root@iWave-G23S~$ umount <Mount Directory>

5.2 Network devices Test

The RZ/G1C SBC will support the below network devices.

- Ethernet

5.2.1 Testing device Requirements

To test the Network devices supported by RZ/G1C SBC, following Items are required.

- Ethernet connection
- Ethernet cable

5.2.2 Ethernet Test

This section explains, how to test the Ethernet with different speed (EtherAVB 100/1000Mbps) and duplex (half/full) in the RZ/G1C SBC.

- Connect the Ethernet cable and to enable the Ethernet device, execute the below command.

  root@iWave-G23S ~$ ifconfig eth0 up

- To set the IP address using DHCP, execute the below command.

  root@iWave-G23S ~$ udhcpc -i eth0

  udhcpc (v1.22.1) started

  Sending discover...

  *Sending select for *********...*

  *Lease of ********** obtained, lease time 43200*

  Deleting routers

  */etc/udhcpc.d/50default: Adding DNS *********

  */etc/udhcpc.d/50default: Adding DNS *********

  */etc/udhcpc.d/50default: Adding DNS *********

- To check the IP address set, execute the below command.

  root@iWave-G23S ~$ ifconfig

  eth0 Link encap:Ethernet HWaddr 00:18:8B:0B:BA:3C

  *inet addr:******** Bcast:0.0.0.0 Mask:255.255.254.0*

  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

  RX packets:14864 errors:17 dropped:2378 overruns:0 frame:17

  TX packets:89 errors:0 dropped:0 overruns:0 carrier:0

  collisions:0 txqueuelen:1000

  RX bytes:1361071 (1.2 MiB) TX bytes:15610 (15.2 KiB)

  Interrupt:195 DMA chan:ff

- To test Ethernet, execute the below command.

  root@iWave-G23S~$ ping <any_ip_addr>

  *PING ********** (**********) 56(84) bytes of data.*

  *64 bytes from **********: icmp_seq=1 ttl=64 time=0.206 ms*

  *64 bytes from **********: icmp_seq=2 ttl=64 time=0.160 ms*

  *64 bytes from **********: icmp_seq=6 ttl=64 time=0.161 ms*

5.2.3 Ethernet speed and duplex settings

This section explains how to set the Ethernet speed to 100/1000 Mbps or half/full duplex in the RZ/G1C SBC. The "ethtool" utility used to query and change settings such as speed, auto- negotiation and checksum offload on many network devices, especially Ethernet devices.

- To turn off Auto-Negotiate feature, execute the belwo command.

  root@iWave-G23S~$ ethtool -s eth0 autoneg off

- To set the desired speed/duplex in the Ethernet. execute the below command.

  root@iWave-G23S~$ ethtool -s ethX speed [SPEED] duplex [DUPLEX]

  Example:

  root@iWave-G23S~$ ethtool -s eth0 speed 100 duplex half

- To check the current Ethernet network speed and duplex settings, execute the below command.

  root@iWave-G23S~$ ethtool eth0

5.2.4 File transfer using TFTP server

- To receive any file from TFTP server to RZ/G1C SBC, execute the below command

  root@iWave-G23S~$ tftp -g <server_ip> -r <file_name>

- To transmit any file from RZ/G1C SBC to TFTP server (host PC), execute the below command

  root@iWave-G23S~$ tftp -p <server_ip> -l <file_name>

5.2.5 Folder mount from NFS

- To mount any folder from NFS server (Host PC) to RZ/G1C SBC, execute the below command

  root@iWave-G23S~$ mount -o nolock -t nfs <server_ip>://<filepath> <mount_directory>

- To view the NFS mounted files and folders, execute below command.

  root@iWave-G23S~$ ls /<mount_directory>/

Note: To configure the host PC (under Linux OS) for TFTP and NFS server refer the **TFTP & NFS Host PC setup** *section.*

5.2.6 TFTP & NFS Host PC setup

This section describes the steps to setup a TFTP server and NFS server in Ubuntu Linux distributions Host PC.

- The following host pc setup is required only once per host.
- Install the nfs-kernel-server, tftpd and xinetd packages required to setup NFS and TFTP server in the host PC.

  $sudo apt-get install nfs-kernel-server xinetd tftpd tftp –y

NFS Server

To set up the NFS server, first set up the configuration files for NFS, and then start the NFS services.

