WaRP7

The IoT and Wearable Development Platform

v1.1, Sep 16th, 2016: User Guide Manual

Table of Contents

1. Getting started with WaRP7 1
  1.1. Introduction 1
  1.2. License 1
  1.3. Platform Purpose 1
  1.4. Kit Contents 1
  1.5. Getting Started with Hardware 2
2. Hardware Overview 6
  2.1. Features/Specifications 6
  2.2. CPU Board 8
    2.2.1. Board Dimensions 8
    2.2.2. CPU 9
    2.2.3. Memory eMCP – LPDDR3 and eMMC 10
    2.2.4. Video and Display 10
    2.2.5. Connectivity 12
    2.2.6. Power Management 14
  2.3. IO Board 16
    2.3.1. Board Dimensions 16
    2.3.2. Audio 17
    2.3.3. Sensors 18
    2.3.4. Peripheral Expansion Port 20
3. Software Overview 21
4. U-Boot 21
  4.1. Obtaining an U-Boot image 22
  4.2. Updating an U-Boot image 22
    Setup WaRP7 for loading U-Boot 22
    Load U-Boot via USB 22
    Boot partition 23
5. Linux Kernel 24
  5.1. Obtaining a Linux Kernel image 24
  5.2. Updating a Linux Kernel image 24
6. The Yocto Project 25
  6.1. Obtaining an image generated by Yocto Project 25
    6.1.1. Update the host needed packages 25
    6.1.2. Dowload the metadata 25
    6.1.3. Building 25
  6.2. Steps to update the image 29
7. FreeRTOS 29
  7.1. Host machine setup 29

7.1.1. Setup Environment 29
7.1.2. Toolchain and source code 29
7.2. Running the "hello_world" demo 30
7.3. Other demos 31
7.3.1. Running "hello_world_ddr" demo 31
7.3.2. Running "hello_world_ocram" demo 31
8. Use Cases 31
8.1. Wi-Fi 31
8.1.1. Murata’s firmware enablement 31

1. Getting started with WaRP7

1.1. Introduction

This User Guide Manual is a moving document hosted at https://github.com/WaRP7/WaRP7-User-Guide.

If you find any mistake, error or typo, please create an issue at https://github.com/WaRP7/WaRP7-User-Guide/issues. Pull requests are also very welcome. You can help us to develop this document sending pull requests for fixes or adding new sections.

You can find help on how to create a pull request use GitHub Help

In case of doubt about the content of this document, please also create an issue at https://github.com/WaRP7/WaRP7-User-Guide/issues.

1.2. License

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.

To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/3.0/ or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

1.3. Platform Purpose

The WaRP7 is the next generation Internet of Things (IoT) and Wearable's Reference Platform. WaRP7 is a powerful, low-cost platform designed for a fast prototyping and reduces time to market. WaRP7 is optimized, comes in a tiny form factor and yet flexible enough to offer all the advantages of traditional development tools. It has been architected and designed from the ground up to address key challenges in the IoT and wearables markets, such as battery life, connectivity, user experience and miniaturization. WaRP7 is based on the NXP i.MX 7Solo applications processor that features an advanced implementation of the ARM® Cortex®-A7 core, as well as the ARM® Cortex®-M4 core. It comes with features such as on-board sensors, connectivity including NFC, Bluetooth®, Bluetooth Smart and Wi-Fi® and on-board external LPDDR3 memory.

This document is intended as an introduction to the WaRP7 CPU Board and IO Board hardware and focuses primarily on its initial setup and basic usage.

1.4. Kit Contents

In the box you will find the following items as shown in **Figure 1**.

- WaRP7 CPU Board
- WaRP7 IO Board
- Lithium Polymer Battery

- Quick Start Guide

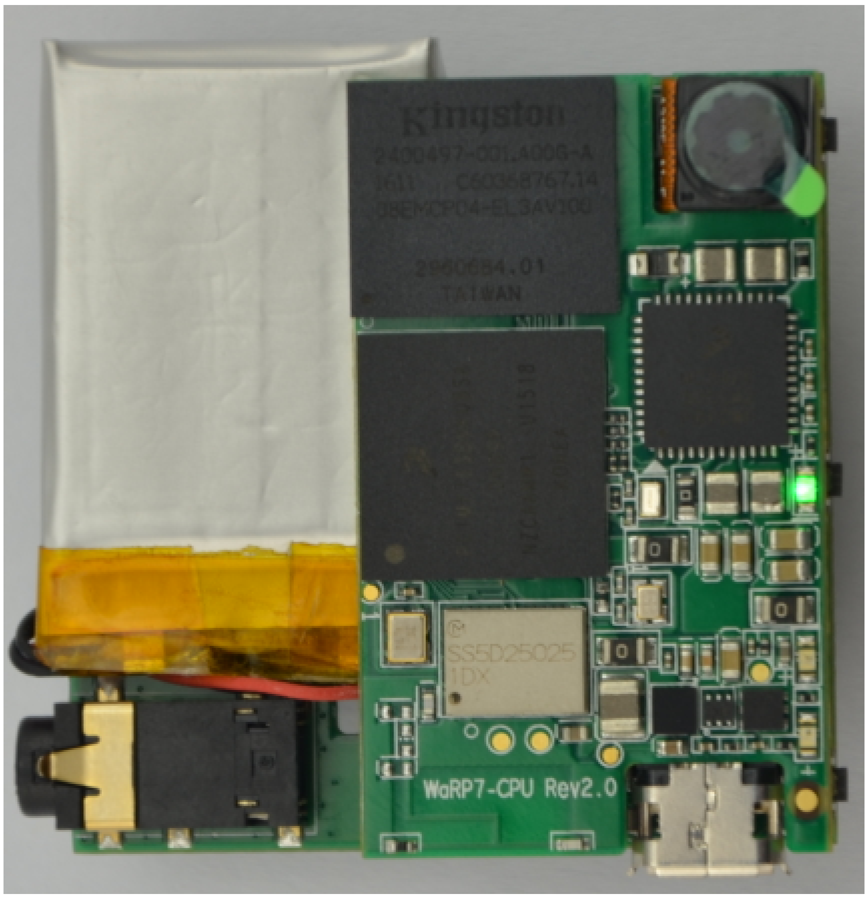


Figure 1. The display for the kit can be purchased separately (details coming soon). Check for availability at www.element14.com/warp7

