**FR801xH Specification**

Bluetooth Low Energy SoC

Rev V1.0



**General Description**

FR801xH is a single-chip low power Bluetooth (BLE) solution. It has the characteristics of low cost, low power consumption and less peripheral components.

The FR801xH supports a flexible memory architecture for storing Bluetooth profiles and custom application code, which can be updated over the air (OTA). The qualified Bluetooth Smart protocol stack is stored in a dedicated ROM. All software runs on the enhanced 32bit RISC CPU processor via a simple scheduler.

The FR801xH comes in 3 different versions:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Device** | **Package** | | | **Features** |
| **Type** | **Size** | **Shipment** |
| FR8012HB | SOP 16 | 10x3.9x1.5mm 1.27mm pitch | Tube | 256KB flash, 7 GPIOs, UART, IIC, SPI, PWM, ADC, I2S, LDO |
| FR8016HA | QFN 32 | 4x4x0.75mm 0.4mm pitch | Tape reel | 512KB flash, 15 GPIOs, UART, IIC, SPI, PWM, ADC, I2S, LDO, Li-Charger, Audio CODEC |
| FR8018HA | QFN 48 | 6x6x0.75mm 0.4mm pitch | Tape reel | 512KB flash, 31 GPIOs, UART, IIC, SPI, PWM, ADC, I2S, LDO, Li-Charger, Audio CODEC |

**Features:**

* Complies with Bluetooth V5.0
* -94 dBm sensitivity in 1 Mbps BLE mode
* -98 dBm sensitivity in 125 Kbps BLE mode (long range)
* +10 dBm TX power (down to -20 dBm)
* Data rates: 2 Mbps, 1 Mbps, 500 Kbps and 125 Kbps
* Single-ended antenna output (Integrated balun)
* 8 mA peak current in TX (0 dBm)
* 9.7 mA peak current in RX
* RISC 32bit processor
  + Configurable frequency: 12MHz, 24MHz, 48MHz
  + support XIP (eXecute In Place)
  + SWD debug
  + 8K cache
* Flexible power management
  + 1.8V-4.3V supply voltage range
  + Integrated Buck DC-DC converter and LDO regulators with modes
  + Fast wake-up using 32kHz internal oscillator
  + 2.7 uA at 3V in System OFF mode, no RAM retention, wake on GPIO or Timer
  + 6.1 uA at 3V in System ON mode, 48K RAM retention, wake on GPIO or Timer
* 128KB ROM, 48KB RAM and 256KB or 512KB FLASH depends on different part number
* Hardware AES-128 cryptographic engine
* TRUE Random Number Generator (TRNG)
* Digital interfaces
  + Up to 32 General Purpose I/Os
  + 4-channel 10-bit ADC
  + 2x UARTs with max 921600 baud rate
  + 2x IIC controllers at up to 1MHz
  + SPI master controllers at up to 24Mbps
  + I2S master interface
  + PDM interface with HW sample rate converter
  + 6-channel PWM output
  + 2x timer with 16 bit counter
  + 16-bit audio codec with max 50mW PA out
  + 16 x external interrupt source
* Power management unit
  + 20 row x 8 column Matrix keyboard scanner with debouncing
  + Watchdog
  + 2 x RTC alarm with 32 bit counter
  + 1 x Quadrature decoder
  + 3-channel PMU PWM output
  + Battery charge module, with max 258mA charge current
  + Analog LDO output with max 120mA driver current
* Friendly development environment with provided SDK
* Embedded multi-protocols and profiles in ROM

Applications:

* Advanced computer peripherals
  + Mouse
  + Keyboard
* Advanced wearable devices
  + Health/fitness sensor and monitor devices
* Internet of things (IOT)
  + Smart home sensors and controllers
  + Industrial IOT sensors and controllers
* Interactive entertainment devices
  + Remote controllers
  + Gaming controllers

**CONTENTS**

[1 System Overview - 6 -](#_Toc33020801)

[1.1 Functional Block Diagram - 6 -](#_Toc33020802)

[1.2 System Blocks - 6 -](#_Toc33020803)

[1.3 Power Domains and Modes - 8 -](#_Toc33020804)

[1.3.1 Power Domains - 8 -](#_Toc33020805)

[1.4 Power Modes - 9 -](#_Toc33020806)

[2 Package and Pin Information - 10 -](#_Toc33020807)

[2.1 Package - 10 -](#_Toc33020808)

[2.2 Package Physical Dimensions - 11 -](#_Toc33020809)

[2.3 Pins Description - 13 -](#_Toc33020810)

[3 Power Management - 17 -](#_Toc33020811)

[3.1 Overview - 17 -](#_Toc33020812)

[3.2 PMU Controller - 17 -](#_Toc33020813)

[3.3 PMU Interrupt Controller - 17 -](#_Toc33020814)

[3.4 Sleep and Wakeup Timing - 18 -](#_Toc33020815)

[3.4.1 Timeout Wakeup - 18 -](#_Toc33020816)

[3.4.2 Asynchronous Event Wakeup - 19 -](#_Toc33020817)

[3.5 Reference Block Description - 20 -](#_Toc33020818)

[4 Clock generation and Reset - 21 -](#_Toc33020819)

[4.1 Clock Tree - 21 -](#_Toc33020820)

[4.2 Oscillators - 22 -](#_Toc33020821)

[5 UART - 23 -](#_Toc33020822)

[5.1 General description - 23 -](#_Toc33020823)

[5.2 Feature list - 23 -](#_Toc33020824)

[5.3 Description of design - 24 -](#_Toc33020825)

[5.3.1 Block diagram - 24 -](#_Toc33020826)

[5.3.2 Block details - 24 -](#_Toc33020827)

[5.4 Register description - 26 -](#_Toc33020828)

[5.4.1 Register and memory mapping summary table - 26 -](#_Toc33020829)

[5.4.2 Register - 26 -](#_Toc33020830)

[5.4.3 Configuration program flow - 31 -](#_Toc33020831)

[6 SSP - 32 -](#_Toc33020832)

[6.1 Overview - 32 -](#_Toc33020833)

[6.2 Feature list - 32 -](#_Toc33020834)

[6.3 Functional description - 32 -](#_Toc33020835)

[6.3.1 Block diagram - 32 -](#_Toc33020836)

[6.3.2 Block details - 33 -](#_Toc33020837)

[6.3.3 SSP operation - 35 -](#_Toc33020838)

[6.4 Register - 37 -](#_Toc33020839)

[6.4.1 Register summary table - 37 -](#_Toc33020840)

[6.4.2 Register description - 37 -](#_Toc33020841)

[7 IIC - 40 -](#_Toc33020842)

[7.1 Overview - 40 -](#_Toc33020843)

[7.2 Feature list - 40 -](#_Toc33020844)

[7.3 Block diagram - 40 -](#_Toc33020845)

[7.4 IIC protocols - 40 -](#_Toc33020846)

[7.4.1 Byte format and timing - 40 -](#_Toc33020847)

[7.4.2 7-bit Addressing - 41 -](#_Toc33020848)

[7.4.3 10-bit Addressing - 42 -](#_Toc33020849)

[7.4.4 Operation description - 43 -](#_Toc33020850)

[7.5 Register - 44 -](#_Toc33020851)

[7.5.1 Register summary table - 44 -](#_Toc33020852)

[7.5.2 Register description - 44 -](#_Toc33020853)

[8 PWM - 48 -](#_Toc33020854)

[8.1 Feature list - 48 -](#_Toc33020855)

[8.2 Block diagram - 48 -](#_Toc33020856)

[8.3 Register - 49 -](#_Toc33020857)

[8.3.1 Register summary table - 49 -](#_Toc33020858)

[8.3.2 Register description - 49 -](#_Toc33020859)

[8.4 PWM mode and operation - 50 -](#_Toc33020860)

[8.4.1 PWM mode - 50 -](#_Toc33020861)

[9 SARADC - 51 -](#_Toc33020862)

[9.1 Feature list - 51 -](#_Toc33020863)

[9.2 Register - 51 -](#_Toc33020864)