- Open file /etc/exports by below command

  $ sudo vim /etc/exports
- Insert the following line in /etc/exports file

  *<path to rootfs> *(rw,sync,no_root_squash)*

  Example:

  */home/iwave/test/rootfs *(rw,sync,no_root_squash)*
- Restart the NFS server.

  $ sudo /etc/init.d/nfs-kernel-server restart

TFTP server

- Create the tftp configuration file and insert the following content.

```
$sudo nano /etc/xinetd.d/tftp
service tftp
{
protocol = udp
prot = 69
socket_type = dgram
wait= yes
user = <user_name>
server = /usr/sbin/in.tftpd
server_args = /tftpboot -s
disable = no
}
```

Example:

```
service tftp
{
protocol = udp
prot = 69
socket_type = dgram
wait = yes
user = iwave
server = /usr/sbin/in.tftpd
server_args = /tftpboot -s
disable = no
}
```

- Change the ownership of the directory.

```
$ sudo mkdir /tftpboot
$ sudo chmod -R 777 /tftpboot
$ sudo chown -R <user_name>:<user_name> /tftpboot
```

- Restart the tftp services,

```
$ sudo service xinetd restart
```

- Verify the TFTP is running correctly or not

```
$netstat -na | grep LIST | grep 22
```

5.3 Display devices Test

The RZ/G1C SBC will support the below display devices.

- HDMI

5.3.1 Testing device Requirements

To test the display devices supported by RZ/G1C SBC, following Items are required.

- HDMI Monitor with cable.

5.3.2 HDMI Test

- While the HDMI is connected to RZ/G1C SBC, GUI will be displayed in HDMI as follows.

Figure 2: HDMI -Yocto GUI

5.4 HID devices Test

The RZ/G1C SBC will support the below Human Interface devices.

- USB HID devices – Mouse and Keyboard

5.4.1 Testing device Requirements

To test the Human Interface Devices supported by RZ/G1C SBC, following Items are required.

- USB mouse and keyboard.

5.4.2 Mouse Test

- Insert the USB Mouse in RZ/G1C development platform USB slot. The following message will be displayed in command prompt.

  usb 1-1.2: new low-speed USB device number 4 using xhci-hcd

  xhci-hcd ee000000.xhci: Can't reset device (slot ID 2) in default state

  xhci-hcd ee000000.xhci: Not freeing device rings.

  usb 1-1.2: ep 0x81 - rounding interval to 64 microframes, ep desc says 80 microframes

  input: PixArt USB Optical Mouse as /devices/ee000000.xhci/usb1/1-1/1-1.2/1-1.2:1.0/input/input2

  hid-generic 0003:0461:4E22.0002: input: USB HID v1.11 Mouse [PixArt USB Optical Mouse] on usb-ee000000.xhci-1.2/input0

5.4.3 Keyboard Test

- Insert the USB Keyboard in RZ/G1C development platform USB slot. The following message will be displayed in command prompt.

  usb 1-1.2: new low-speed USB device number 6 using xhci-hcd

  xhci-hcd ee000000.xhci: Can't reset device (slot ID 2) in default state

  xhci-hcd ee000000.xhci: Not freeing device rings.

  usb 1-1.2: ep 0x81 - rounding interval to 64 microframes, ep desc says 80 microframes

  input: DELL Dell USB Entry Keyboard as /devices/ee000000.xhci/usb1/1-1/1-1.2/1-1.2:1.0/input/input3

  hid-generic 0003:413C:2107.0003: input: USB HID v1.10 Keyboard [DELL Dell USB Entry Keyboard] on usb-ee000000.xhci-1.2/input0

5.5 UART Test

The given BSP supports the UART with different baud rates. This section explains how to test the UART in the RZ/G1C SBC.

- Connect the RZ/G1C SBC's data UART port with serial port of host PC using serial cable.
- The supported UART devices and its nodes are listed below,

  UART 1 - /dev/ ttySC1 (SCIF1 & Debug console)

  UART 2 - /dev/ ttySC2 (SCIF2)

  UART 3 - /dev/ ttySC3 (SCIF4)

  UART 4 - /dev/ ttySC4 (SCIF5)

  UART 5 - /dev/ ttySC5 (HSCIF1 with CTS, RTS)

  UART 6 - /dev/ ttySC6 (HSCIF2 with CTS, RTS)

- In case of data UART, Open another UART console in host PC and set the serial port settings as mentioned below.