1.5. Getting Started with Hardware

1. Connect NFC antenna to IO board

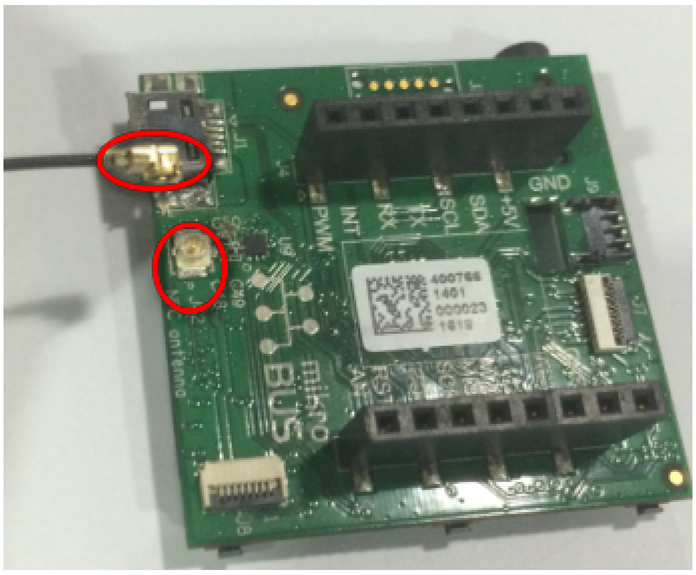


Figure 2. Where to connect the NFC antenna

2. Connect CPU board

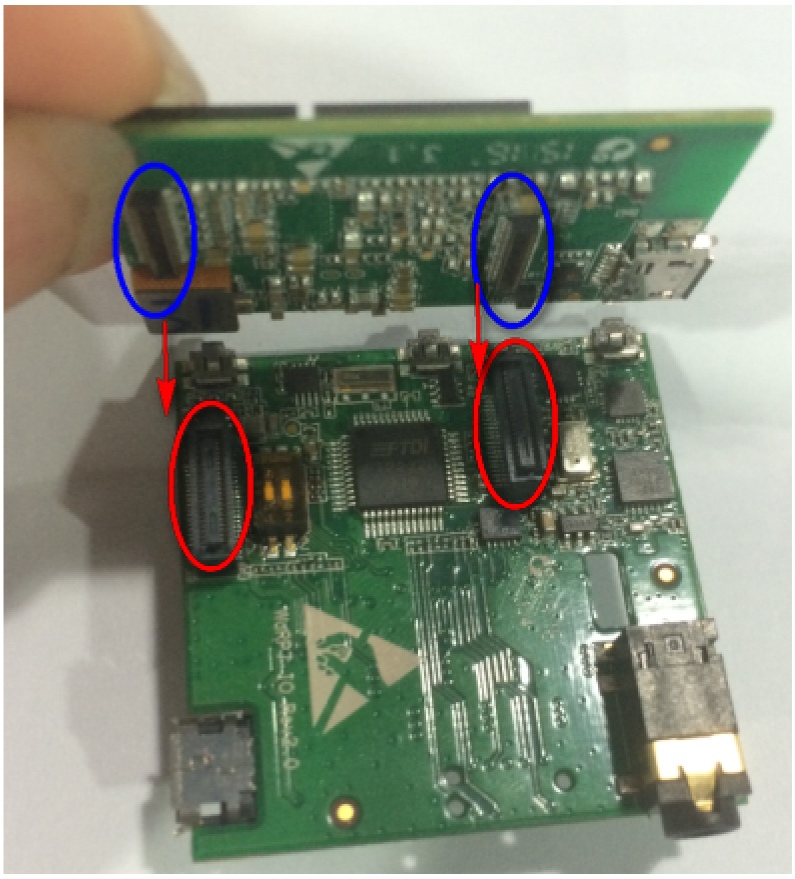


Figure 3. The connectors for the CPU board

3. Connect battery

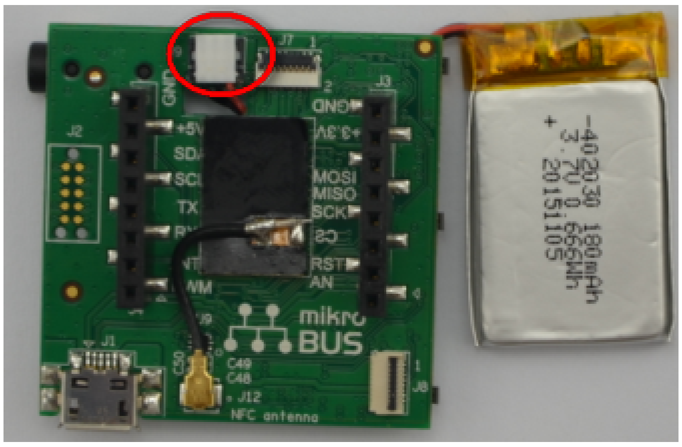


Figure 4. The connector for the battery

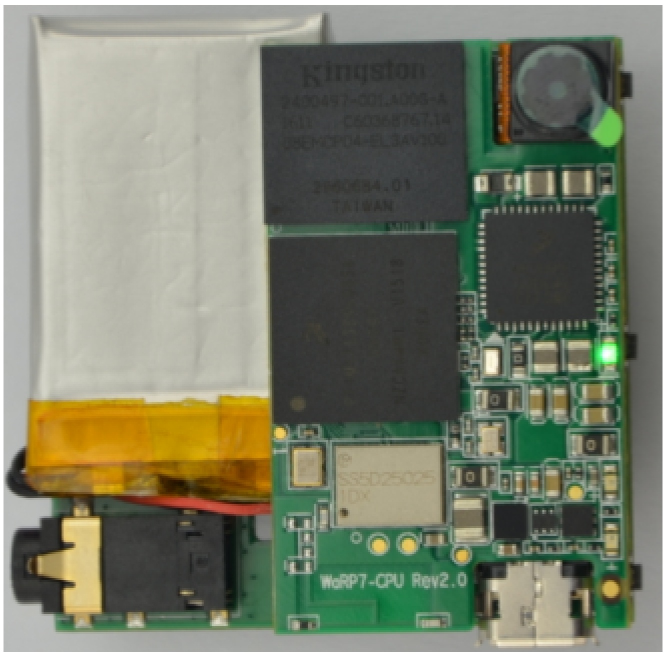


Figure 5. How the battery should be placed after attached

4. Connect LCD

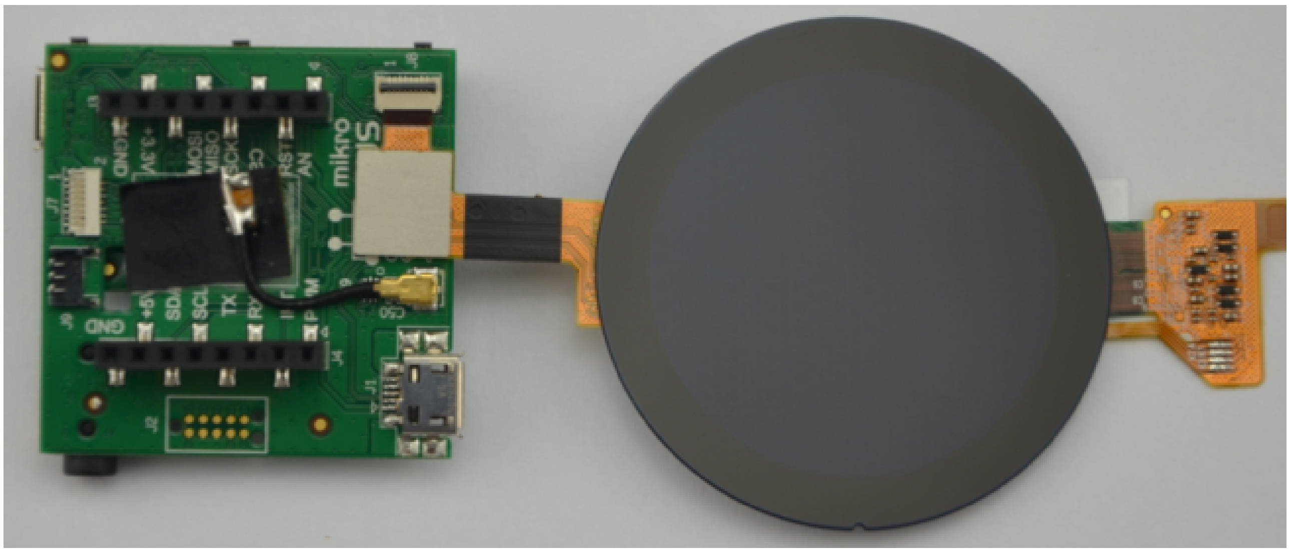


Figure 6. The first flat cable connected

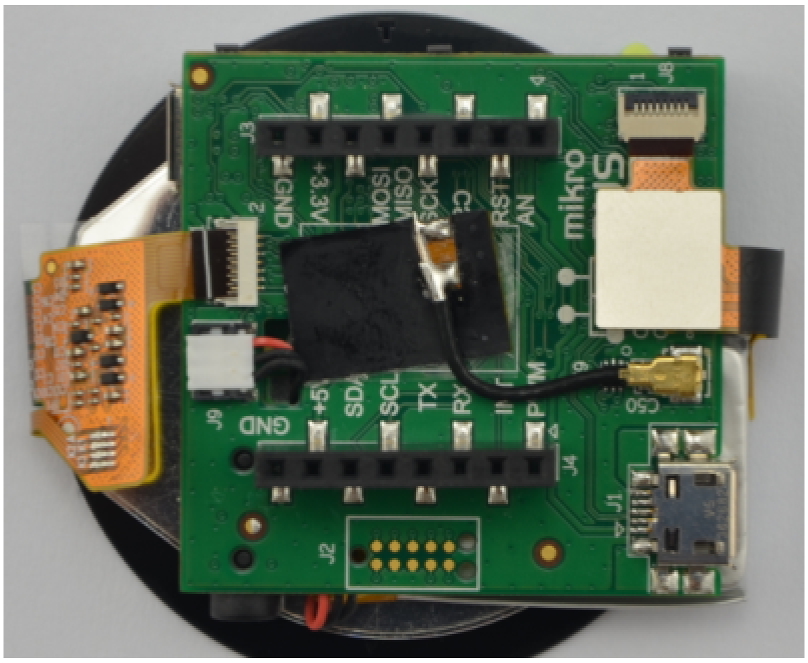


Figure 7. How to fold the flat cables and how to connect the second flat

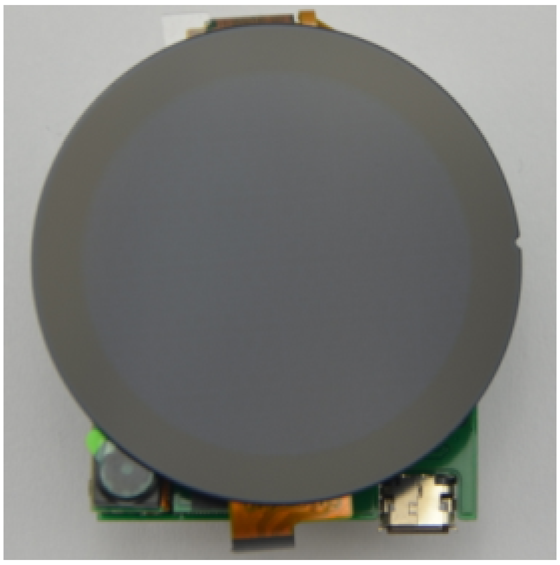


Figure 8. The WaRP7 board with the display

5. Connect USB

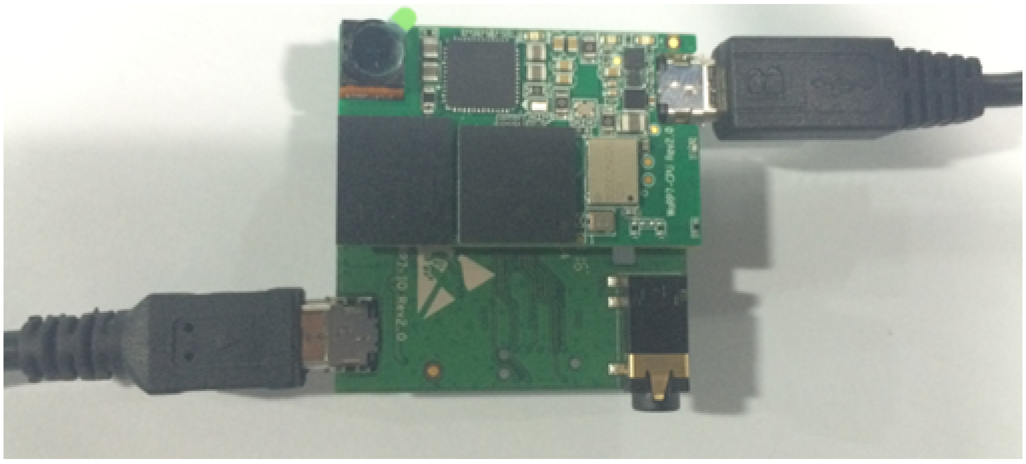


Figure 9. Both USB cable connected

2. Hardware Overview

The WaRP7 is the next generation Wearable's Reference Platform based on the NXP iMX7 Solo applications processor. The kit consists of 2 boards:

- A **CPU board** featuring the ARM Cortex-A7 based iMX7 processor, memory, power management and a combo WiFi/BLE module.
- An **IO board** which provides interface to variety of sensors, expansion and debugging capabilities.

2.1. Features/Specifications

Temperature Range

- The CPU & IO boards operate in the commercial temperature range, 0°C ~ 85°C
- The LCD operates temperature range : 0°C ~ 70°C.
- The battery operates temperature range : 0°C ~ 45°C.

The features present on CPU Board are listed on CPU Board.

Table 1. CPU Board

Features	CPU Board
Processor	NXP iMX7S ARM Cortex-A7/Cortex-M4
Memory	8GB, 8bit Embedded MMC/ 512MB LPDDR3
PMIC	PF3000 PMIC and BC3770 battery charger.
Wireless	Combo WiFI/BLE
Display/Camera interface	MIPI-DSI connector / MIPI - CSI connector
Power Source	USB/Battery
PCB	38mm x 23mm / 10 layers
Indicators	1 - Power, 1 - User

The features present on IO Board are listed on IO Board.

Table 2. IO Board

Features	IO Board
Debug Support	JTAG, Serial Header
Sensors	Accelerometer, Magnetometer Pressure Sensor Gyroscope
Audio	Stereo codec - Mic In / Line Out
Expansion	MikroE Click header

The following block diagrams show the connection present on each board and how the devices are connected.