[10 PDM - 54 -](#_Toc33020865)

[10.1 Overview - 54 -](#_Toc33020866)

[10.2 Feature list - 54 -](#_Toc33020867)

[10.3 Module operation - 54 -](#_Toc33020868)

[10.4 Register - 55 -](#_Toc33020869)

[11 I2S - 58 -](#_Toc33020870)

[11.1 Overview - 58 -](#_Toc33020871)

[11.2 Feature list - 58 -](#_Toc33020872)

[11.3 Operations - 58 -](#_Toc33020873)

[11.3.1 Mode - 58 -](#_Toc33020874)

[11.3.2 BCLK - 58 -](#_Toc33020875)

[11.3.3 Frame Clock - 59 -](#_Toc33020876)

[11.3.4 Audio Interface Timing - 59 -](#_Toc33020877)

[11.4 Register - 60 -](#_Toc33020878)

[12 TIMER - 63 -](#_Toc33020879)

[12.1 Overview - 63 -](#_Toc33020880)

[12.2 Feature list - 63 -](#_Toc33020881)

[12.3 Timer Block Diagram - 63 -](#_Toc33020882)

[12.4 Register - 63 -](#_Toc33020883)

[12.5 Operation Description - 64 -](#_Toc33020884)

[13 EFUSE - 65 -](#_Toc33020885)

[13.1 Register - 65 -](#_Toc33020886)

[14 FRSPIM(freq-chip designed spi master controller) - 67 -](#_Toc33020887)

[15 PMU (Power management unit) - 68 -](#_Toc33020888)

[15.1 PIN Configuration - 68 -](#_Toc33020889)

[15.2 Clock and Reset - 71 -](#_Toc33020890)

[15.3 Interrupt - 72 -](#_Toc33020891)

[15.4 Debounce(Anti-Shake) - 75 -](#_Toc33020892)

[15.5 Watchdog - 77 -](#_Toc33020893)

[15.6 RTC - 77 -](#_Toc33020894)

[15.7 Matrix Keyboard - 79 -](#_Toc33020895)

[15.8 Quadrature decoder - 81 -](#_Toc33020896)

[15.9 Calibration - 84 -](#_Toc33020897)

[15.10 PWM - 86 -](#_Toc33020898)

[16 CODEC - 89 -](#_Toc33020899)

[16.1 Overview - 89 -](#_Toc33020900)

[16.2 Feature list - 89 -](#_Toc33020901)

[16.3 Register - 89 -](#_Toc33020902)

[17 SYSTEM REGS - 97 -](#_Toc33020903)

[17.1 Register - 97 -](#_Toc33020904)

[17.2 Port IO MUX - 103 -](#_Toc33020905)

[18 GPIO - 104 -](#_Toc33020906)

[18.1 General purpose ports - 104 -](#_Toc33020907)

[18.2 Register - 104 -](#_Toc33020908)

[18.3 GPIO output - 105 -](#_Toc33020909)

[18.4 GPIO input - 106 -](#_Toc33020910)

[19 EXTI(external interrupt) - 107 -](#_Toc33020911)

[19.1 Register - 107 -](#_Toc33020912)

[20 BLE Core - 110 -](#_Toc33020913)

[21 Memory Map - 111 -](#_Toc33020914)

[22 Electrical Characteristics - 112 -](#_Toc33020915)

[22.1 Absolute Maximum Ratings - 112 -](#_Toc33020916)

[22.2 Recommended Operating Conditions - 112 -](#_Toc33020917)

[22.3 IO Input/Output Electrical Logical Characteristics - 112 -](#_Toc33020918)

[22.4 Internal Resistor Characteristics - 112 -](#_Toc33020919)

[22.5 Audio CODEC - 113 -](#_Toc33020920)

[22.6 Crystal oscillator - 113 -](#_Toc33020921)

[22.7 BT Characteristics - 113 -](#_Toc33020922)

[22.8 Power Consumption - 114 -](#_Toc33020923)

[23 Revision History - 115 -](#_Toc33020924)

# System Overview

## Functional Block Diagram



## System Blocks

The FR801xH contains the following blocks:

**RISC CPU**. The processor has a 32-bit instruction set that implements a superset of 16 and 32-bit instructions to maximize code density and performance. It is used for implementing the higher layers of the Bluetooth Low Energy protocol. It is accompanied by a powerful cache controller which can minimize flash wait states when fetching instructions.

**BLE 5.0 Core**. This is the baseband hardware accelerator for the Bluetooth Low Energy protocol.

**ROM**. This is a 128KB ROM containing the Bluetooth Low Energy protocol stack as well as the boot code sequence.

**Exchange RAM**. This is a 8KB SRAM used for data exchange between firmware and the BLE baseband.

**SRAM**. This is a 48KB SRAM used for data storage.

**Flash Cache RAM**. This is a 8KB data RAM used primarily by the cache controller. The cache controller executes directly from external QSPI FLASH, thus reducing accesses to FLASH.

**UART**. Asynchronous serial interface with a FIFO of 32 bytes depth, baud rate vary from 4800 to 921600.

**SPI**. Serial peripheral interface with a FIFO of 128 bytes depth and 8-bit wide. Max bus frequency is 24MHz.

**I2C**. This is Master I2C interfaces used for sensors and/or host MCU communication. It includes a RX FIFO of 8-bit width, 8 bytes depth and a TX FIFO of 8-bit width, 10 bytes depth. Max bus frequency is 2MHz.

**QSPI Controller**. Interface to a Quad SPI FLASH device, with max 24MHz bus frequency.

**SARADC**. Differential successive approximation register analog-to-digital converter. It supports up to eight external analog input channels, 10-bit width and 1MHz sample rate.

**Radio transceiver**. This block implements the RF part of the Bluetooth Low Energy protocol at 2.4GHz.

**Clock generator**. This block is responsible for the clocking of the system. It contains a 24MHz crystal oscillators which is used for the active mode of the system. There is also a 62.5 kHz oscillator (RC62.5K) with precision (< 300 ppm). The RCX oscillator can be used as a sleep clock to improve the power dissipation, while reducing the bill of materials of the system.

**Timers**. 2 separate 16-bit counter Timer.

**PWM.** PWM module circuit implements pulse width modulation wave output. 6-channels digital PWM output module with super high up to (1/48M) resolution. Also there are another 3–channels PMU PWM output module, which has 1/(PMU system clk) resolution.

**Keyboard scanner**. This circuit implements scanning and debouncing of a keyboard matrix and generates an interrupt upon a configurable action without the need of CPU.

**AHB/APB bus**. Implements the AMBA Lite version of the AHB and APB specifications.

**I2S and PDM port**. This part enables audio streaming by means of a Pulse Density Modulation (PDM) and Inter-IC Sound (I2S) interface. It supports a digital microphone, an analog microphone and MONO speaker using PDM interface and internal codec block.

I2S interface is with 64 16-bit width FIFO depth, max 24MHz bus speed, and 16 kHz/8kHz, 16-bit sample rates.

PDM interface is with 64 16-bit width FIFO depth, 1MHz/2MHz bus speed and 16 kHz/ 8kHz, 16-bit sample rates.

**Audio codec**，It consists 1-ch 16bit ∑ΔADC, 1-ch 16bit ∑ΔDAC, which samples rate is up to 48kHz. And there are internal microphone bias equal to 0.9\*CODEC power voltage , input PGA amplifier -17.25dB - 30dB gain range, output earphone PA which output power is up to 50mW inside the audio codec block.

**Power management**. A sophisticated power management circuit with a Buck DC-DC converter and several LDOs that can be turned on/off via PMU block. Extra pins are provided for supplying external devices, even when the FR801xH is in sleep/deep sleep mode.

It also comprises a Constant Current/Constant Voltage (CCCV) charger for the battery charging and a state-of-charge fuel gauge circuit. And the CCCV charger current varies from 48mA to 258mA.

This block also include one LDO output with max 125mA drive capability, and 1.8～3.5V output voltage range.

Pin pad drive capability is 12mA. The total drive capability of all pads is equal to LDO output driver capacity, which is 125mA.