  Bits per second: 9600 bps

  Data bits: 8

  Parity: none

  Stop bits: 1

  Flow control: none

- To transmit data through the UART, execute the below command.

  root@iWave-G23S/$ echo "uart_test_message" > /dev/<node>

  Example:

  root@iWave-G23S/$ echo iW-RainboW-G23S > /dev/ttySC3

- To receive the data by UART, execute the below command.

  root@iWave-G23S/$ cat /dev/<node>

  Example:

  root@iWave-G23S/$ cat /dev/ttySC3

- To set the Baud rate to a different value, execute the below command

  root@iWave-G23S/$ stty -F /dev/<node> <crtscts> <Baud rate>

  Example:

  Without CTS-RTS support:

  root@iWave-G23S/$ stty -F /dev/ttySC3 115200

  With CTS-RTS support:

  root@iWave-G23S/$ stty -F /dev/ttySC5 crtscts 115200

- The default UART baud rate is 9600bps and the tested Baud rates are given below:

  300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 & 230400

5.6 Multimedia test

The RZ/G1C SBC supports below video devices.

- GPU Test
- Gstreamer package to play video files

5.6.1 Testing device Requirements

To test Multimedia devices supported by RZ/G1C SBC, following Items are required:

- HDMI monitor with cable.
- Analog Video Input

5.6.2 Analog Video Input Test

The given BSP supports analog video input. This section explains how to test analog video input with wayland in the RZ/G1C SBC.

- The analog video decoder supports NTSC and PAL formats.
- The analog video decoder interface will be detected in below device node.

  VIN - /dev/video0
- To export the Runtime Directory, execute the below command.

  root@iWave-G23S~$ export XDG_RUNTIME_DIR=/run/user/root
- To set environment variable and exporting libraries required for Gstreamer, execute the below command

  root@iWave-G23S~$ export LD_LIBRARY_PATH="/lib:/usr/lib:/usr/local/lib:"
- To load memory needed for Gstreamer, execute the below command.

  root@iWave-G23S~$ modprobe -a mmngr mmngrbuf s3ctl uvcs_cmn vspm fdpm
- To overlay the analog video input, execute the below command.

  root@iWave-G23S~$ gst-launch-1.0 -e v4l2src device=<device node> io-mode=dmabuf ! video/x-raw,width=<input width>,height=<input height>,format=UYVY ! vspfilter ! video/x-raw, format=BGRA, width=1920, height=1080 ! waylandsink

  Example:

  For NTSC:

  root@iWave-G23S/$ gst-launch-1.0 -e v4l2src device=<device node> io-mode=dmabuf ! video/x-raw,width=720,height=480,format=UYVY ! vspfilter ! video/x-raw, format=BGRA, width=1920, height=1080 ! waylandsink

  For PAL:

  root@iWave-G23S/$ gst-launch-1.0 -e v4l2src device=<device node> io-mode=dmabuf ! video/x-raw,width=720,height=576,format=UYVY ! vspfilter ! video/x-raw, format=BGRA, width=1920, height=1080 ! waylandsink

5.6.3 GPU Test

The given BSP supports GPU libraries. The Processors supports OpenGL ES. This section explains how to test GPU in the RZ/G1C SBC.

- To test GPU information in Wayland, execute the below command

  root@iWave-G23S~$ export XDG_RUNTIME_DIR=/var/run/user/root

  root@iWave-G23S~$ weston-simple-egl

5.6.4 Gstreamer Test

This section explains how to play a multimedia file using Gstreamer framework in Wayland in the RZ/G1C SBC.

- To export the Runtime Directory, execute the below command.

  root@iWave-G23S~$ export XDG_RUNTIME_DIR=/run/user/root

- To load memory needed for Gstreamer, execute the below command

  root@iWave-G23S~$ modprobe -a mmngr mmngrbuf s3ctl uvcs_cmn vspm fdpm

- To play Video file, execute the following command

  root@iWave-G23S~$ gst-launch-1.0 filesrc location=<file name with path> ! qtdemux name=demux demux.audio_0 ! queue ! aacparse ! faad ! alsasink device=hw:0,0 demux.video_0 ! queue ! h264parse ! omxh264dec ! vspfilter ! video/x-raw, format=BGRA, width=<output width>, height=<output height> ! waylandsink

  Example:

  root@iWave-G23S~$ gst-launch-1.0 filesrc location=/iwtest/KungFuHustle.mov ! qtdemux name=demux demux.audio_0 ! queue ! aacparse ! faad ! alsasink device=hw:0,0 demux.video_0 ! queue ! h264parse ! omxh264dec ! vspfilter ! video/x-raw, format=BGRA, width=1920, height=1080 ! waylandsink

5.7 GPIO Test

The external interface of this module is based on Linux. The interface for operating GPIO pin from a user land is GPIO sysfs. Device node of this module is shown below.