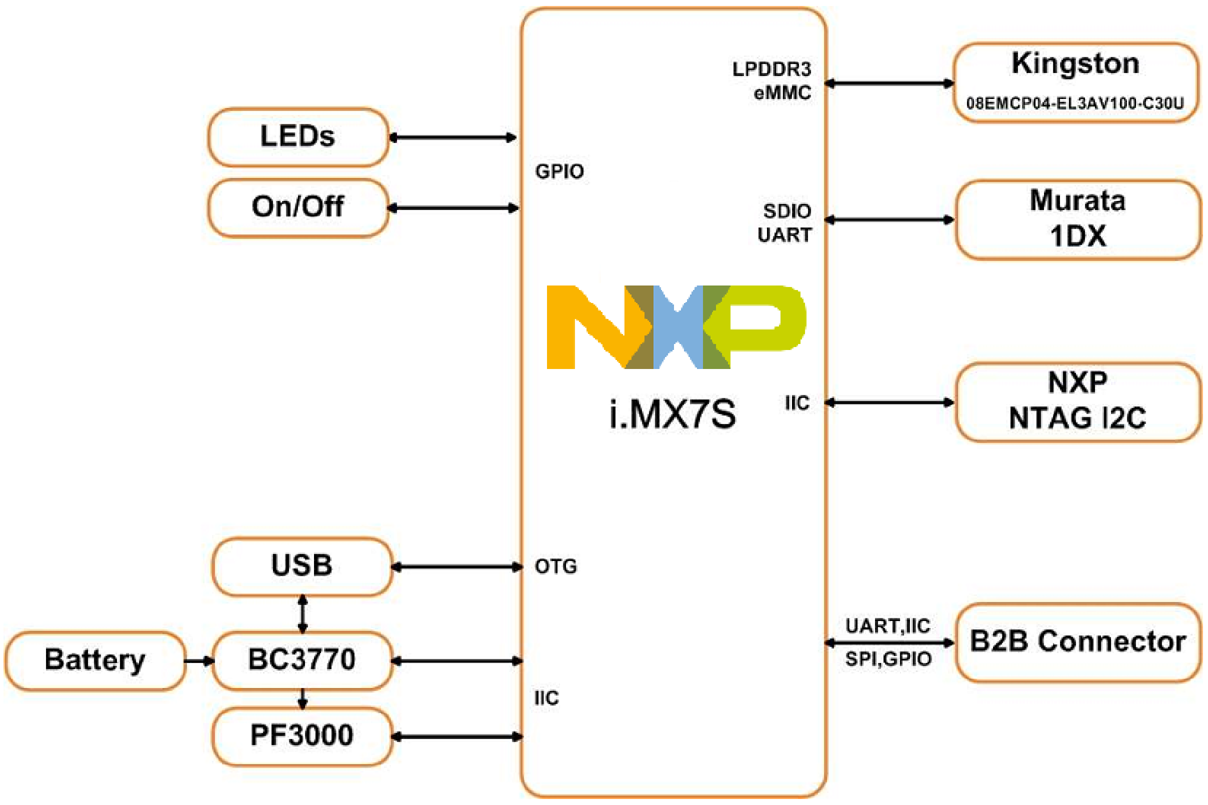


Figure 10. CPU Board

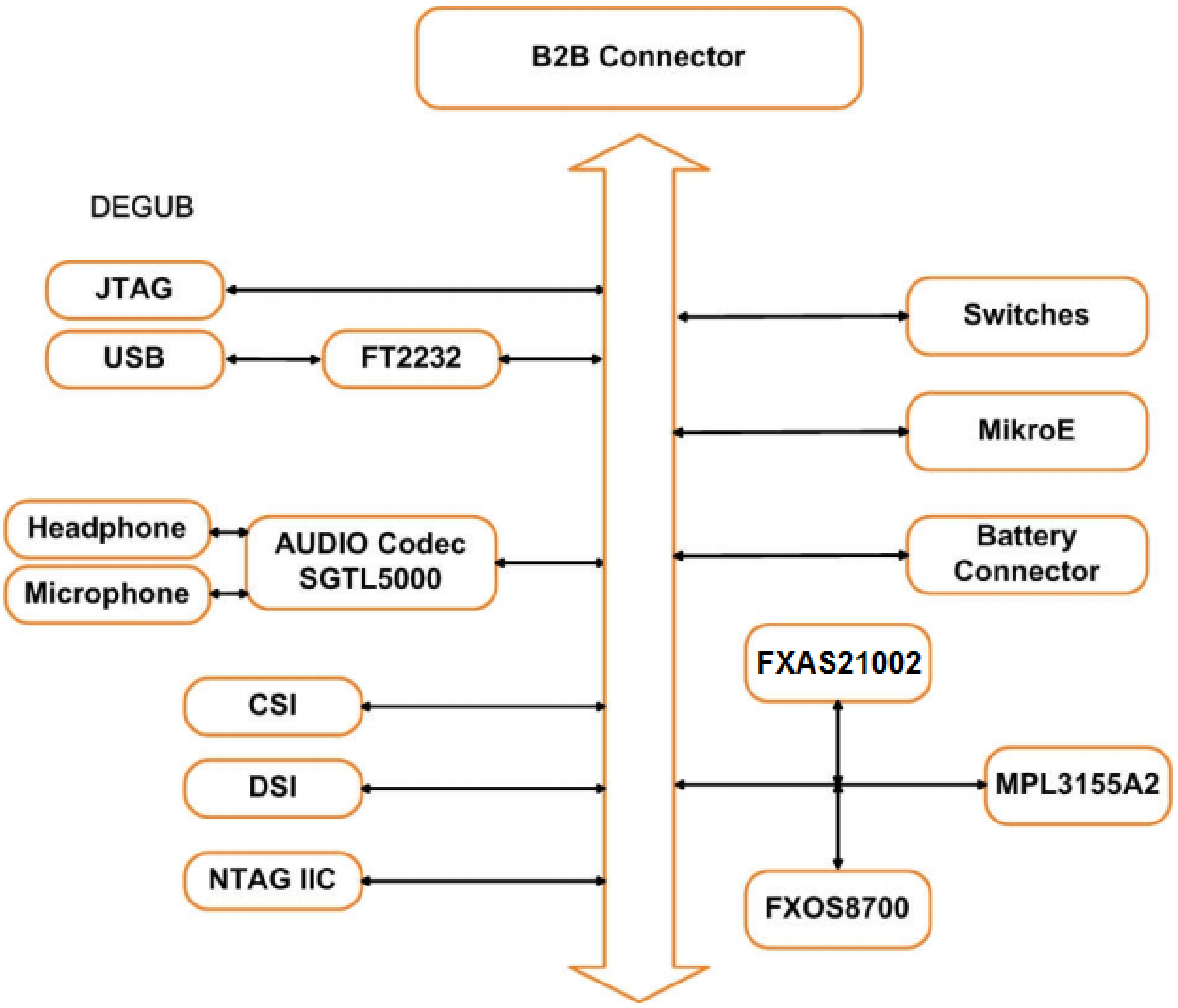


Figure 11. IO Board

2.2. CPU Board

2.2.1. Board Dimensions

Table 3. WaRP7 CPU Board Dimensions

Length	38 mm
Width	23 mm
PCB Thickness	1.0 mm

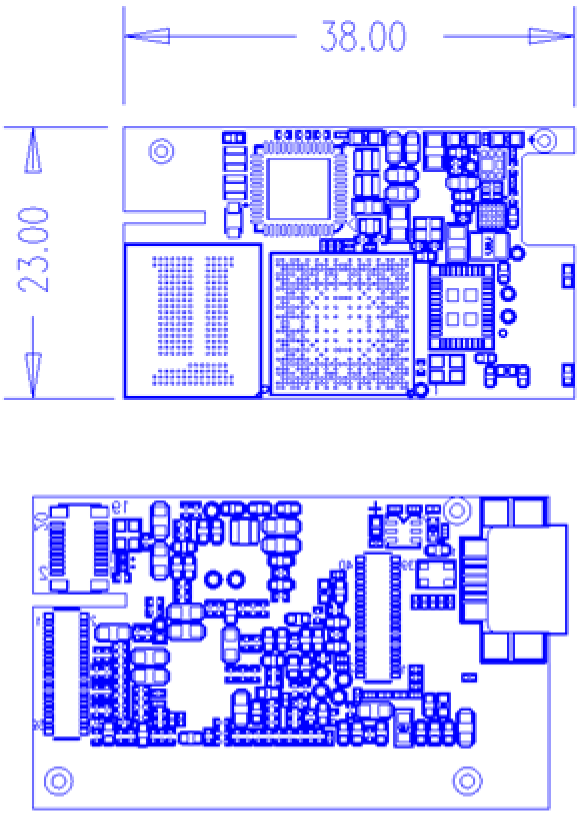


Figure 12. Board Dimensions

2.2.2. CPU

Main Processor: NXP MCIMX7S3DVK08SA - i.MX7S with 12mm x 12mm MAPBGA, 0.4mm.

The i.MX7S applications processor has an ARM Cortex-A7 core and an ARM Cortex-M4. The device is targeted for IoT, Wearable and general embedded markets.

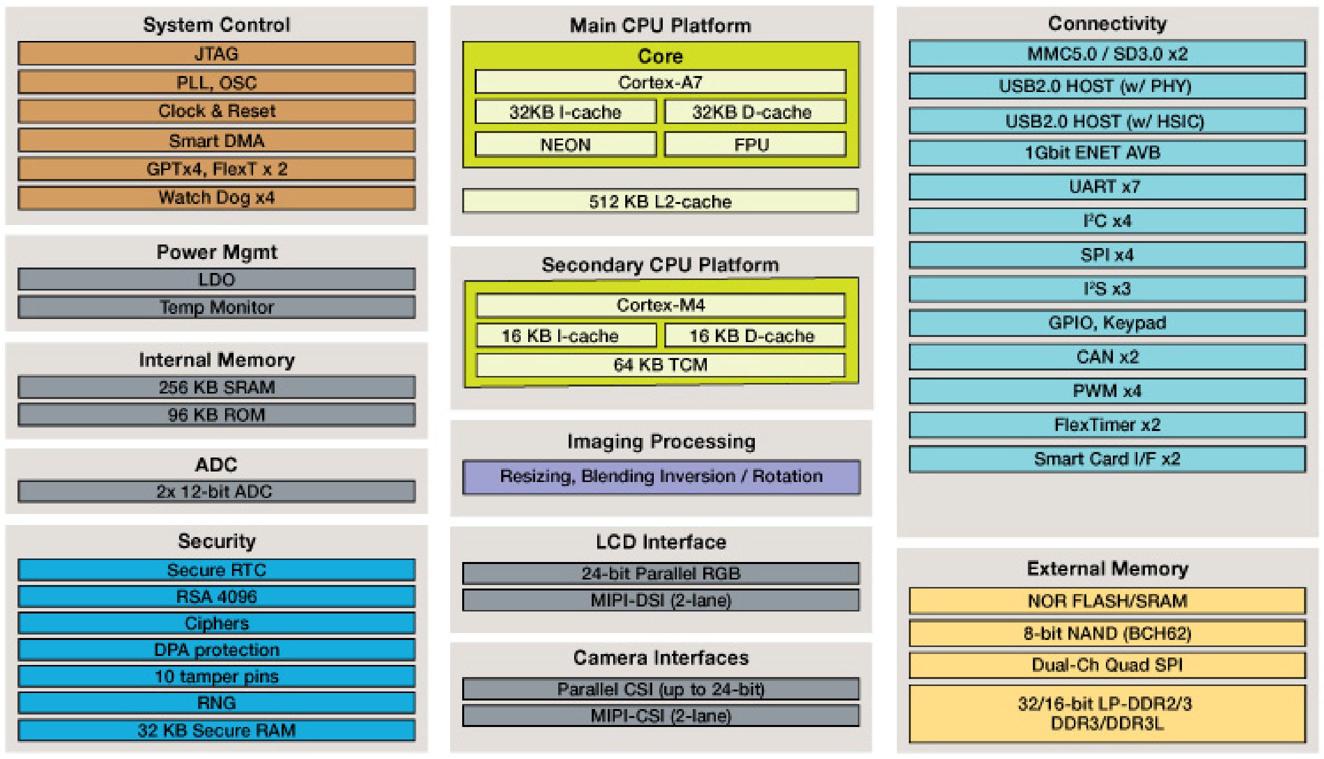


Figure 13. iMX7S SoC Diagram

2.2.3. Memory eMCP – LPDDR3 and eMMC

The CPU Board features a Multi-Chip Package Memory 08EMCP04-EL3AV100-C30U from Kingston which combines 8GB eMMC and 512MB Low Power DDR3 synchronous dynamic RAM. This comes in 221-ball FBGA package.

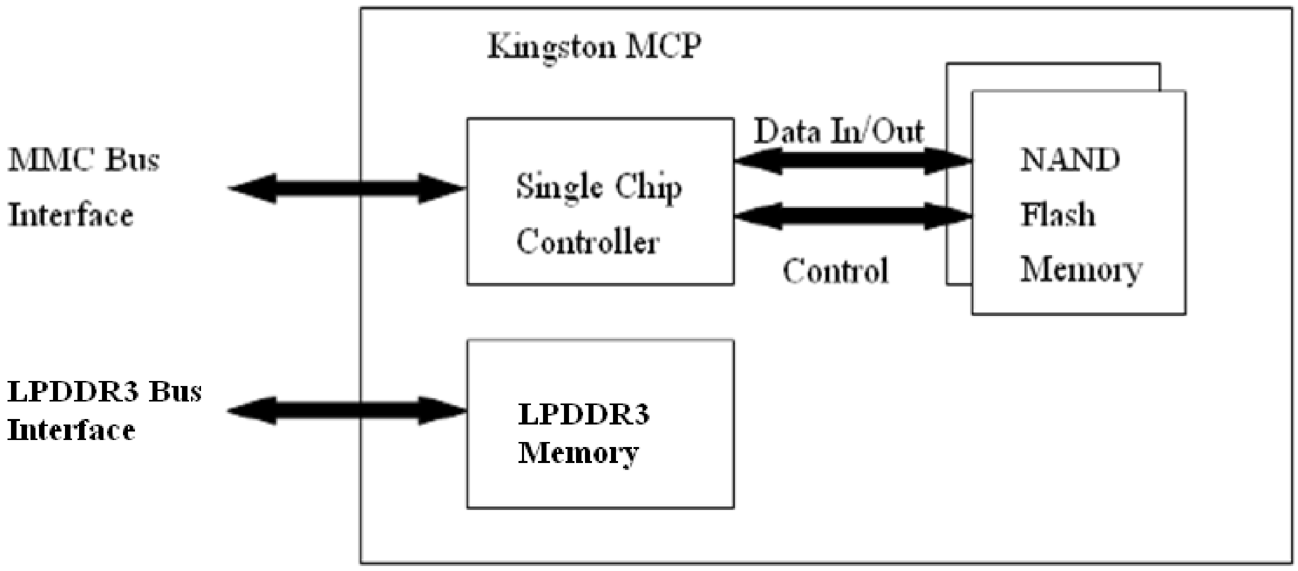


Figure 14. Kingston eMCP Block Diagram

2.2.4. Video and Display

The WaRP7 CPU board provides output video from MIPI-DSI and accepts input through MIPI-CSI.

MIPI-DSI

The CPU board includes a MIPI-DSI connector for outputting the video from the i.MX7S MIPI-DSI PHY via the MIPI-DSI interface.

DSI

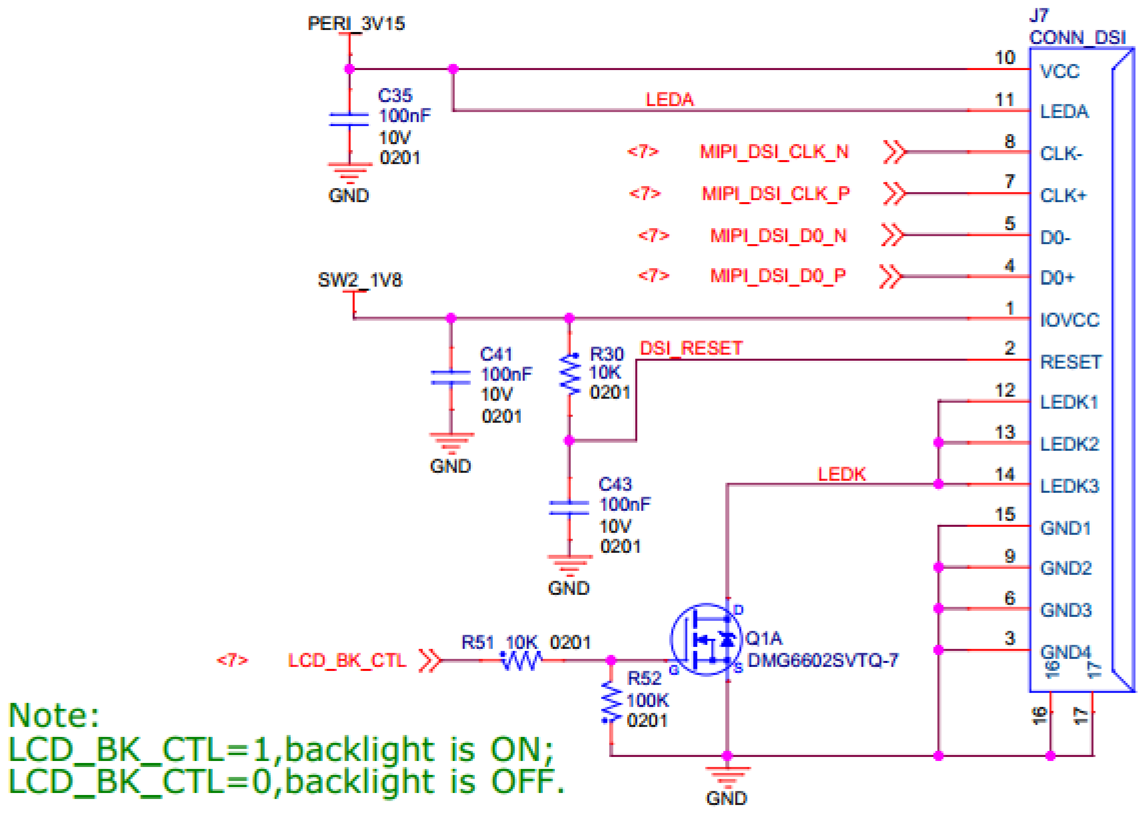


Figure 15. MIPI-DSI connector

Capacitive Touch Screen

Capacitive touch screen is supported by I2C via touch screen port.