A more detailed description of each of the components of the FR801xH is presented in the following sections.

## Power Domains and Modes

### Power Domains

The FR801xH comprises several different power domains, these are controlled by power switching elements, thus eliminating leakage currents by totally powering them down.

* PD Domain, power down in sleep mode
* Option PK Domain, power down or not depending on the configuration in PMU register
* PK Domain, always on domain

An illustration of the power domains on the chip block diagram is presented as below.



## Power Modes

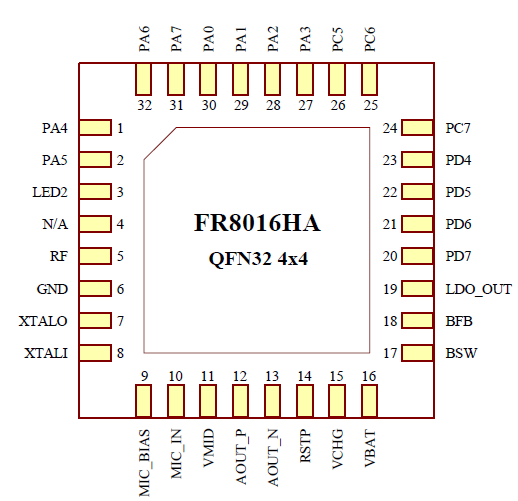
The FR801xH has four main power modes, which are distinguished by the power domains and clocks that are active:

* Active Mode, all analog blocks turn on, all clocks are available and all memories are powered up and accessible.
* Light Sleep Mode, all analog blocks turn off, all digital block’s power is switched on, but some block’s clocks can be switched off according to the Clock-Gating.
* Deep Sleep Mode, all analog blocks turn off, all digital block is power off except partial retention SRAM. The system can be waked up by the external interrupt or internal timer.
* Shutdown Mode, all analog and digital block is off. The system can be waked up by PMU due to the special pin (onkey) interrupt.

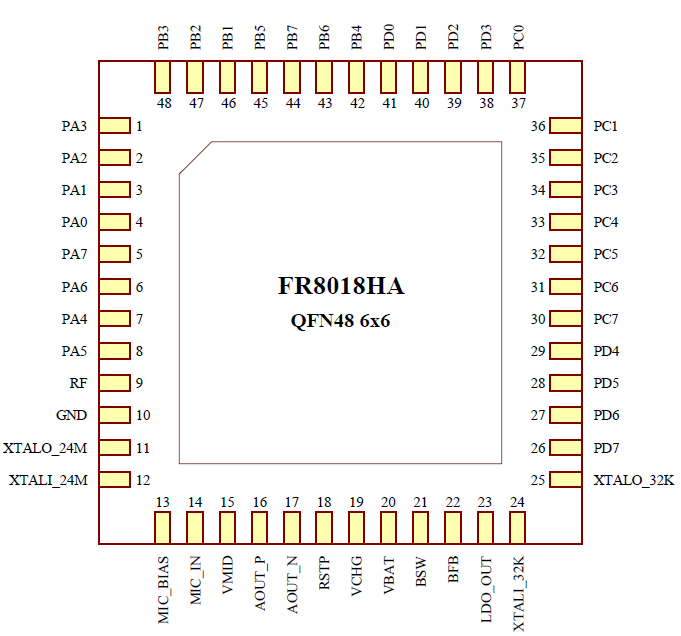
# Package and Pin Information

## Package

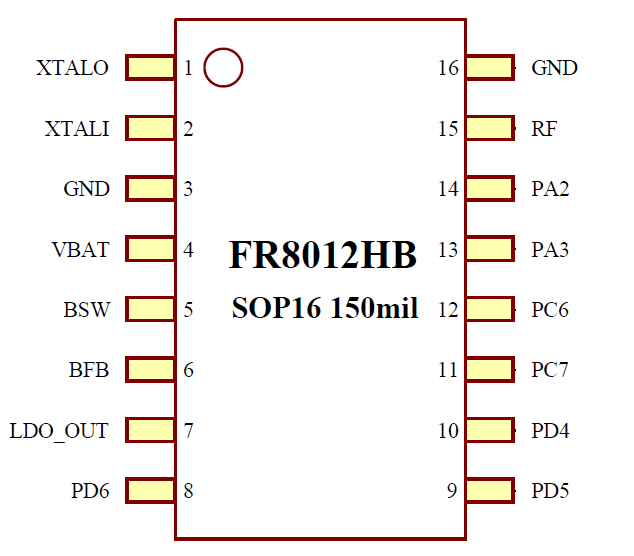
* The FR8016HA comes in a 4\*4mm QFN package with 32 pins. The pin assignment is shown as below.



* The FR8018HA comes in a 6\*6mm QFN package with 48 pins. The pin assignment is shown as below.

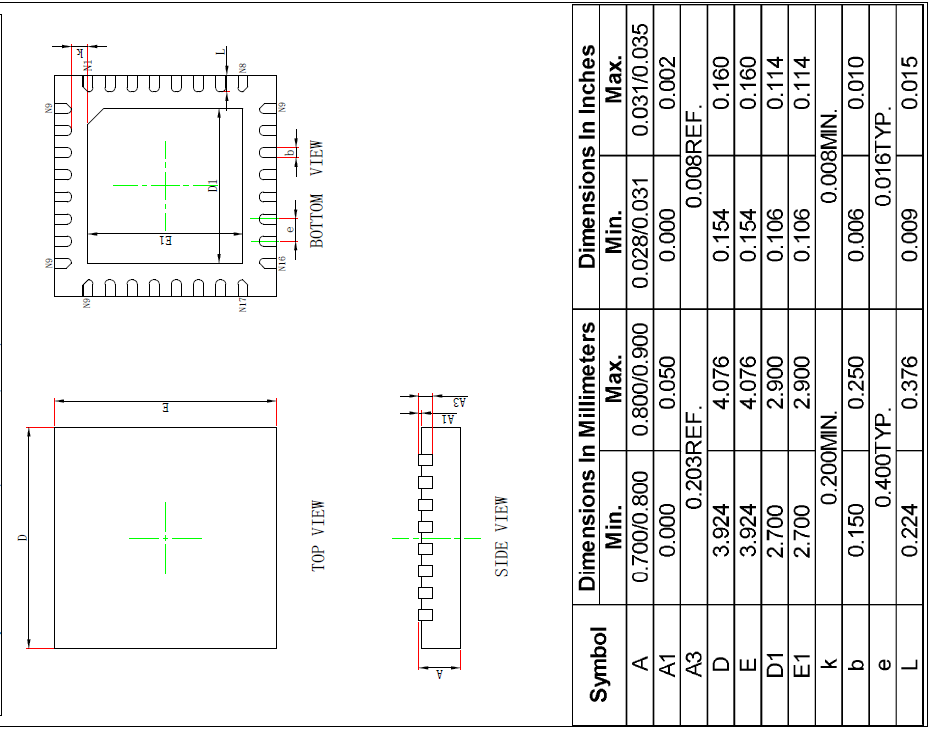


* The FR8012HB comes in a SOP package with 16 pins. The pin assignment is shown as below.