Table 4: GPIO device node

GPIO Port	GPIO Pin	GPIO PIN Number
GPIO0	GP-0-0 : GP-0-31	/sys/class/gpio/gpio992 : /sys/class/gpio/gpio1023
GPIO1	GP-1-0 : GP-1-29	/sys/class/gpio/gpio960 : /sys/class/gpio/gpio989
GPIO2	GP-2-0 : GP-2-31	/sys/class/gpio/gpio928 : /sys/class/gpio/gpio959
GPIO3	GP-3-0 : GP-3-31	/sys/class/gpio/gpio896 : /sys/class/gpio/gpio927
GPIO4	GP-4-0 : GP-4-31	/sys/class/gpio/gpio864 : /sys/class/gpio/gpio895
GPIO5	GP-5-0 : GP-5-31	/sys/class/gpio/gpio832 : /sys/class/gpio/gpio863

- To configure the iomux functionality of the pin as GPIO, execute the below command

  root@iWave-G23S~$ echo XXX > /sys/class/gpio/export

  XXX – GPIO pin No

- To set the direction of the GPIO pin, execute the below command

  root@iWave-G23S~$ echo < in/out > /sys/class/gpio/gpioXXX/direction

- To read the GPIO value when set as input, execute the below command

  root@iWave-G23S~$ cat /sys/class/gpio/gpioXXX/value

- To set the GPIO value when set as output, execute the below command

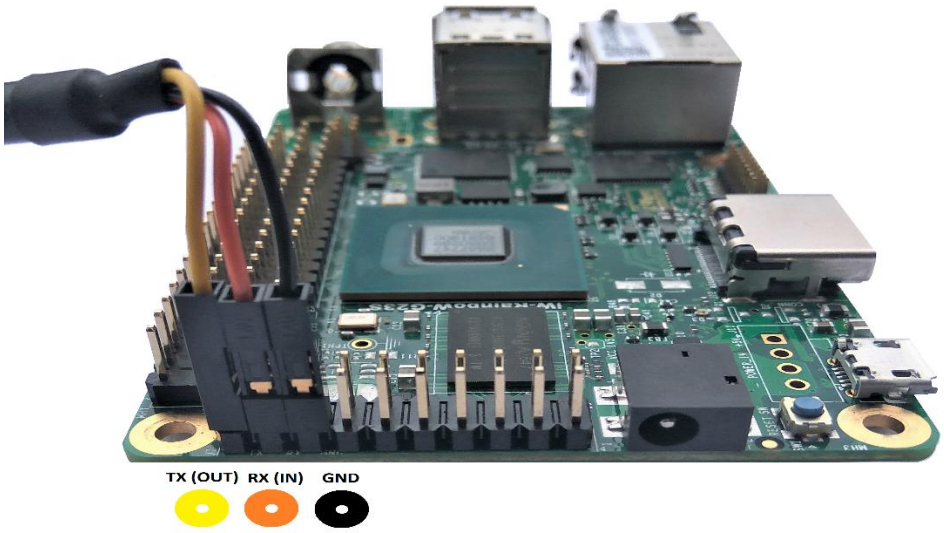
  root@iWave-G23S~$ echo <0/1> > /sys/class/gpio/gpioXXX/value

Note: Before changing the pin GPIO functionality and values, make sure that the pins are used as GPIO in the hardware.

6. APPENDIX – FREQUENTLY ASKED QUESTIONS

1. **How to connect the Debug port of RZ/G1C SBC with host PC / What are the settings to be done in host side to connect the debug port of RZ/G1C SBC?**

   Refer the below image to connect the Debug port.



2. **How to get the default environment variables in u-boot level after the environment variables modification/ Is it possible to erase the modified environment variables in u-boot level?**

   To set the default environment variables in u-boot, refer the Optional features setting

- To set the MAC address for the platform and to save, execute the below commands.
- iWave-G23S> setenv ethaddr '<MAC addr>'
- iWave-G23S> *saveenv*

By default dual core will be in active state. To limit the no of processors add the below string in the boot arguments.

maxcpus=<*maximum no of cpu to be active*>

Example

iWave-G23S> setenv bootargs_mmc 'setenv bootargs ${bootargs_base} root=/dev/mmcblk1p2 rootwait rootfstype=ext4 rw maxcpus=1'

Default Environment Variable section for RZ/G1C SBC.