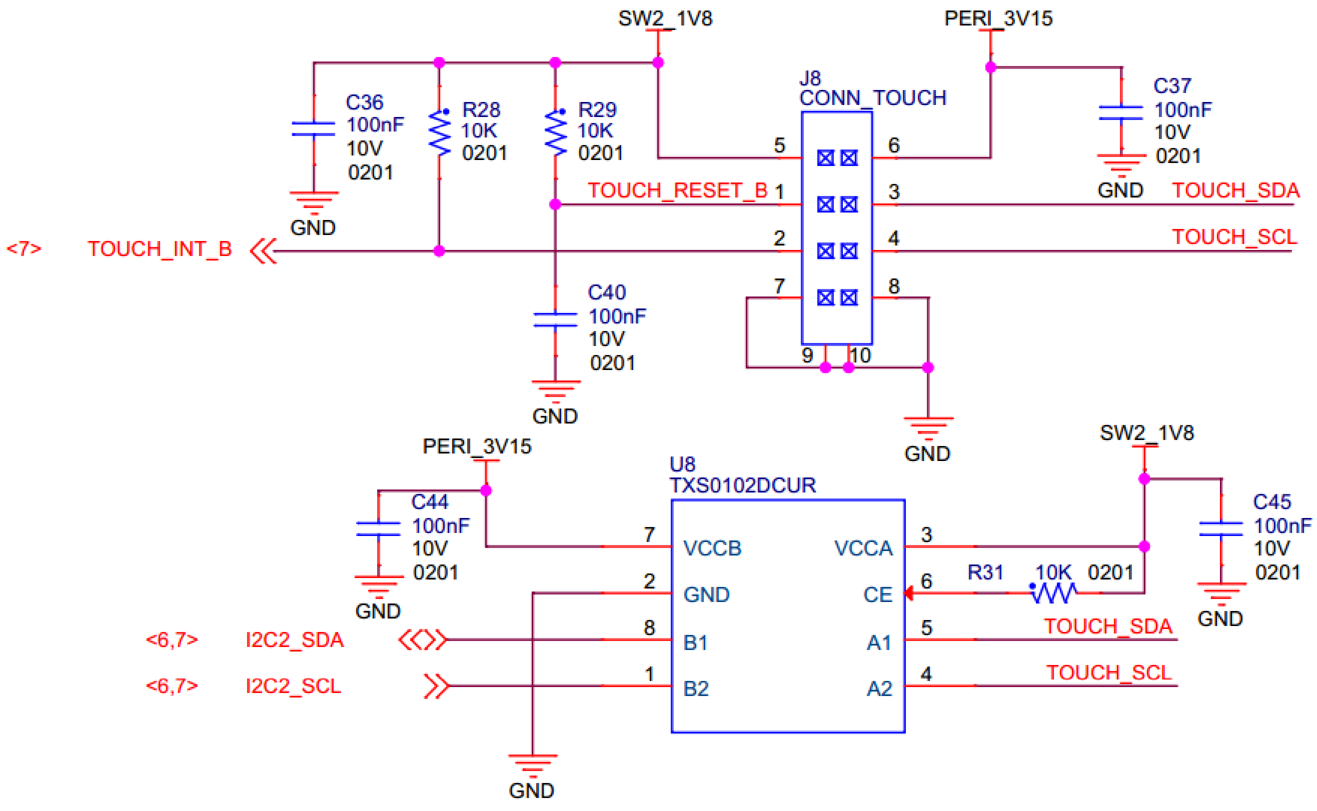
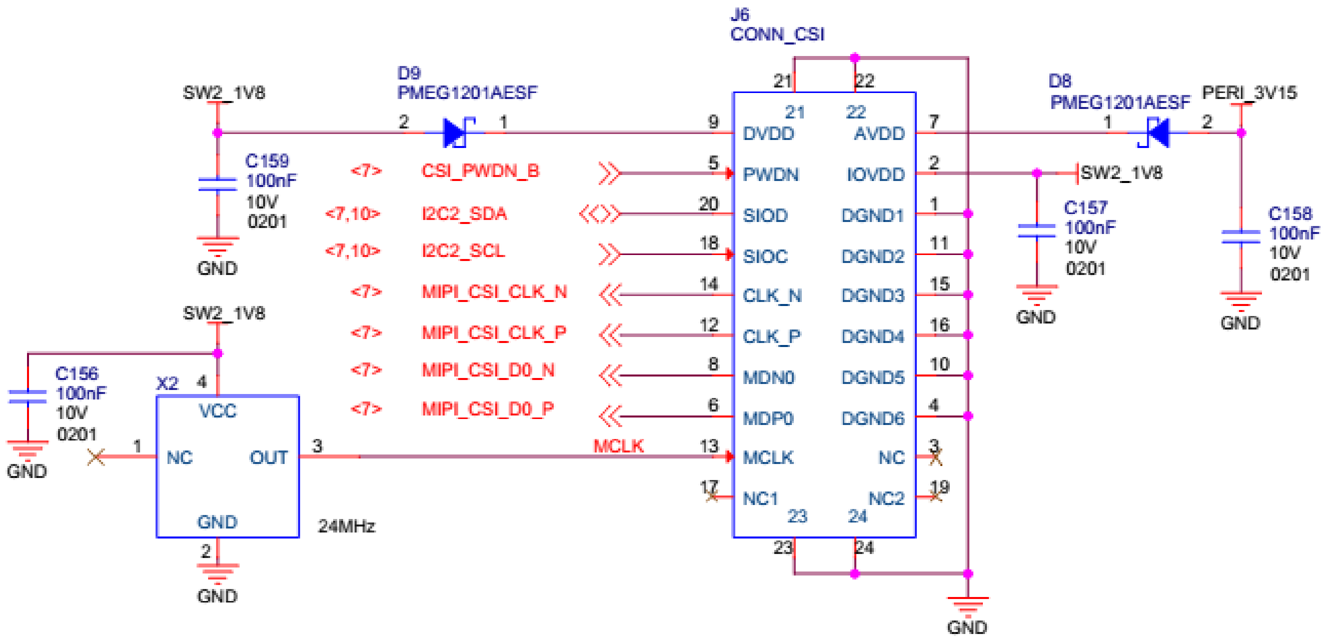


Figure 16. Touch Screen Interface

MIPI-CSI

The CPU board includes a MIPI-CSI camera connector for connecting a CSI camera module.



Note:The camera module's IIC address is 0x6C(write),0x6D(read).

Figure 17. MIPI CSI connector

2.2.5. Connectivity

The WaRP7 board provides a number of connectivity including Wi-Fi, Bluetooth, Bluetooth (BLE), and USB-OTG. There is provision for NFC as a passive tag primarily for Bluetooth pairing.

Wi-Fi/Bluetooth

The Murata Type 1DX module is an ultra-small module that includes 2.4GHz WLAN IEEE 802.11b/g/n and Bluetooth Version 4.1 plus EDR functionality. Based on Broadcom BCM4343W, the module provides high-efficiency RF front end circuits.

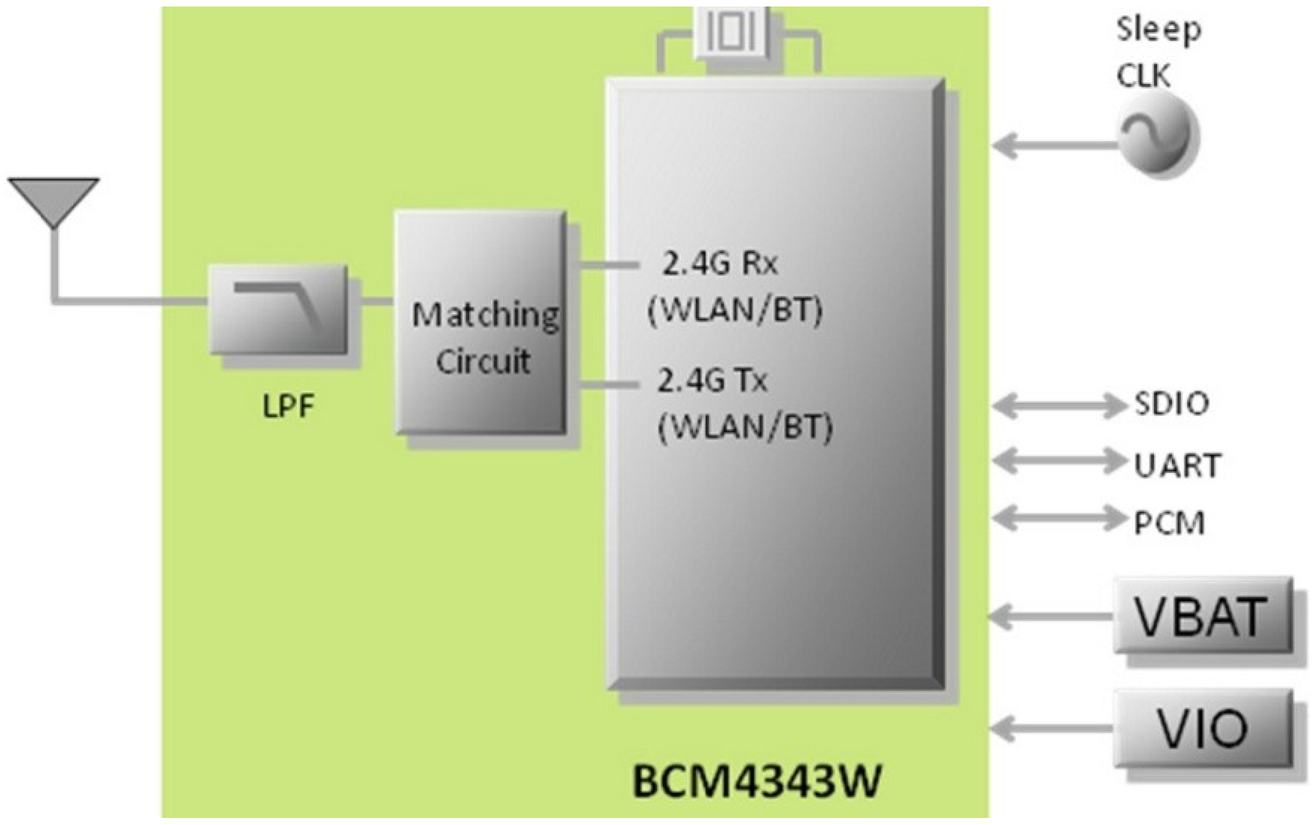


Figure 18. Murata 1DX module

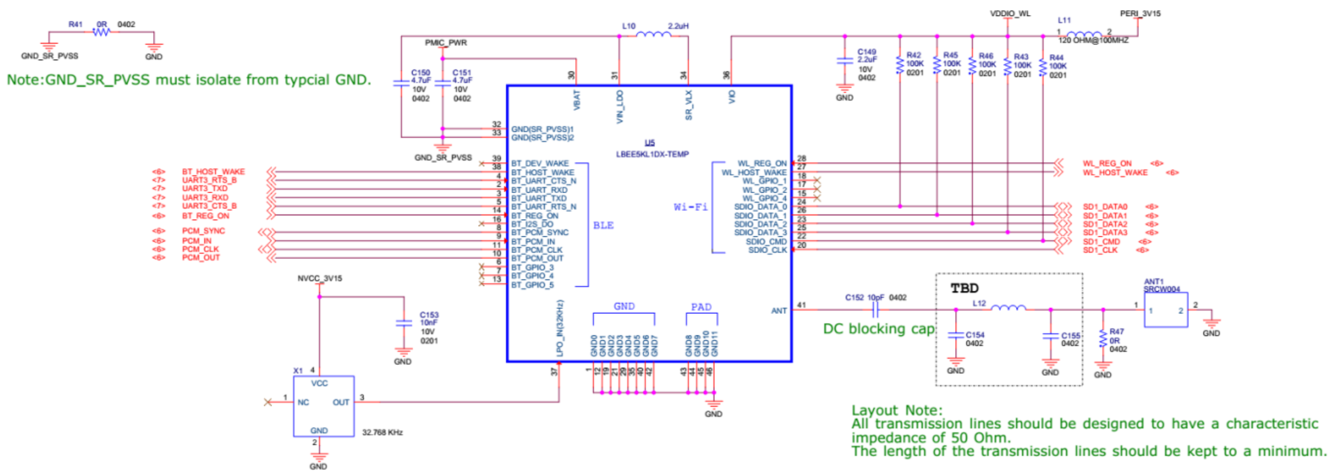

Note:GND_SR_PVSS must isolate from typcial GND.
U5
LBEE5KL1DX-TEMP
BLE
Wi-Fi
GND
PAD
TBD
DC blocking cap
32.768 KHz
Layout Note:
All transmission lines should be designed to have a characteristic impedance of 50 Ohm.
The length of the transmission lines should be kept to a minimum.