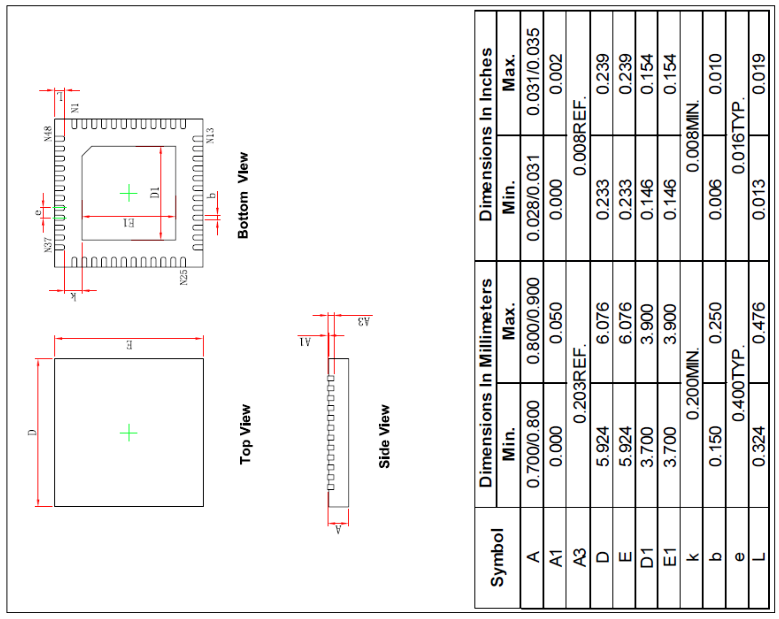


## Package Physical Dimensions

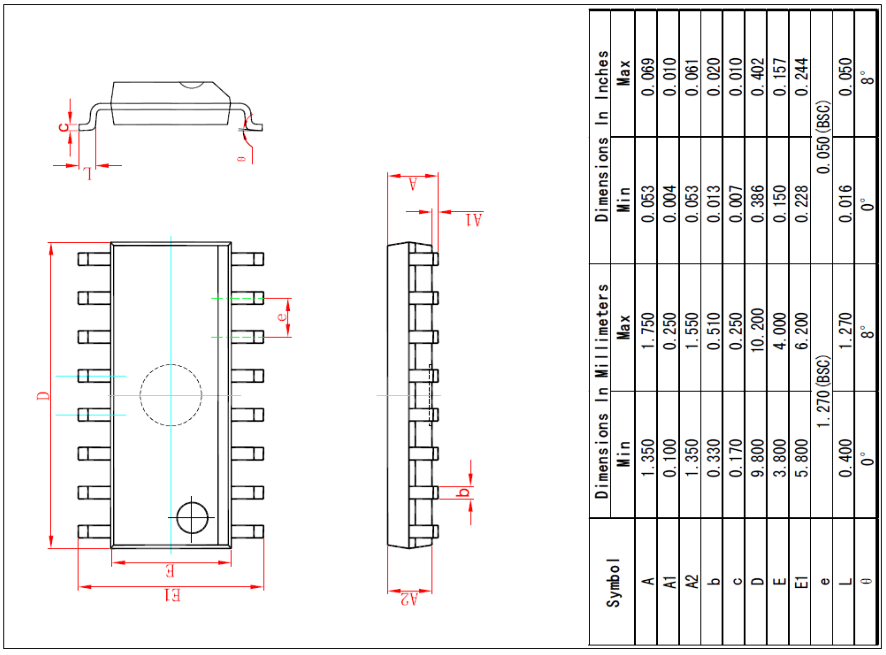
* QFN32 - 4x4 mm package outline



* QFN48 - 6x6 mm package outline



* SOP16 - 150mil package outline



## Pins Description

FR801xH is a CMOS device. Floating level on input signals will cause unstable device operation and abnormal current consumption. Pull-up or Pull-down resistors should be used appropriately for input or bidirectional pins.

|  |  |
| --- | --- |
| **Notation** | **Description** |
| I | Digital Input |
| O | Digital Output |
| AI | Analog input |
| AO | Analog output |
| IO | Bidirectional(digital) |
| OD | Open Drain |
| PWR | Power |
| GND | Ground |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Pin number** | | | **Pin name** | **Type** | **Description** |
| **FR8018HA** | **FR8016HA** | **FR8012HB** |
| 1 | 27 | 13 | PA3 | DIO | SDA1/I2SDIN/PWM3/SSPDIN/UTXD0/UTXD1/ANTCTL1/PDMDAT/PWM2 |
| 2 | 28 | 14 | PA2 | DIO | SCL1/I2SDOUT/PWM2/SSPDOUT/URXD0/URXD1/ANTCTL0/PDMCLK/PWM3 |
| 3 | 29 | - | PA1 | DIO | SDA0/I2SFRM/PWM1/SSPCSN/UTXD0/UTXD1/ANTCTL0/PDMDAT/PWM0 |
| 4 | 30 | - | PA0 | DIO | SCL0/I2SCLK/PWM0/SSPCLK/URXD0/URXD1/CLKOUT/PDMCLK/PWM1 |
| 5 | 31 | - | PA7 | DIO | SDA1/I2SDIN/PWM1/SSPDIN/UTXD0/UTXD1/ANTCTL0/PDMDAT/PWM0 |
| 6 | 32 | - | PA6 | DIO | SCL1/I2SDOUT/PWM0/SSPDOUT/URXD0/URXD1/CLKOUT/PDMCLK/PWM1 |
| 7 | 1 | - | PA4 | DIO | SCL0/I2SCLK/PWM4/SSPCLK/URXD0/URXD1/CLKOUT/PDMCLK/PWM5 |
| 8 | 2 | - | PA5 | DIO | SDA0/I2SFRM/PWM5/SSPCSN/UTXD0/UTXD1/ANTCTL1/PDMDAT/PWM4 |
| - | 3 | - | LED2 | DO | LED2 control output |
| - | 4 | - | N/A | - | Not applicable |
| 9 | 5 | 15 | RF | AIO | RF input and output |
| 10 | 6 | 3, 16 | GND | GND | Ground |
| 11 | 7 | 1 | XTALO\_24M | AO | 24MHz Crystal oscillator output |
| 12 | 8 | 2 | XTALI\_24M | AI | 24MHz Crystal oscillator input |
| 13 | 9 | ­- | MIC\_BIAS | AO | Microphone bias output |
| 14 | 10 | - | MIC\_IN | AI | Microphone input |
| 15 | 11 | - | VMID | AI | Common mode voltage |
| 16 | 12 | - | AOUT | AO | Speaker output positive |
| 17 | 13 | - | AOUT | AO | Speaker output negative |
| 18 | 14 | - | RSTP | AI | Global reset (high active) |
| 19 | 15 | - | VCHG | PWR | Charger supply input |
| 20 | 16 | 4 | VBAT | PWR | Battery positive supply input |
| 21 | 17 | 5 | BSW | AO | DC/DC output terminal |
| 22 | 18 | 6 | BFB | AI | DC/DC feedback input terminal |
| 23 | 19 | 7 | LDO\_OUT | AO | Analog linear regulator output |
| 24 | - | - | XTALI\_32K | AI | 32KHz Crystal oscillator input |
| 25 | - | - | XTALO\_32K | AO | 32KHz Crystal oscillator output |
| 26 | 20 | - | PD7 | DIO | SDA1/I2SDIN/PWM1/SSPDIN/UTXD0/UTXD1/ANTCTL1/PDMDAT/PWM0/ADC3 |
| 27 | 21 | 8 | PD6 | DIO | SCL1/I2SDOUT/PWM0/SSPDOUT/URXD0/URXD1/CLKOUT/PDMCLK/PWM1/ADC2 |
| 28 | 22 | 9 | PD5 | DIO | SDA0/I2SFRM/PWM5/SSPCSN/UTXD0/UTXD1/ANTCTL0/PDMDAT/PWM4/ADC1 |
| 29 | 23 | 10 | PD4 | DIO | SCL0/I2SCLK/PWM4/SSPCLK/URXD0/URXD1/ANTCTL0/PDMCLK/PWM5/ADC0 |
| 30 | 24 | 11 | PC7 | DIO | SDA1/I2SDIN/PWM5/SSPDIN/UTXD0/UTXD1/SWDIO/PDMDAT/PWM4 |
| 31 | 25 | 12 | PC6 | DIO | SCL1/I2SDOUT/PWM4/SSPDOUT/URXD0/URXD1/SWTCK/PDMCLK/PWM5 |
| 32 | 26 | - | PC5 | DIO | SDA0/I2SFRM/PWM5/SSPCSN/UTXD0/UTXD1/SWV/PDMDAT/PWM4 |
| 33 | - | - | PC4 | DIO | SCL0/I2SCLK/PWM4/SSPCLK/URXD0/URXD1/ANTCTL1/PDMCLK/PWM5 |
| 34 | - | - | PC3 | DIO | SDA1/I2SDIN/PWM3/SSPDIN/UTXD0/UTXD1/SWV/PDMDAT/PWM2 |
| 35 | - | - | PC2 | DIO | SCL1/I2SDOUT/PWM2/SSPDOUT/URXD0/URXD1/SWV/PDMCLK/PWM3 |
| 36 | - | - | PC1 | DIO | SDA0/I2SFRM/PWM1/SSPCSN/UTXD0/UTXD1/SWV/PDMDAT/PWM0 |
| 37 | - | - | PC0 | DIO | SCL0/I2SCLK/PWM0/SSPCLK/URXD0/URXD1/SWV/PDMCLK/PWM1 |
| 38 | - | - | PD3 | DIO | SDA1/I2SDIN/PWM3/SSPDIN/UTXD0/UTXD1/WLANRX/PDMDAT/PWM2 |
| 39 | - | - | PD2 | DIO | SCL1/I2SDOUT/PWM2/SSPDOUT/URXD0/URXD1/WLANTX/PDMCLK/PWM3 |
| 40 | - | - | PD1 | DIO | SDA0/I2SFRM/PWM1/SSPCSN/UTXD0/UTXD1/BLERX/PDMDAT/PWM0 |
| 41 | - | - | PD0 | DIO | SCL0/I2SCLK/PWM0/SSPCLK/URXD0/URXD1/BLETX/PDMCLK/PWM1 |
| 42 | - | - | PB4 | DIO | SCL0/I2SCLK/PWM4/SSPCLK/URXD0/URXD1/CLKOUT/PDMCLK/PWM5 |
| 43 | - | - | PB6 | DIO | SCL1/I2SDOUT/PWM2/SSPDOUT/URXD0/URXD1/ANTCTL1/PDMCLK/PWM3 |
| 44 | - | - | PB7 | DIO | SDA1/I2SDIN/PWM3/SSPDIN/UTXD0/UTXD1/CLKOUT/PDMDAT/PWM2 |
| 45 | - | - | PB5 | DIO | SDA0/I2SFRM/PWM5/SSPCSN/UTXD0/UTXD1/ANTCTL0/PDMDAT/PWM4 |
| 46 | - | - | PB1 | DIO | SDA0/I2SFRM/PWM1/SSPCSN/UTXD0/UTXD1/BLERX/PDMDAT/PWM0 |
| 47 | - | - | PB2 | DIO | SCL1/I2SDOUT/PWM2/SSPDOUT/URXD0/URXD1/WLANTX/PDMCLK/PWM3 |
| 48 | - | - | PB3 | DIO | SDA1/I2SDIN/PWM3/SSPDIN/UTXD0/UTXD1/WLANRX/PDMDAT/PWM2 |