3. **Why host PC thrown an error "uImage cannot build" while linux kernel compilation?**

   If the host thrown an error "uImage cannot build" while linux kernel standalone compilation, install the below package on host package.

   $sudo apt-get install uboot-mkimage

4. **How to make the cursor blink/unblink on the primary console display?**

   To disable the cursor blink on the primary console display, execute the below command.

   $echo 0 > /sys/class/graphics/fbcon/cursor_blink

   To enable the cursor blink on the primary console display, execute the below command.

   $echo 1 > /sys/class/graphics/fbcon/cursor_blink

5. **How to make display to wake up when screen goes off?**

   To make screen on, execute the below command

   DISPLAY=:0 xset -dpms s off

6. **What are the contents of jtag programming file?**

   The JTAG configuration file for the PEEDI JTAG debugger contains the following information.

```
[LICENSE]
KEY=UPDATE_15NOV2016, 1HNU-IHH5-EAB9-O
KEY=ARM7_ARM9, TKAV-OSOJ-2TSO-P
KEY=GDB_REMOTE, RH03-0Q8R-G57R-6
KEY=CORTEX_DEMO, BG2Q-JCLH-2K1I-V
[DEBUGGER]
PROTOCOL                    = gdb_remote    ; gdb remote
REMOTE_PORT                 = 2000          ; TCP/IP port
GDB_READ_INGNORE_TIME       = 3000
[TARGET]
PLATFORM                    = Cortex-A      ; platform is CortexA8
[PLATFORM_Cortex-A]
JTAG_CHAIN                  = 4         ; list of TAP controllers in the JTAG chain
JTAG_CLOCK                  = 5000  ; JTAG Clock in [kHz]
TRST_TYPE                   = PUSHPULL      ; type of TRST output: OPENDRAIN or PUSHPULL
RESET_TIME                  = 100           ; length of RESET pulse in ms; 0 means no RESET
CORE0                       = Cortex-A, 0, 0x4BA00477
CORE0_STARTUP_MODE          = RESET         ; stop the core immediately after reset
CORE0_ENDIAN                = LITTLE        ; core is little endian
CORE0_BREAKMODE             = SOFT          ; breakpoint mode
CORE0_WORKSPACE             = 0x40300000, 0xE000   ; address, length in bytes
CORE0_FLASH0                = SPI_FLASH
CORE0_PATH                  = "tftp://192.168.1.1"
CORE0_FILE                  = "test.bin", BIN, 0x20000000
[OS_ARM_LINUX_v26]
BASE                        = 4, 0xC02FF068
NEXT                        = 4, -0xB4
PID                         = 4, 0xE0
NAME                        = 16, 0x1FC
REGISTER                    = r4, 0x1C, 0x00
REGISTER                    = r5, 0x1C, 0x04
REGISTER                    = r6, 0x1C, 0x08
REGISTER                    = r7, 0x1C, 0x0C
REGISTER                    = r8, 0x1C, 0x10
```

```
REGISTER                    = r9, 0x1C, 0x14
REGISTER                    = r10,0x1C, 0x18
REGISTER                   = r11,0x1C, 0x1C
REGISTER                    = sp, 0x1C, 0x20
REGISTER                   = pc, 0x1C, 0x24
[SPI_FLASH]
CHIP                        = SPI25_FLASH
CPU                         = Cortex-A8
SPI_DIV                     = 2
FILE                        = "u-boot.bin", bin, 0x000
[SERIAL]                    ; serial port configuration
BAUD                       = 115200
STOP_BITS                   = 1
PARITY                     = NONE
TCP_PORT                    = 0
[TELNET]
PROMPT                    = "RZ/G1x> "       ; telnet prompt
;BACKSPACE                 = 127            ; comment out for autodetect
[DISPLAY]
BRIGHTNESS                  = 20             ; LED indicator brightness
VOLUME                      = 25             ; beeper volume
[ACTIONS]                   ; user defined scripts
1 = prog
[prog]                  ; program flash
flash set 0
flash erase
flash prog
```


iWave
Embedding Intelligence