Figure 19. Design implementation of 1DX

USB-OTG

The CPU board provides an USB micro-AB connector to support USB-OTG function powered by the by USB OTG1 module on i.MX7S.

NFC

The board provides support for NFC using the NXP NT3H1101W0FHK. In addition to the passive

NFC Forum compliant contactless interface, the IC features an I2C contact interface, which can communicate with i.MX7 if NTAG I2C is powered from an external power supply. An additional externally powered SRAM mapped into the memory allows a fast data transfer between the RF and I2C interfaces and vice versa, without the write cycle limitations of the EEPROM memory.

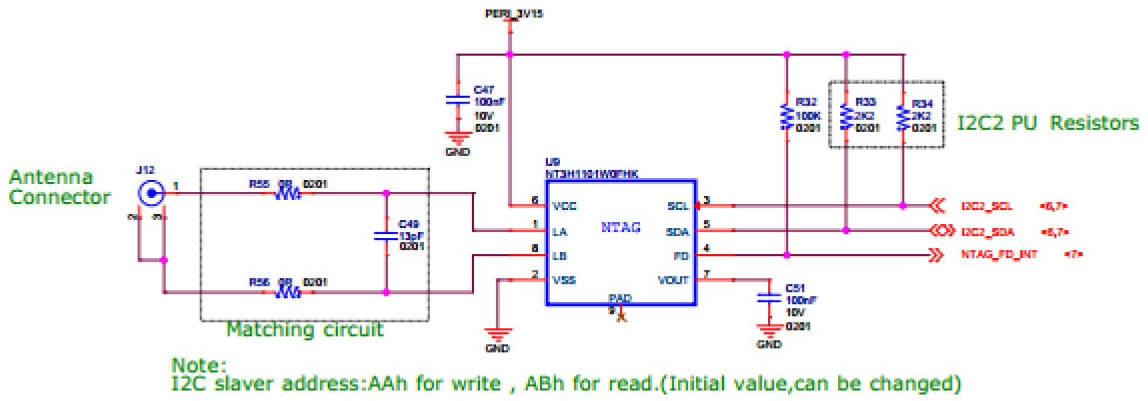


Figure 20. NFC circuitry

2.2.6. Power Management

Power Management IC

The NXP PF3000 power management integrated circuit (PMIC) features a configurable architecture that supports numerous outputs with various current ratings as well as programmable voltage and sequencing. This enables the PF3000 to power the core processor, external memory and peripherals to provide a single-chip system power solution.

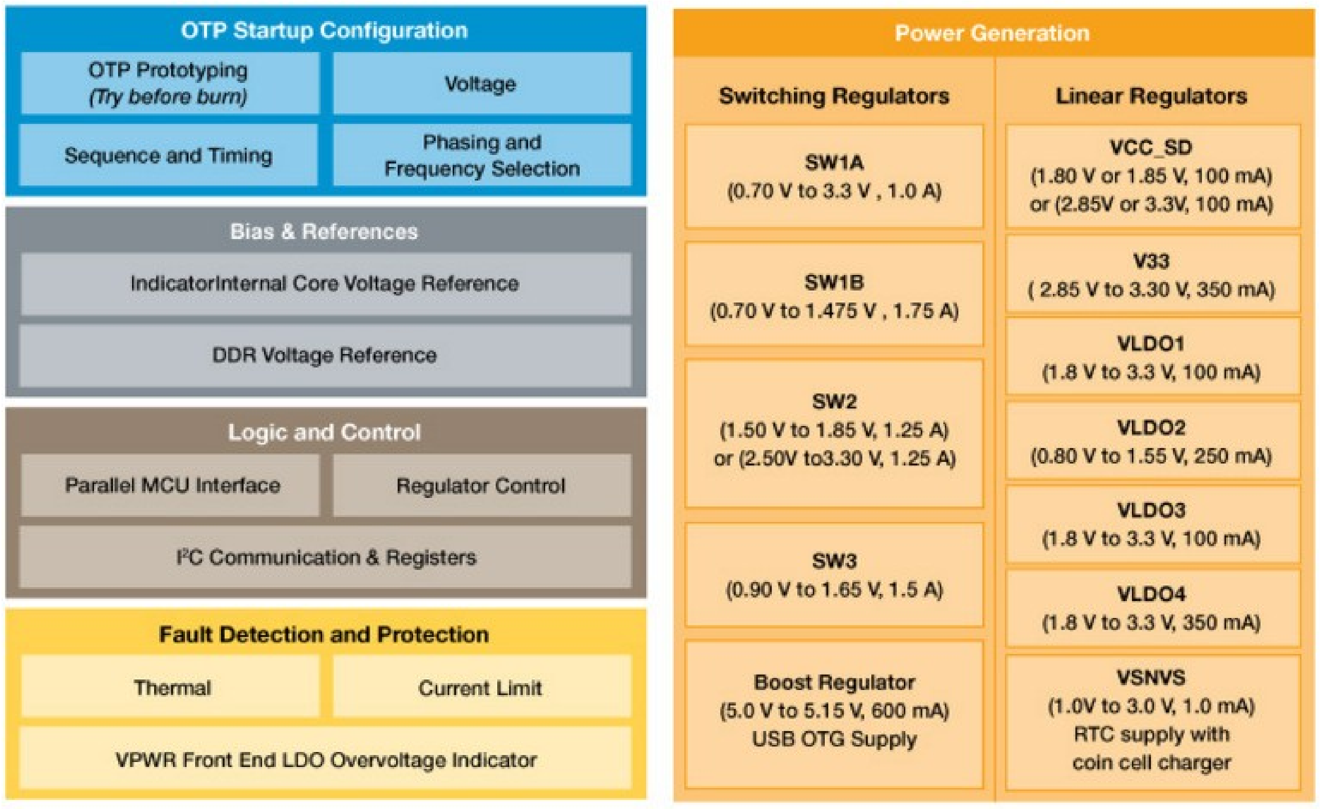

OTP Startup Configuration
OTP Prototyping (Try before burn)
Voltage
Sequence and Timing
Phasing and Frequency Selection
Bias & References
IndicatorInternal Core Voltage Reference
DDR Voltage Reference
Logic and Control
Parallel MCU Interface
Regulator Control
I²C Communication & Registers
Fault Detection and Protection
Thermal
Current Limit
VPWR Front End LDO Overvoltage Indicator
Power Generation
Switching Regulators
Linear Regulators
SW1A (0.70 V to 3.3 V , 1.0 A)
SW1B (0.70 V to 1.475 V , 1.75 A)
SW2 (1.50 V to 1.85 V, 1.25 A) or (2.50V to3.30 V, 1.25 A)
SW3 (0.90 V to 1.65 V, 1.5 A)
Boost Regulator (5.0 V to 5.15 V, 600 mA) USB OTG Supply
VCC_SD (1.80 V or 1.85 V, 100 mA) or (2.85V or 3.3V, 100 mA)
V33 (2.85 V to 3.30 V, 350 mA)
VLDO1 (1.8 V to 3.3 V, 100 mA)
VLDO2 (0.80 V to 1.55 V, 250 mA)
VLDO3 (1.8 V to 3.3 V, 100 mA)
VLDO4 (1.8 V to 3.3 V, 350 mA)
VSNVS (1.0V to 3.0 V, 1.0 mA) RTC supply with coin cell charger

Figure 21. PF3000 Functional Block diagram

Power Tree Design

The usage of PF3000 output is as shown in [PF3000 Output Power Up Sequence and Usage] below.

Table 4. PF3000 Output Power Up Sequence and Usage

PF3000 Channel	Voltage	Power up sequence	Output Current	i.MX7 Power Rail
SW1A	1.15 V	1	1000 mA	VDD_ARM
SW1B	1.15 V	1	1750 mA	VDD_SOC
SW2	1.8 V	2	1250 mA	VDDA_1P8_IN FUSE_FSOURCE VDD_XTAL_1P8 VDD_ADC1_1P8 VDD_ADC2_1P8 VDD_TEMPSENOR _1P8
SW3	1.5 V	3	1500 mA	NVCC_DRAM NVCC_DRAM_CKE
VSNVS	3.0 V	0	1 mA	VDD_SNVS_IN
SWBST		-	600 mA	
VREFDDR		3	10 mA	DRAM_VREF
VLDO1	1.8 V	2	100 mA	VDD_LPSR_IN
VLDO2	1.2 V	-	250 mA	
VLDO3	1.8 V	2	100 mA	NVCC_GPIO1/2
VLDO4	1.8 V	-	350 mA	
V33	3.15 V	2	350 mA	NVCC_xxx VDD_USB_OTG1_3 P3_IN VDD_USB_OTG2_3 P3_IN
VCC_SD	3.15 V	3	100 mA	NVCC_SD2

The following i.MX7S power rails must use the internal LDO outputs.

Table 5. iMX7S Power Rails – Internal LDO

i.MX7S internal LDO output	i.MX7S Power Rail
VDDD_1P0_CAP	VDD_MIPI_1P0 PCIE_VP PCIE_VP_RX PCIE_VP_TX
VDDA_PHY_1P8	VDDA_MIPI_1P8 PCIE_VPH PCIE_VPH_RX PCIE_VPH_TX
VDD_1P2_CAP	USB_VDD_H_1P2

Battery Charger

The NXP BC3770 is a fully programmable switching charger with dual-path output for single-cell Li-Ion and Li-Polymer battery. The dual-path output allows mobile applications with a fully discharged battery to boot up the system.

- High efficiency and switch-mode operation reduces heat dissipation and allows higher current capability for a given package size.
- Single input with a 20V withstanding input and charges the battery with an input current up to 2A.
- Charging parameters and operating modes are fully programmable over an I2C Interface that operates up to 400 kHz.
- Highly integrated featuring OVP and Power FETs.
- Supports 1.5 MHz switching capabilities.

2.3. IO Board

2.3.1. Board Dimensions

Table 6. WaRP7 IO Board Dimensions

Length	38 mm
Width	38 mm
PCB Thickness	1.0 mm

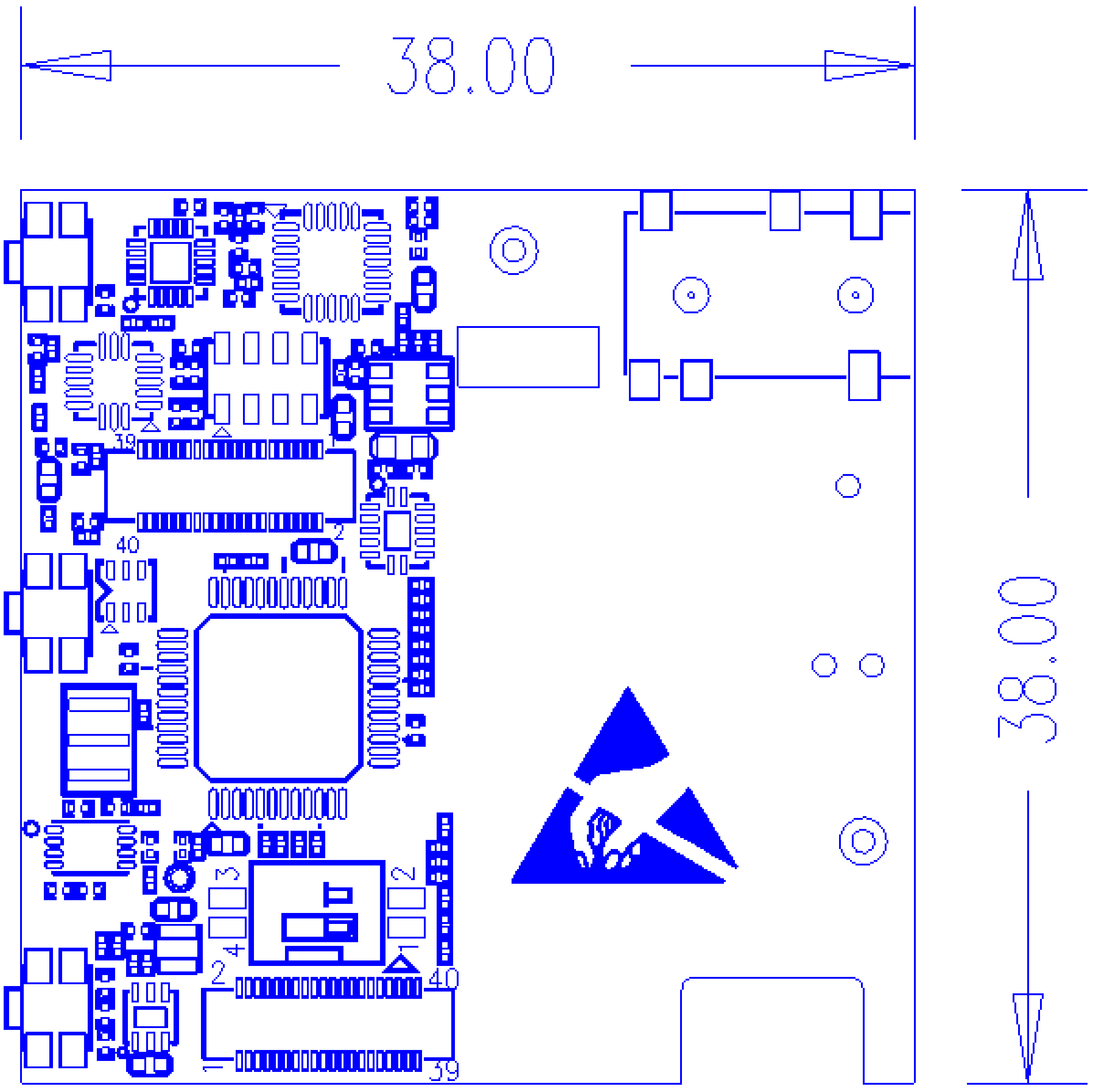


Figure 22. Board Dimensions

2.3.2. Audio

The IO board includes the NXP SGTL5000 – an ultra-low power audio codec with MIC In and Line Out capability.