# Power Management

## Overview

The FR801xH has a complete integrated power management unit (PMU) which comprises a Single Inductance Single Output (SISO) DC-DC converter, various LDOs for the different power domains of the system, a Constant Current Constant Voltage (CCCV) charger for battery recharging, a charge detection Circuit, an interrupt controller, on-off logic, BLE sleep counter, RTC, watch dog, keyscan, QDEC, PWM, calibration block and GPIO monitor.

The system diagram of the analog Power Management Unit (PMU) is presented in the following diagram.



## PMU Controller

This block controls the DC-DC converter, RC OSC, backup Voltage, various LDO and interrupt status from PMU interrupt controller. The Block generates timing logic according to the actual analog on-off timing. The timing information can be set by the PMU Register.

## PMU Interrupt Controller

This block is used to enable or disable interrupt for asynchronous event and output interrupt status signal. Also, this block can generate ‘VBAT\_OK’, ‘VBATOR’ and analog ‘ON-KEY’ signals based on the first power-on event on VBAT. Furthermore, It generates first power-on signal, power-off interrupt signal and power-on signal in shutdown mode.

The PMU interrupt controller includes following interrupt source:

* Calibration: calibration done interrupt
* Wakeup\_lp: BLE sleep timer wakeup interrupt (also in baseband sleep interrupt)
* RTC\_ALARMA: RTC alarm A interrupt
* RTC\_ALARMB: RTC alarm B interrupt
* KEYSCAN: Keyscan block valid key pressed or released
* Onkey\_on/off: Onkey power on/off interrupt
* WDT: Watchdog interrupt
* Charger: Plug in, pull off charger, battery full and insufficient power interrupt
* Ulvd\_off: Ultra low power poweroff interrupt
* GPIO: GPIO interrupt if GPIO is controlled by the PMU
* QDEC: Quadrature decoder interrupt
* OTP: Over-temperature interrupt

Above-mentioned interrupt can be forbidden or cleared through related PMU registers. Details about the PMU interrupt can be found in **PMU register** section.

## Sleep and Wakeup Timing

In FR801xH, Sleep and wakeup function is realized by BLE baseband timer. In order to reduce power consumption, BLE baseband clock is in PD domain, and be woke up by RC clock. When the system enter sleep, baseband, CPU and digital logic are switched off; retention memory stores the SOC registers. BLE baseband sleep timing also support BLE event schedule and asynchronous interrupt wakeup.

Main feature about the baseband sleep timing is shown as below:

* Baseband clock retention before sleep
* Baseband clock resume when wake up
* Support asynchronous interrupt wakeup
* Generate Power control signals

There are two modes based on BLE sleep timing, Timeout wakeup and Asynchronous event wakeup. The details are described in the following sections.

### Timeout Wakeup

Before the system goes to sleep, it should set sleep time, and then start the BLE sleep timer. When the timer is reached, the PMU will generate sleep end signal to trigger wakeup interrupt. Actually, PMU will output ‘OSC\_EN’ signal and power up related analog block before it generate sleep end signal.

The timing diagram is shown as below:



Note:

* At t5, start recover system power
* At t7, stop sleep timer, resume BLE clock
* At t8, process the timeout interrupt triggered at t7, compensate the BLE clock according to real sleep time ‘N’
* TWOSC = t7- t5, analog power on preparing time

### Asynchronous Event Wakeup

The PMU block will power on the analog block if there is an asynchronous wakeup event triggered in the sleep mode. When it happens, the BLE sleep timer will keep accumulating until additional TWEXT lowpow clock passed. Then it will generate sleeping ending signal used to trigger wakeup interrupt.



Note:

* At t5, wakeup\_req event which will trigger system to power on
* At t7, stop BLE sleep timer and resume the BLE clock
* At t8, process the timeout interrupt triggered at t7, compensate the BLE clock according to real sleep time ‘N - X’
* TWEXT = t7-t5, analog power on preparing time

## Reference Block Description

The PMU controller blocks like WDT, QDEC, PWM, KEYSCAN, RTC and so on. These blocks are described in the **PMU register** section and can be controlled by corresponding registers.

# Clock generation and Reset

## Clock Tree

The system’s clocks is described below.



The above diagrams depict all possible clock sources and all different divisions and multiplexing paths for each block’s clock. Most of logic blocks are driven by 48MHz clock. CLK\_OSC (24MHz) is for fixed frequency logic block.

## Oscillators

**Crystal oscillators:**

For facility design, the external crystal oscillator is fixed at 24MHz.Base on it, another 48MHz clock is generated for higher soc running frequency. By the way, for lower system power, there is no PLL inside.

**RC oscillators:**

The RC oscillator is designed for system sleep & shutdown mode. Its frequency is 62.5KHz. Calibration should be done before use it as BLE sleep clock counter.

# UART

The FR801xH contains two uart block, UART0 and UART1 without flow control.

## General description

This block performs serial to parallel conversion of data received from a peripheral to processor, and also parallel to serial conversion of a processor data for transmitting to a peripheral. It’s an APB slave device.

## Feature list

* Full compatibility with UART industry standard 16550
* 32-Byte depth FIFOs for both Rx/Tx
* Programmable baud rate generator, baud rate range varies from 4800 to 921600
* Independently receiver clock input
* Prioritized and independently controlled interrupts
* Fully programmable serial interface characteristics:

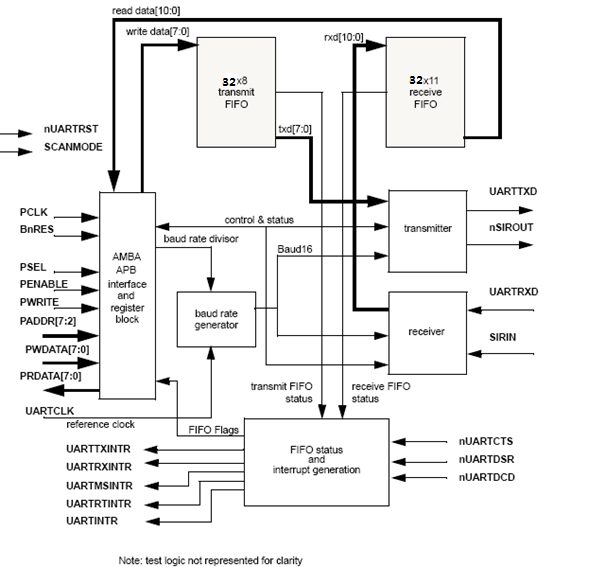
- 5-,6-,7- or 8-bit characters

- Even-, Odd-, or No-Parity bit generation and detection

- 1-,1.5- or 2-Stop bit generation

## Description of design

### Block diagram



### Block details

**AMBA APB interface**

The AMBA APB interface generates read and write decodes for accesses to status/control registers and transmit/receive FIFO memories. The AMBA APB is a local secondary bus which provides a low-power extension to the higher bandwidth *Advanced High-performance Bus* (AHB), or *Advanced System Bus* (ASB), within the AMBA system hierarchy. The AMBA APB groups narrow-bus peripherals to avoid loading the system bus and provides an interface using memory mapped registers which are accessed under programmed control.

**Register block**

The register block stores data to be written or read across the AMBA APB interface.

**Baud rate generator**

The baud rate generator contains free-running counters which generate the baud rate x16 clocks, Baud16, and the IrLPBaud16 signal. Baud16 provides timing information for UART transmit and receive control. Baud16 is a stream of pulses with a width of one UARTCLK clock period and a frequency of sixteen times the baud rate. IrLPBaud16provides timing information to generate the pulse width of the IrDA encoded transmit bit stream when in low-power mode.

**Transmit FIFO**

The transmit FIFO is an 8-bit wide, 32-byte depth, first-in, first-out memory buffer. CPU data written across the APB interface is stored in the FIFO until read out by the transmit logic. The transmit FIFO can be disabled to act as a one-byte holding register.

**Receive FIFO**

The receive FIFO is an 11-bit wide, 32-byte depth, first-in, first-out memory buffer. Received data, and corresponding error bits, are stored in the receive FIFO by the receive logic until read out by the CPU across the APB interface. The FIFO can be disabled to act as a one-byte holding register.

**Transmit logic**

The transmit logic performs parallel-to-serial conversion on the data read from the transmit FIFO. Control logic outputs the serial bit streams beginning with a start bit, data bits, least significant bit (LSB) first, followed by parity bit, and then stop bits according to the programmed configuration in control registers.

**Receive logic**

The receive logic performs serial-to-parallel conversion on the received bit stream after a valid start pulse has been detected. Parity, frame error checking and line break detection are also performed, and the data with associated parity, framing and break error bits is written to the receive FIFO.

**Interrupt generation logic**

Four individual maskable active HIGH interrupts can be generated by the UART, and a combined interrupt output will be generated as an OR function of the individual interrupt requests. The single combined interrupt may be used with a system interrupt controller, it allows use of modular device drivers which will always know where to find the interrupt source control register bits.

The individual interrupt requests could also be used with a system interrupt controller that provides masking for the outputs of each peripheral. In this way, a global interrupt service routine would be able to read the entire set of sources from one wide register in the system interrupt controller. This is attractive because the time to read from the peripheral registers is significant in a real-time system.

The peripheral supports both the above methods, since the overhead is small. The transmit and receive dataflow interrupts, UARTRXINTR and UARTTXINTR, are separated from the status interrupts so that they may be used independently by a *Direct Memory Access* (DMA) controller.

**Synchronizing registers and logic**

The UART supports both asynchronous and synchronous operation of the clocks, PCLK and UARTCLK. Synchronization registers and handshaking logic have been implemented, and are active at all times. This has a minimal impact on performance or area. Synchronization of control signals is performed on both directions of data flow, that is from the PCLK to the UARTCLK domain and vice versa.

## Register description

### Register and memory mapping summary table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Offset** | **Width** | **Access** | **Description** |
| RBR | 0x00 | 8 | RO | Receiver Buffer Register |
| THR | 0x00 | 8 | WO | Transmitter Holding Register |
| DLL | 0x00 | 8 | R/W | Divisor Latch LSB |
| IER | 0x04 | 4 | W/R | Interrupt Enable Register |
| DLM | 0x04 | 8 | R/W | Divisor Latch MSB |
| IIR | 0x08 | 8 | RO | Interrupt Identification Register |
| FCR | 0x08 | 8 | WO | FIFO Control Register |
| LCR | 0x0C | 8 | R/W | Line Control Register |
| MCR | 0x10 | 6 | R/W | Modem Control Register |
| LSR | 0x14 | 8 | R | Line Status Register |
| MSR | 0x18 | 8 | R | Modem Status Register |
| SCR | 0x1c | 8 | R/W | Scratch Register |

### Register

**Receiver Buffer Register – RBR（offset 00h）**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bit** | **Name** | **Type** | **Default** | **Description** |
| 7:0 | RX\_DATA | R | 0x00 | Accessed only if LCR[7]=0  If FIFOs Enable (FIFO Mode): RBR indicates Receiver FIFO  If FIFOs Disable (Non-FIFO Mode): RBR indicates a 8-bit register. |

**Transmitter Holding Register – THR(offset 00h)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bit** | **Name** | **Type** | **Default** | **Description** |
| 7:0 | TX\_DATA | W | 0x00 | Accessed only if LCR[7]=0  If FIFO Enable (FIFO Mode): RBR indicates Transmitter FIFO,  If FIFO Disable (Non-FIFO Mode): RBR indicates a 8-bit register. |

**Divisor Latch LSB –DLL(offset 00h)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bit** | **Name** | **Type** | **default** | **Description** |
| 7:0 | DLL | R/W | 00 | Accessed only if LCR[7] = 1  This register is programmed for baud rate generator. Divisor  latch MSB and LSB are concatenated to form 16 bit divisor value signal |

**Interrupt Enable Register – IER(offset 04h)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bit** | **Name** | **Type** | **default** | **Description** |
| 7:4 | REV | R | 0 | These bits are always cleared (Reserved bits). |
| 3 | EMSI | R/W | 0 | “1”: Enable Modem Status Interrupt (EMSI)  “0”: Disable EMSI |
| 2 | ERLSI | R/W | 0 | “1”: Enable Receive Line Status Interrupt (ERLSI)  “0”: Disable ERLSI |
| 1 | ETI | R/W | 0 | “1”: Enable THR Empty Interrupt (ETI)  “0”: Disable ETI |
| 0 | ERDI | R/W | 0 | “1”: Enable Received Data Available Interrupt (ERDI)  “0”: Disable ERDI |

**Divisor Latch MSB –DLM(offset 04h)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bit** | **Name** | **Type** | **default** | **Description** |
| 7:0 | DLM | R/W | 00 | Accessed only if LCR[7] = 1  This register is programmed for baud rate generator. Divisor  latch MSB and LSB are concatenated to form 16 bit divisor value signal |

**Interrupt Identification Register –IIR(offset 08h)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bit** | **Name** | **Type** | **Default** | **Description** |
| 7:4 | REV | R | 0 | Reserved |
| 3:1 | INT\_ID | R | 3’b000 | Interrupt ID.  011 Receiver line status  010 Receiver data available  110 Character time-out indication  001 THR empty  000 Idle status |
| 0 | INT\_ST | R | 1’b1 | 0= interrupt is pending;  1= no interrupt is pending |