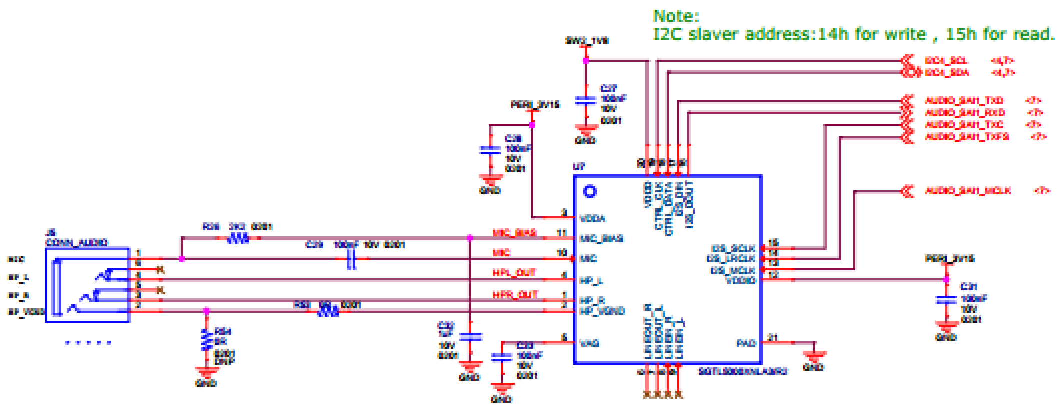


Figure 23. NXP SGTL5000 Audio Codec

2.3.3. Sensors

The WaRP7 board includes three sensors: altimeter, accelerometer and gyroscope. These three sensor chips share the I2C bus on i.MX7S. The sensors interrupts are wired to the processor as OR circuit. The software determines which device asserted the interrupt.

Altimeter

The board features NXP's MPL3115A2 precision altimeter. The MPL3115A2 is a compact piezoresistive absolute pressure sensor with an I2C interface. MPL3115 has a wide operating range of 20kPa to 110 kPa, a range that covers all surface elevations on Earth. The fully internally compensated MEMS in conjunction with an embedded high resolution 24-bit equivalent ADC provide accurate pressure [Pascals] / altitude [meters] and temperature [degrees Celsius] data.

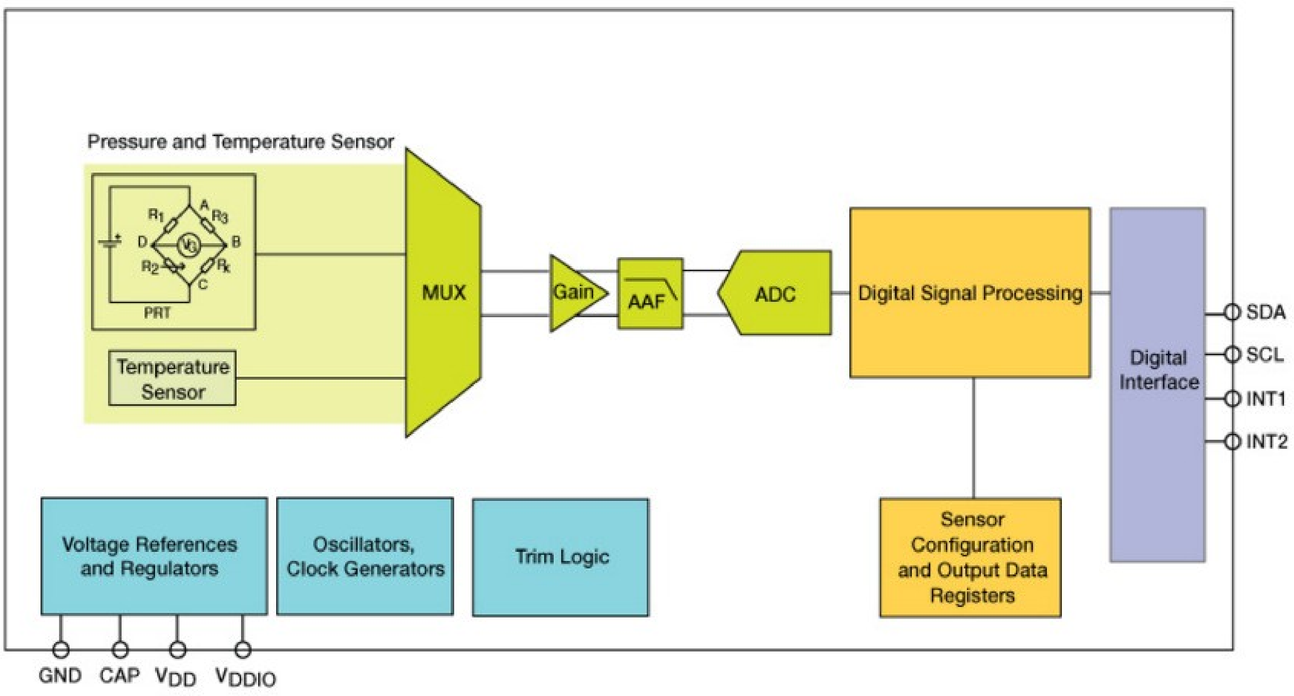

Pressure and Temperature Sensor
PRT
Temperature Sensor
MUX
Gain
AAF
ADC
Digital Signal Processing
Digital Interface
SDA
SCL
INT1
INT2
Sensor Configuration and Output Data Registers
Voltage References and Regulators
Oscillators, Clock Generators
Trim Logic
GND
CAP
VDD
VDDIO

Figure 24. MPL3115A2 Block Diagram

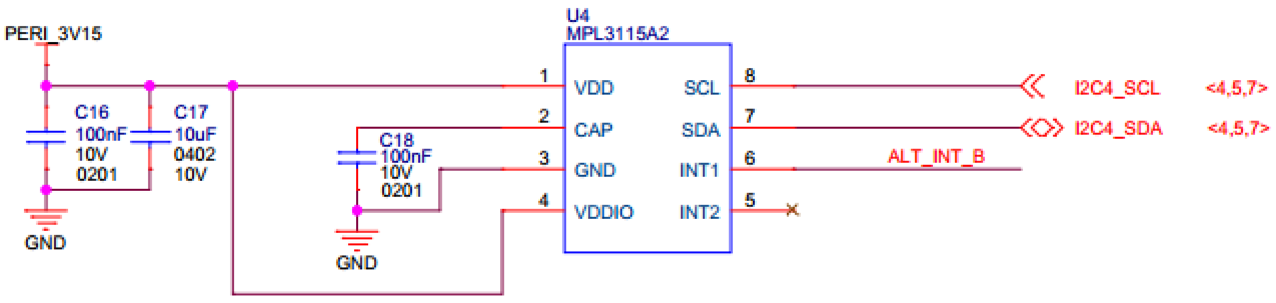


Figure 25. Altimeter schematics

Accelerometer and Magnetometer

The board also features FXOS8700CQ 6-axis sensor combines industry-leading 14-bit accelerometer and 16-bit magnetometer sensors in a small 3 x 3 x 1.2 mm QFN plastic package.

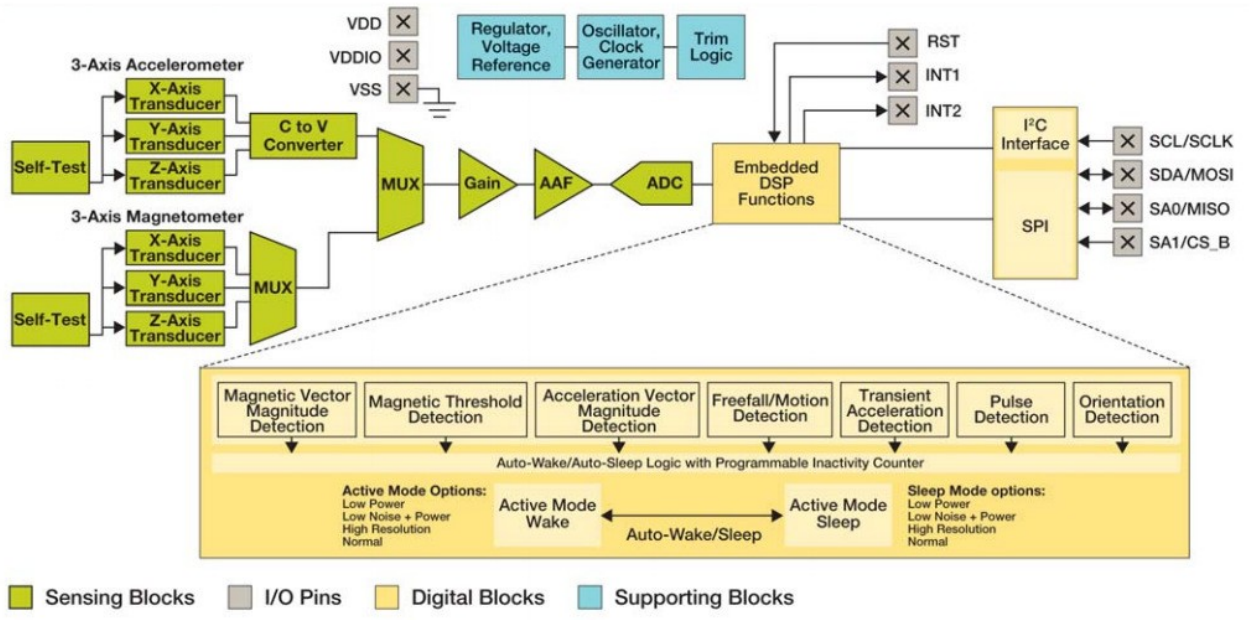


Figure 26. FXOS8700CQ – Accelerometer/Magnetometer Block Diagram

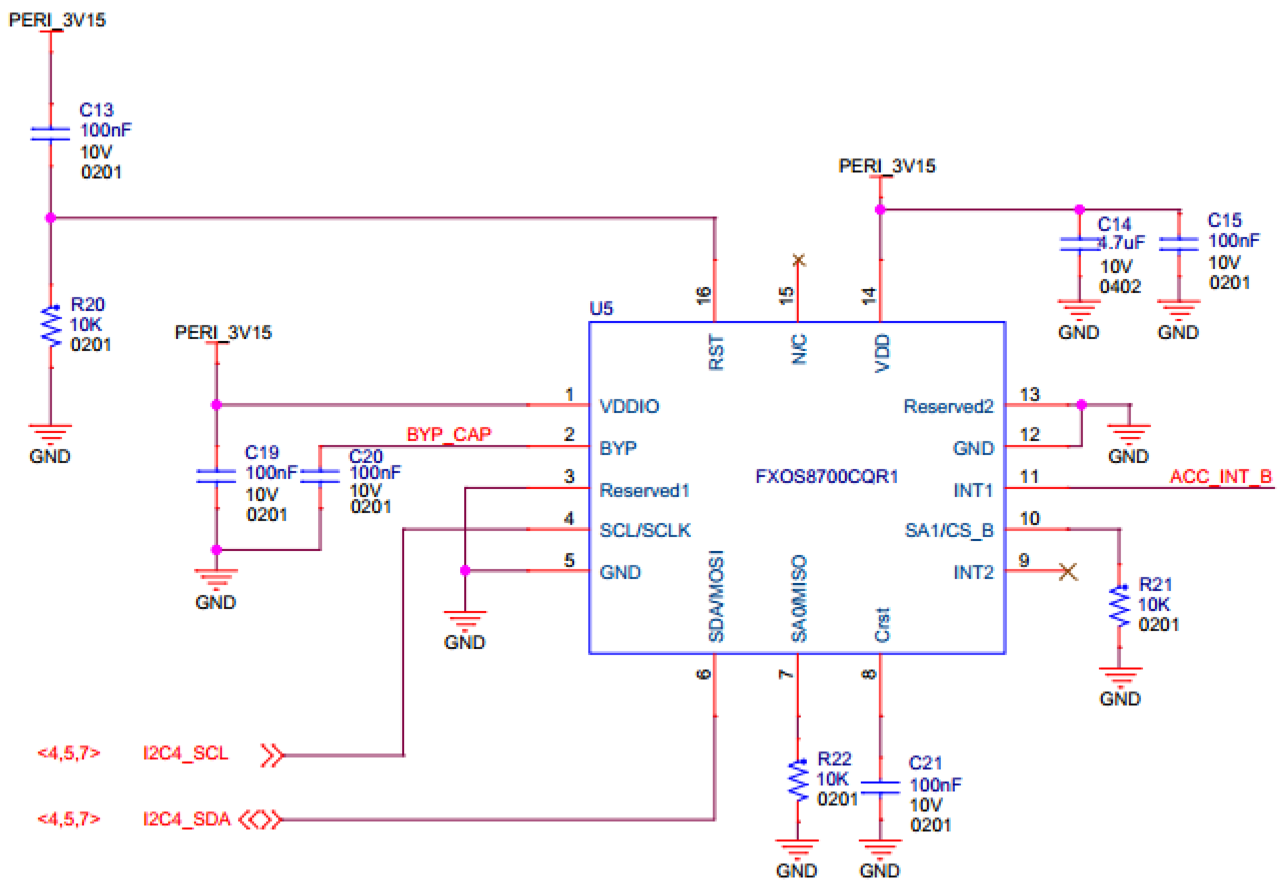


Figure 27. Accelerometer/Magnetometer schematics

Gyroscope

The IO board also features the NXP's 3-axis digital gyroscope - FXAS21002.