**FIFO Control Register –FCR(offset 08h)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bit** | **Name** | **Type** | **Default** | **Description** |
| 7:6 | RX\_TRIGGER | W | 2’b00 | Indicate the trigger level for the receiver FIFO interrupt  00 --- 1 character in the FIFO  01 --- FIFO 1/4 full  10 --- FIFO 1/2 full  11 --- FIFO 1/8 less full |
| 5:4 | TX\_TRIGGER | W | 2’b00 | Indicate the trigger level for the transmit FIFO interrupt.  00 --- Empty FIFO  01 --- 2 character in the FIFO  10 --- FIFO 1/4 full  11 --- FIFO 1/2 full |
| 3 | DMA\_MOD | W | 0 | 0= TXRDYN,RXRDYN signal work in DMA mode 0  1= TXRDYN,RXRDYN signal work in DMA mode 1 |
| 2 | TX\_FIFO\_RST | W | 0 | When this bit is set, transmitter FIFO reset. The logic one written to this bit is self clearing. |
| 1 | RX\_FIFO\_RST | W | 0 | When this bit is set, receiver FIFO reset. The logic one written to this bit is self clearing |
| 0 | FIFO\_EN | W | 0 | 1= Enables transmitter and receiver FIFO  0= Disable and clear the Tx/Rx FIFO |

**Line Control Register –LCR(offset 0Ch)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bit** | **Name** | **Type** | **Default** | **Description** |
| 7 | DLA | R/W | 0 | Divisor Registers Access.  Bit 7 must be set to access the divisor latches(LSB&MSB) of baud generator.  Bit 7 must be cleared to access buffers(THR or IER) |
| 6 | - |  | 0 | Reserved |
| 5 | SP | R/W | 0 | Stick Parity Bit .  If bits[5:3] are 111, parity bit is checked as 0.  If bits[5:3] are 101, parity bit is checked as 1.  If bit5 is 0, stick parity is disabled |
| 4 | EVENARITY | R/W | 0 | Even Parity Enable  1: even parity is selected.  0: odd parity is selected. |
| 3 | PARITY\_EN | R/W | 0 | Parity Enable Bit  1: parity check enable  0: parity check disable |
| 2 | STOP\_BIT\_SEL | R/W | 0 | Stop Bits  0= one stop bit after data bits  1=  1.5 stop bit for a 5-bit data character,  OR 2 stop bits for a 6-,7-,or 8-bit character.    The receiver checks the first stop bit only regardless of the number of stop bits selected |
| 1:0 | WORD\_LEN | R/W | 2’b00 | Word Length  00: 5 bits  01: 6 bits  10: 7 bits  11: 8 bits |

**Modem Control Register – MCR(offset 10h)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bit** | **Name** | **Type** | **default** | **Description** |
| 7:6 | RSVD | 00 | 00 | These bits are always cleared (Reserved bits). |
| 5 | AFE | R/W | 0 | Auto Flow control Enable  “1”: Enable AFE  “0”: Disable AFE |
| 4 | LOOP | R/W | 0 | Loop Back Mode  1: enable local loop back feature for diagnostic testing of the UART.  0: disable local loop back feature |
| 3:0 | - | - | 3’b00 | Reserved |

**Line Status Register – LSR(offset 14h)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bit** | **Name** | **Type** | **Default** | **Description** |
| 7 | ERR\_FLAG | R | 0 | Error Flag  In No-FIFO mode, LSR[7] is always 0.  In FIFO mode, LSR[7] is 1 when there is at least one parity, framing or break error in the FIFO.  It is cleared after LSR is read |
| 6 | T\_EMT | R | 1 | Transmitter Empty Indicator  TEMT is set when the both THR and TSR are empty.  TEMT is cleared when either THR or TSR contains a data character. |
| 5 | T\_HRE | R | 1 | Transmitter Holding Register Empty Indicator  THRE is set when the THR is empty.  THRE is set when the THR contains at least a data character |
| 4 | BI | R | 0 | Break Interrupt Indicator  When BI is set, it indicates that received data input was held  in the low state for longer than the full transmission time.  BI is cleared every time the CPU reads the contents of the LSR.  In FIFO mode it indicates the BI status of the character on  the top of the FIFO.  Full Transmission Time = Start + Data + Parity + Stop bits |
| 3 | FE | R | 0 | Framing Error indicator  When FE is set it indicates that received character does not  have a valid stop bit.  FE is cleared every time CPU reads thecontents of the LSR.  In FIFO mode it indicates the FE status of the character on top of the FIFO. |
| 2 | PE | R | 0 | Parity Error  When PE is set it indicates that parity of received character does not match the parity selected in LCR[4].  PE is cleared every time the CPU reads the contents of LSR.  In FIFO mode it indicates the parity of the character on the top of the FIFO |
| 1 | OE | R | 0 | Overrun Error Indicator  When OE is set, it indicates that before the character in the  reg RBR was read, it was overwritten by the next character  transferred into the register.  OE is cleared every time the CPU reads the contents of the LSR.  In FIFO mode, overrun error occurs only after the FIFO is full and the next character has been completely received in the shift register.  Note：The character in the shift register is overwritten, but it is not transferred to the FIFO. |
| 0 | DR | R | 0 | Data Ready Indicator  DR is set when the complete incoming character is received and transferred into RBR or the FIFO.  It is cleared by reading all data in the FIFO or RBR. |

**Modem Status Register – MSR(offset 18)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bit** | **Name** | **Type** | **Default** | **Description** |
| 7 | DCD | R | 0 | Data Carrier Detect  When the UART is in the diagnostic test mode(LOOP),this bit is equal to the MCR bit 3(OUT2). |
| 6 | RI | R | 0 | Ring Indicator  When the UART is in diagnostic test mode, this bit is equal to MCR bit 2(RI). |
| 5 | DSR | R | 0 | Data Send Request  When the UART is in diagnostic test node, this bit is equal to MCR bit 0(DSR). |
| 4 | CTS | R | 0 | Clear To Send  When UART is in diagnostic test mode, this bit is equal to MCR bit1 |
| 3 | DCD | R | 0 | DCD Changed  It indicates that dcd has changed since it was last read by the CPU. |
| 2 | TERI | R | 0 | RI Rising Edge  When TERI is set and the modem status interrupt is enabled, a modem status interrupt is generated |
| 1 | DSR | R | 0 | DSR Changed  It indicates that DSR has changed states since last time it was read by the CPU. |
| 0 | CTS | R | 0 | CTS Changed  It indicates the CTS has changed states since last time it was Read by the CPU. |

**Scratch Register –SCR(offset 1Ch)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bit** | **Name** | **Type** | **default** | **Description** |
| 7:0 | TMP\_VAL | R/W | FF | User temporary data  This register can be used by user as a temporary value |

### Configuration program flow

Below following is only an example for configure UART to receive and transmitter data:

**Example for Receiver Program Flow (Interrupt FIFO Mode):**

1. Program the baud rate (Write FCR[7]=1 first, then program the DLL and DLM).

2. Write the LCR to program Line Control Information (word length, parity, stop and etc.)

3. Write the FCR to enable FIFO and program the Rx Trigger Level.

4. Write the Interrupt Enable Register to enable the corresponding interrupt

5. Wait for receive data from Serial Line

6. If interrupt, read the Interrupt Identification Register to judge the error interrupt or Rx Trigger Level interrupt. If error interrupt, there have error on Receive Line Data

7. If Rx Trigger Level interrupt, MCU transfer (read) the data from Rx FIFO (Read RBR)

8. If character timeout interrupt, MCU read remain data in the Rx FIFO till empty (Read Line Status Register to confirm the Rx FIFO empty or not, LSR[0]).

**Example for Transmitter Data Program Flow (Interrupt FIFO mode)**

1. Program the baud rate.

2. Write the LCR to program line control information

3. Write the FCR to enable FIFO and program Tx trigger level.

4. Write the IER to enable interrupt.

5. Wait for interrupt

6. If THRE interrupt, then write transmitter data into FIFO (Write THR)

7. Then go to 5, wait for interrupt and write Tx data into FIFO.

# SSP

The SSP is a master interface that enables synchronous serial communication witch slave peripherals.