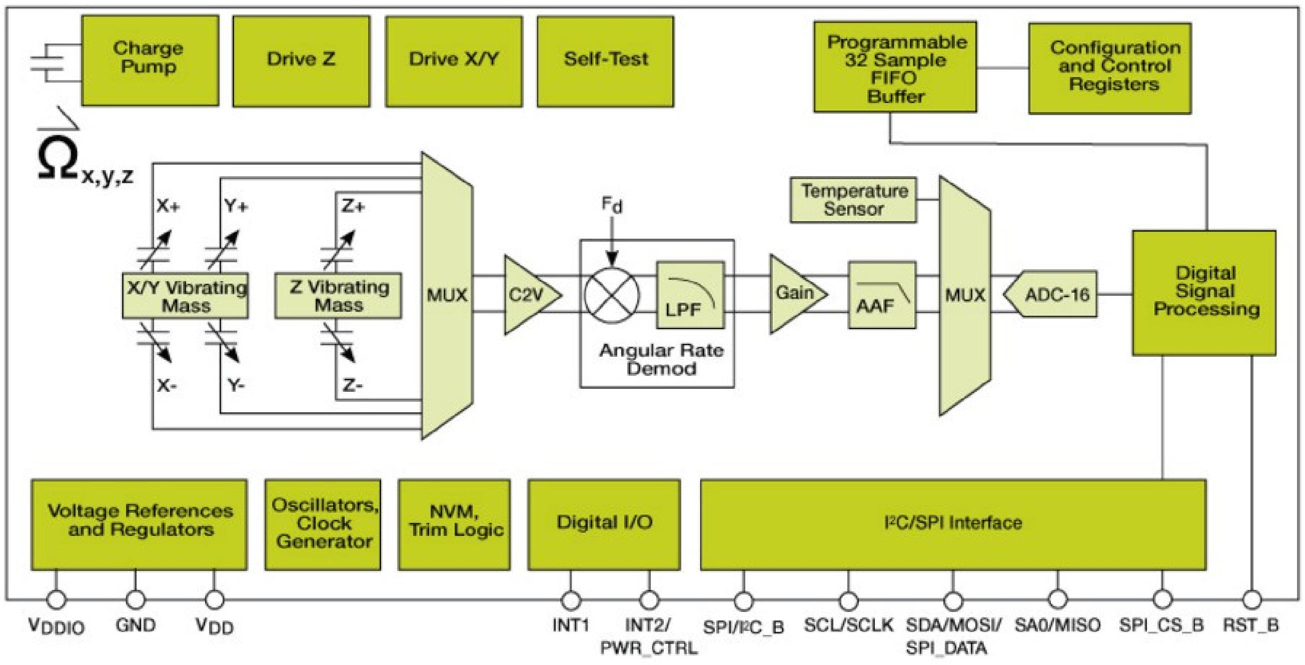


Figure 28. FXAS21002 Gyroscope Block Diagram

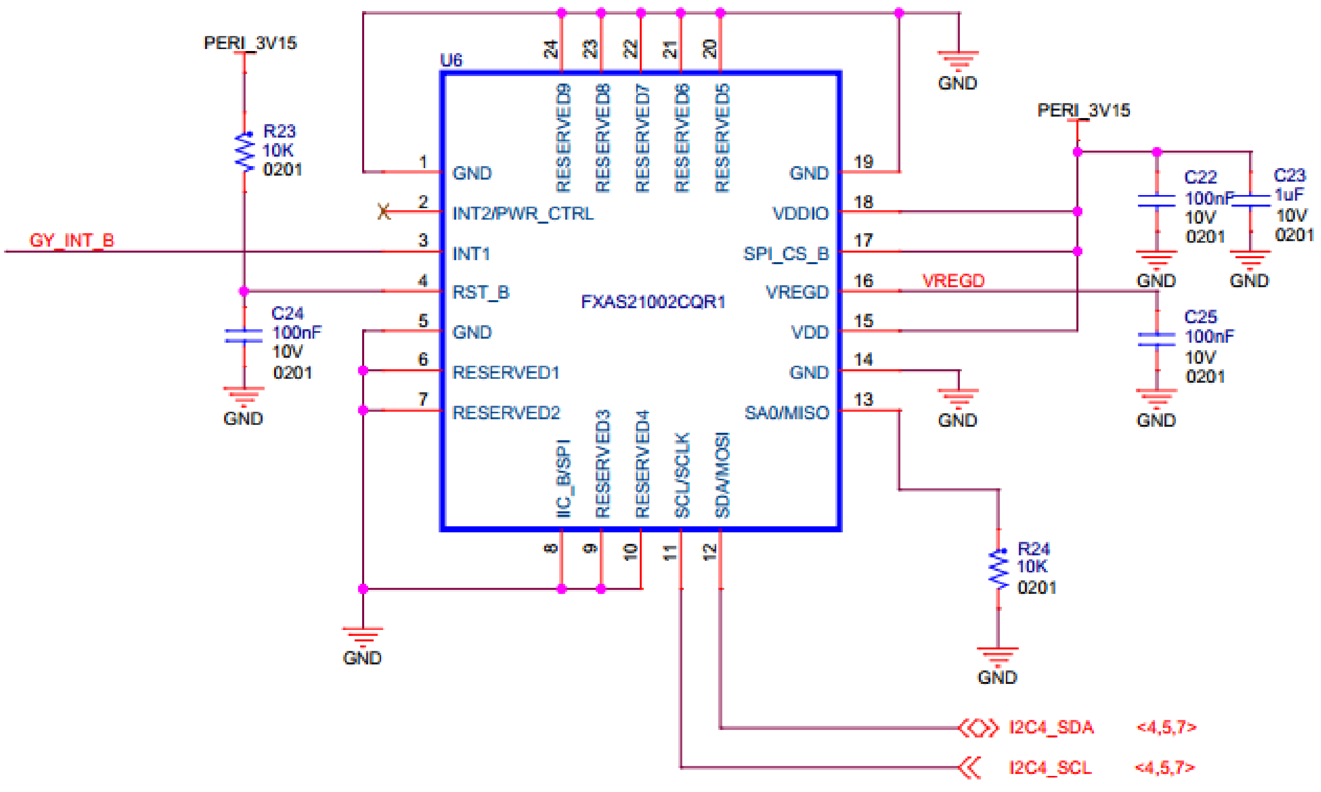


Figure 29. Gyroscope schematics

2.3.4. Peripheral Expansion Port

The board provides expansion headers compatible with the **mikroBUS™** socket connection standard for accessing the following communication modules on i.MX7S:

- I2C
- SPI
- PWM

- UART
- GPIO

3. Software Overview

The software implementation is a moving target. The table below shows the current status of the software implementation for each hardware feature.

In order to get the latest pre-built image, please go to http://freescale.github.io/#download

Table 7. Current Software implementation

PMIC (PF3000)	**YES**	**<1>**
Battery Charger (BC3770)	YES	<1>
WiFi/BLE (BCM4343W)	YES	<2>
Accelerometer (FXOS8700CQ)	YES	<1>
Magnetometer (FXOS8700CQ)	YES	<1>
Pressure Sensor(MPL3115A2)	YES	<1>
Gyroscope (FXAS21002)	YES	<1>
Audio (SGTL5000)	YES	<1>
Display	YES	<1>
Touchscreen	NO	<2>
Camera interface	NO	<2>

Legend

1. No firmware needed.
2. Firmware is needed and is not installed automatically.
3. Firmware is needed and installed.

Note:

Command shown with prefix **$** are to be run on host machine (like ubuntu, etc.).

Command shown with prefix ⇒ are to be run on u-boot prompt.

Command shown with prefix ~# are to be run on board after linux up.

4. U-Boot

4.1. Obtaining an U-Boot image

Required software on the host PC:

- imx_usb_loader: https://github.com/boundarydevices/imx_usb_loader
- dfu-util: http://dfu-util.sourceforge.net/releases/ (if you are in a Debian distribution then you can get it via libdfu-dev package)
- libusb: http://libusb.org/ (if you are in a Debian distribution then you can get it via libusb-dev and libusb-1.0-0-dev)

Clone U-Boot repository:

```
$ git clone https://github.com/Freescale/u-boot-fslc.git
```

Go inside the u-boot-fslc folder, build U-Boot for WaRP7:

```
$ cd u-boot-fslc/
$ git checkout -b <name_local_branch> origin/2016.07+fslc
$ make mrproper
$ make warp7_config
$ make
```

This will generate the U-Boot binary called u-boot.imx.

4.2. Updating an U-Boot image

Setup WaRP7 for loading U-Boot

1. Remove the CPU board from the base board then put switch 2 in the upper position
2. Reconnect the CPU board to the base board
3. Connect a USB to serial adapter between the host PC and WaRP7 (USB port on the base board)
4. Connect a USB cable between the OTG WaRP7 port and the host PC (USB port on the CPU board)

Load U-Boot via USB

Copy u-boot.imx to the imx_usb_loader folder.

```
$ sudo ./imx_usb u-boot.imx
```

Then U-Boot should start and its messages will appear in the console program.

Open a terminal program such as minicom

Use the default environment variables:

```
=> env default -f -a
=> saveenv
```

Run the DFU command:

```
=> dfu 0 mmc 0
```

Transfer u-boot.imx that will be flashed into the eMMC:

```
$ sudo dfu-util -D u-boot.imx -a boot
```

The following message should be seen on the U-Boot prompt after a successful upgrade:

#DOWNLOAD … OK Ctrl+C to exit …

Remove power from the WaRP7 board.

Put WaRP7 board into normal boot mode (put the switch 2 in the lower position)

Power up the board and the new updated U-Boot should boot from eMMC

Content based on: http://git.denx.de/?p=u-boot.git;a=blob;f=board/warp7/README

Boot partition

It can be the case that the raw partition of your WaRP7 is unlocked. Burning an image that doesn't contain U-Boot into the eMMC while on this state erases the original U-Boot. To avoid this, the user can run the command below from the U-Boot prompt:

```
=> mmc partconf 0 1 1 0
```

This command makes the raw partition read-only and prevents u-boot from being changed.

If you delete U-Boot by mistake and your board does not boot, please load U-boot from your host machine usign imx_usb_loader. Run the command above once you get to the U-boot prompt to prevent U-boot from being deleted again. You can follow the steps in the Load U-Boot via USB section to learn how to boot U-Boot from your host machine.

Inversely if your partition is locked and would like to update U-Boot as described on this chapter, you can unlock the raw partition using the command:

```
=> mmc partconf 0 1 0 0
```

5. Linux Kernel

5.1. Obtaining a Linux Kernel image

```
$ git clone https://github.com/Freescale/linux-fslc.git
$ cd linux-fslc
$ git checkout -b <name_your_branch> origin/4.1-1.0.x-imx
```

Use the commands below to compile:

```
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- mrproper
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- imx_v7_defconfig
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- zImage
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- modules
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- imx7s-warp.dtb
```

5.2. Updating a Linux Kernel image

Power up your WaRP7, U-boot prompt will come (use any serial console i.e. minicom).

Run the following command on u-boot:

```
=> ums 0 mmc 0
```

You will be able to see emmc as storage device on your computer.

Use any standard utility to make partition table. (i.e. gparted)

Create 2 partitions, a 100MB FAT32 partition and define the remaing space as an ext4 partition.

Copy zImage to the fat32 partition.

Copy imx7s-warp.dtb to the fat32 partition.

Install kernel modules to the rootfs as shown below:

```
$ sudo make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- modules_install
INSTALL_MOD_PATH=/media/<user>/<rootfs partition>/lib/
```

Note: If you don’t have a rootfs on your WaRP7 partition, you can obtain one by following the steps on **Chapter 6 The Yocto Project**

Unmount the partitions using

```
sudo umount /dev/sd<X>
```

In u-boot prompt, hit **Ctrl+C**, to cancel the mounted mmc.

Reboot your board, linux must be up and running.

6. The Yocto Project

6.1. Obtaining an image generated by Yocto Project

The first step is to prepare the host machine and download the source code.

6.1.1. Update the host needed packages

Ubuntu

```
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib \
     build-essential chrpath socat libsdl1.2-dev xterm
```

NOTE

If you use a different distribution, see http://www.yoctoproject.org/docs/current/yocto-project-qs/yocto-project-qs.html

6.1.2. Dowload the metadata

```
$ mkdir ~/bin
$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo >  ~/bin/repo
$ chmod a+x ~/bin/repo
$ PATH=${PATH}:~/bin
$ mkdir fsl-community-bsp
$ cd fsl-community-bsp
$ repo init -u https://github.com/Freescale/fsl-community-bsp-platform -b krogoth
$ repo sync
```

You can also download a prebuilt image from http://freescale.github.io/#download and test it on your board.

The download can take some time (such as 15 minutes) and depends on your Internet connection (please, make sure your proxy does allow your to download from external sources).

After the download is completed, enable your environment:

6.1.3. Building

```
$ source setup-environment build
```

NOTE

Please read the EULA and only press **y** if you accept it.

After the environment is setup you have the following files:

user@b19406-2:/code/yocto/master/build2$ tree

```
$ tree
.
└── conf
    ├── bblayers.conf
    ├── local.conf
    ├── local.conf.sample
    └── templateconf.cfg
```

Change file `conf/local.conf` to configure the build system to target the WaRP7 machine as the following example:

```
MACHINE ??= 'imx7s-warp'
DISTRO ?= 'poky'
PACKAGE_CLASSES ?= "package_rpm"
EXTRA_IMAGE_FEATURES ?= "debug-tweaks"
USER_CLASSES ?= "buildstats image-mklibs"
PATCHRESOLVE = "noop"
BB_DISKMON_DIRS = "\
    STOPTASKS,${TMPDIR},1G,100K \
    STOPTASKS,${DL_DIR},1G,100K \
    STOPTASKS,${SSTATE_DIR},1G,100K \
    STOPTASKS,/tmp,100M,100K \
    ABORT,${TMPDIR},100M,1K \
    ABORT,${DL_DIR},100M,1K \
    ABORT,${SSTATE_DIR},100M,1K \
    ABORT,/tmp,10M,1K"
PACKAGECONFIG_append_pn-qemu-native = " sdl"
PACKAGECONFIG_append_pn-nativesdk-qemu = " sdl"
CONF_VERSION = "1"

DL_DIR ?= "${BSPDIR}/downloads/"
ACCEPT_FSL_EULA = "1"
```

NOTE

The WaRP7 machine name is **imx7s-warp**

After configuring the Yocto Project to use WaRP7 machine, you can build any desired image, such as:

```
$ bitbake fsl-image-machine-test
```

The first build can take several hours (depending on your machine). When it completes the result can be found on `tmp/deploy/image/imx7s-warp` as show in the following example:

```
$ ls -l tmp/deploy/images/imx7s-warp/
total 263084
-rw-r--r-- 1 user user 67108864 Ago  8 11:58 core-image-base-imx7s-warp-
20160808141615.rootfs.ext4
-rw-r--r-- 1 user user     9568 Ago  8 11:58 core-image-base-imx7s-warp-
20160808141615.rootfs.manifest
-rw-r--r-- 1 user user 83886080 Ago  8 11:58 core-image-base-imx7s-warp-
20160808141615.rootfs.sdcard
-rw-r--r-- 1 user user 67108864 Ago  8 12:55 core-image-base-imx7s-warp-
20160808155513.rootfs.ext4
-rw-r--r-- 1 user user     9568 Ago  8 12:55 core-image-base-imx7s-warp-
20160808155513.rootfs.manifest
-rw-r--r-- 1 user user 83886080 Ago  8 12:55 core-image-base-imx7s-warp-
20160808155513.rootfs.sdcard
lrwxrwxrwx 1 user user       53 Ago  8 12:55 core-image-base-imx7s-warp.ext4 -> core-
image-base-imx7s-warp-20160808155513.rootfs.ext4
lrwxrwxrwx 1 user user       57 Ago  8 12:55 core-image-base-imx7s-warp.manifest ->
core-image-base-imx7s-warp-20160808155513.rootfs.manifest
lrwxrwxrwx 1 user user       58 Ago  8 12:55 core-image-base-imx7s-warp.sdcard.gz ->
core-image-base-imx7s-warp-20160808155513.rootfs.sdcard.gz
-rw-rw-r-- 2 user user   804917 Ago  8 10:28 modules--4.1-1.0.x+git0+b8fb01d418-r0-
imx7s-warp-20160808132254.tgz
lrwxrwxrwx 1 user user       67 Ago  8 10:28 modules-imx7s-warp.tgz -> modules--4.1-
1.0.x+git0+b8fb01d418-r0-imx7s-warp-20160808132254.tgz
-rw-r--r-- 2 user user      294 Ago  8 12:55 README_-
_DO_NOT_DELETE_FILES_IN_THIS_DIRECTORY.txt
lrwxrwxrwx 1 user user       47 Ago  8 10:31 u-boot.imx -> u-boot-sd-
v2016.07+gitAUTOINC+ae973bc45d-r0.imx
lrwxrwxrwx 1 user user       47 Ago  8 10:31 u-boot-imx7s-warp.imx -> u-boot-sd-
v2016.07+gitAUTOINC+ae973bc45d-r0.imx
lrwxrwxrwx 1 user user       47 Ago  8 10:31 u-boot-imx7s-warp.imx-sd -> u-boot-sd-
v2016.07+gitAUTOINC+ae973bc45d-r0.imx
lrwxrwxrwx 1 user user       47 Ago  8 10:31 u-boot.imx-sd -> u-boot-sd-
v2016.07+gitAUTOINC+ae973bc45d-r0.imx
-rwxr-xr-x 2 user user   347136 Ago  8 10:31 u-boot-sd-v2016.07+gitAUTOINC+ae973bc45d-
r0.imx
lrwxrwxrwx 1 user user       66 Ago  8 10:28 zImage -> zImage--4.1-
1.0.x+git0+b8fb01d418-r0-imx7s-warp-20160808132254.bin
-rw-r--r-- 2 user user  6514048 Ago  8 10:28 zImage--4.1-1.0.x+git0+b8fb01d418-r0-
imx7s-warp-20160808132254.bin
-rw-r--r-- 2 user user    33845 Ago  8 10:28 zImage--4.1-1.0.x+git0+b8fb01d418-r0-
imx7s-warp-20160808132254.dtb
lrwxrwxrwx 1 user user       66 Ago  8 10:28 zImage-imx7s-warp.bin -> zImage--4.1-
1.0.x+git0+b8fb01d418-r0-imx7s-warp-20160808132254.bin
lrwxrwxrwx 1 user user       66 Ago  8 10:28 zImage-imx7s-warp.dtb -> zImage--4.1-
1.0.x+git0+b8fb01d418-r0-imx7s-warp-20160808132254.dtb
```

The complete image (rootfs + kernel) is **core-image-base-imx7s-warp-20160808155513.rootfs.sdcard.gz** (the numbers on image name vary depending on build date)

6.2. Steps to update the image

board steps (u-boot)

```
=> ums 0 mmc 0
```

host steps

```
$ gunzip core-image-base-imx7s-warp-20160808155513.rootfs.sdcard.gz
$ sudo dd if=core-image-base-imx7s-warp-20160808155513.rootfs.sdcard of=/dev/sdX
```

It may take few minutes. As soon as the dd command is finished, you can reboot the board.

7. FreeRTOS

FreeRTOS is a real time operating system that can be used to run applications on the WaRP7 Cortex-M4 core. This operating system allows us to create applications that leverage multi-core functionality and that could potentially help us create power efficient and better performing applications when compared to its single core counterparts.

7.1. Host machine setup

7.1.1. Setup Environment

Install "cmake" to build the demo applications:

```
$ sudo apt-get install cmake
```

7.1.2. Toolchain and source code

Download the required gcc toolchain from the Launchpad website

Extract the toolchain to any location you like and follow the steps below. The variable 'ARMGCC_DIR' is required:

```
$ export ARMGCC_DIR=<path to toolchain>/gcc-arm-none-eabi-5_4-2016q2/
```

Clone the WaRP7 FreeRTOS source code from the WaRP7 GitHub repository. You can use the following command to do so:

```
$ git clone <substitute for final repo address>
```

Move inside the `WaRP7_FreeRTOS/examples/warp7/demo_apps/` directory, the demo applications implemented for WaRP7 are located here.

In order to build each application, move into the `<demo_name>/armgcc/` directory and use the `build_all.sh` script to compile the demo.

```
$ sh ./build_all.sh
```

After running the script, a "release" and "debug" folders are created. The `<demo_name>.bin` file can be found under the "debug" and "release" folder.

In order to run this demo, place the `<demo_name>.bin` file (from the debug or release folder) into the FAT partition of your WaRP7. To open said partition on the host machine, run the following command from the U-Boot prompt:

```
$ ums 0 mmc 0
```

This commands gets all the partitions on WaRP7 mounted on your host machine. Copy the `<demo_name>.bin` file directly into the partition that should already include your zImage and .dtb file.

NOTE

> To reach the U-Boot prompt, press any key on your terminal console before the U-Boot countdown reaches zero.

7.2. Running the "hello_world" demo

To run this demo, two serial port connections need to be opened. Programs like Minicom on Linux or Putty on Windows can be used. Usually, this ports show up in Linux as `/dev/ttyUSB0` for the Cortex-A7 serial port and `/dev/ttyUSB1` for the Cortex-M4. The device numbering can change depending on the number of USB devices connected to the host machine. The convention used in the following steps is `/dev/ttyUSB0` and `/dev/ttyUSB1`.

TIP

> In Windows, each serial connection has an assigned COM# port which name would be needed for the serial console program. WaRP7 opens two consequently numbered COM ports, the lower numbered for the Cortex-A7 and the higher numbered for the Cortex-M4 core.

The first step to run the demos is to power up the WaRP7 and stop the booting process. To do so, press any key on the `/dev/ttyUSB0` console before the U-Boot count reaches zero. Once the U-Boot prompt shows up in the console, type the following commands:

```
    => fatload mmc 0:1 0x7F8000 hello_world.bin
    => dcache flush
    => bootaux 0x7F8000
```

The phrase "Hello World!" appears on the second serial console `/dev/ttyUSB1`

7.3. Other demos

The WaRP7 FreeRTOS source code comes with two other demos for different types of memories. To run the other demos, follow the same steps for building, loading the `<demo_name>.bin` and powering up the WaRP7. Substitute the commands on U-Boot with the following commands:

7.3.1. Running "hello_world_ddr" demo

The following demo stores the message "Hello World!" on the DDR memory using the Cortex-A7 core and reads it back usign the Cortex-M4 core.

```
=> fatload mmc 0:1 0x9FF00000 hello_world_ddr.bin
=> dcache flush
=> bootaux 0x9FF00000
```

The phrase "Hello World!" appears on the second serial console `/dev/ttyUSB1`

7.3.2. Running "hello_world_ocram" demo

The following demo stores the message "Hello World!" on the OCRAM memory using the Cortex-A7 core and reads it back usign the Cortex-M4 core.

```
=> fatload mmc 0:1 0x00910000 hello_world_ocram.bin
=> dcache flush
=> bootaux 0x00910000
```

The phrase "Hello World!" appears on the second serial console `/dev/ttyUSB1`

8. Use Cases

This chapter demonstrates the enablement and use of the WaRP7 peripherals. Reference code for most of the WaRP7 peripherals can be found on the WaRP7 GitHub page under the unit test repository.

8.1. Wi-Fi

WaRP7 supports the Murata 1DX Wi-Fi/BT module. On way to use this module is to enable it with the enablement firmware provided by Murata.

8.1.1. Murata’s firmware enablement

The Yocto Project WaRP7 image produces a kernel configured for this firmware and a rootfs which includes the firmware artifacts. A pre-built image resulting from the Yocto Project build can be found here. This image is an option but not a requirement for the use of this firmware. If you like to build your own image your can follow the steps on The Yocto Project chapter.

After updating your WaRP7 with a Murata's firmware enabled image, you can run the Wi-Fi unit test or the following commands to enable the WaRP7's Wi-Fi:

```
$ modprobe bcmdhd
$ ifconfig wlan0 up
$ wpa_passphrase <Network Name> <Password> > /etc/wpa_supplicant.conf
$ wpa_supplicant -B -i wlan0 -c /etc/wpa_supplicant.conf -D nl80211 &
$ udhcpc -i wlan0
```

After this steps you should be able to do a ping test to check your connection.