## Overview

The SSP performs serial-to-parallel conversion on data received from a peripheral device. The CPU accesses data, control, and status information through the AMBA APB interface. The transmit and receive paths are buffered with internal FIFO memories enabling up to 128 8-bit values to be stored independently in both transmit and receive modes. Serial data is transmitted on **SSPTXD** and received on **SSPRXD**.

The SSP includes a programmable bit rate clock divider and prescaler to generate the serial output clock, **SSPCLKOUT**, from the input clock, **SSPCLK**. Bit rates are supported up to 24MHz, subject to choice of frequency for **SSPCLK**, and the maximum bit rate is determined by peripheral devices.

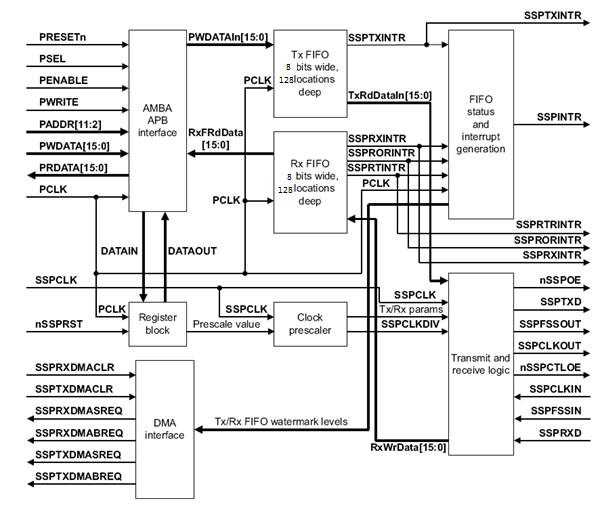
You can use the control registers SSPCR0 to program the SSP operating mode, frame format, and size.

## Feature list

* Master operation.
* Programmable clock bit rate and prescale. Bus speed varies from 500kHz to 24Mhz.
* Separate transmit and receive FIFOs with 8 bits wide, 128 locations depth.
* Independent masking of transmit FIFO, receive FIFO, and receive overrun interrupts.
* Programmable data frame size from 4 to 8 bits.

## Functional description

### Block diagram



### Block details

**AMBA APB interface**

The AMBA APB interface generates read and write decodes for accesses to status and control registers, and transmit and receive FIFO memories. The AMBA APB is a local secondary bus that provides a low-power extension to the higher bandwidth AMBA *Advanced High-performance Bus* (AHB) within the AMBA system hierarchy. The AMBA APB groups narrow-bus peripherals to avoid loading the system bus and provides an interface using memory-mapped registers, that are accessed under programmed control.

**Register block**

The register block stores data written, or to be read, across the AMBA APB interface.

**Clock prescaler**

When configured as a master, an internal prescaler, comprising two free-running reloadable serially linked counters, provides the serial output clock SSPCLKOUT. You can program the clock prescaler, using the SSPCPSR register, to divide SSPCLK by a factor of 2-254 in steps of two. By not utilizing the least significant bit of the SSPCPSR register, division by an odd number is not possible and this ensures that a symmetrical, equal mark space ratio, clock is generated. The output of the prescaler is divided again by a factor of 1-256, by programming the SSPCR0 control register, to give the final master output clock SSPCLKOUT.

**Transmit FIFO**

The common transmit FIFO is a 8-bit wide, 128-locations deep, *First-In, First-Out* (FIFO) memory buffer. CPU data written across the AMBA APB interface are stored in the buffer until read out by the transmit logic. When configured as a master, parallel data is written into the transmit FIFO prior to serial conversion, and transmission to the attached slave, through the SSPTXD pin.

**Receive FIFO**

The common receive FIFO is a 8-bit wide, 128-locations deep, first-in, first-out memory buffer. Received data from the serial interface are stored in the buffer until read out by the CPU across the AMBA APB interface. When configured as a master, serial data received through the SSPRXD pin is registered prior to parallel loading into the attached slavereceive FIFO respectively.

**Transmit and receive logic**

When configured as a master, the clock to the attached slaves is derived from a divided-down version of SSPCLK through the prescaler operations that previous sections describe. The master transmit logic successively reads a value from its transmit FIFO and performs parallel to serial conversion on it. Then, the serial data stream and frame control signal, synchronized to SSPCLKOUT, are output through the SSPTXD pin to the attached slaves. The master receive logic performs serial to parallel conversion on the incoming synchronous SSPRXD data stream, extracting and storing values into its receive FIFO, for subsequent reading through the APB interface.

**Interrupt generation logic**

The SSP generates four individual maskable, active-HIGH interrupts. A combined interrupt output is also generated as an OR function of the individual interrupt requests. You can use the single combined interrupt with a system interrupt controller that provides another level of masking on a per-peripheral basis. This enables use of modular device drivers that always know where to find the interrupt source control register bits. You can also use the individual interrupt requests with a system interrupt controller that provides masking for the outputs of each peripheral. In this way, a global interrupt controller service routine can read the entire set of

sources from one wide register in the system interrupt controller. This is attractive where the time to read from the peripheral registers is significant compared to the CPU clock speed in a real-time system.

The peripheral supports both the above methods. The transmit and receive dynamic data-flow interrupts, SSPTXINTR and SSPRXINTR, are separated from the status interrupts so that data can be read or written in response to the FIFO trigger levels.

**Synchronizing registers and logic**

The SSP supports both asynchronous and synchronous operation of the clocks, PCLK and SSPCLK. Synchronization registers and handshaking logic have been implemented, and are active at all times. This has a minimal impact on performance or area. Synchronization of control signals is performed on both directions of data flow, that is:

• from the PCLK to the SSPCLK domain

• from the SSPCLK to the PCLK domain.

### SSP operation

**Enable SSP operation**

You can either prime the transmit FIFO, by writing up to 128 8-bit values when the SSP is disabled, or permit the transmit FIFO service request to interrupt the CPU. Once enabled, transmission or reception of data begins on the transmit, **SSPTXD**, and receive, **SSPRXD**, pins.

**Clock ratios**

There is a constraint on the ratio of the frequencies of **PCLK** to **SSPCLK**. The frequency of **SSPCLK** must be less than or equal to that of **PCLK**. This ensures that control signals from the **SSPCLK** domain to the **PCLK** domain are guaranteed to get synchronized before one frame duration:

FSSPCLK <= FPCLK.

The setup and hold times on **SSPRXD**, with reference to SSPCLKIN, must be more conservative to ensure that it is at the right value when the actual sampling occurs within the SSPMS. To ensure correct device operation, **SSPCLK** must be at least 12 times faster than the maximum expected frequency of SSPCLKIN.

The frequency selected for SSPCLK must accommodate the desired range of bit clock rates. The ratio of minimum **SSPCLK** frequency to **SSPCLKOUT** maximum frequency for the master mode, it is two.

To generate a maximum bit rate of 24Mbps in the master mode, the frequency of **SSPCLK** must be at least 48MHz. With an **SSPCLK** frequency of 48MHz, the SSPCPSR register must be programmed with a value of 2, and the SCR[7:0] field in the SSPCR0 register must be programmed with a value of 0.

The minimum frequency of **SSPCLK** is governed by the following equations, both of which must be satisfied:

FSSPCLK(min) => 2 x FSSPCLKOUT(max), for master mode

The maximum frequency of **SSPCLK** is governed by the following equations, both of which must be satisfied:

FSSPCLK(max) <= 254 x 256 x FSSPCLKOUT(min), for master mode

**Programming the SSPCR0 Control Register**

The SSPCR0 register is used to:

• program the serial clock rate

• select one of the three protocols

• select the data word size, where applicable

The *Serial Clock Rate* (SCR) value, in conjunction with the SSPCPSR clock prescale divisor value, CPSDVSR, is used to derive the SSP transmit and receive bit rate from the external **SSPCLK**. The frame format is programmed through the FRF bits, and the data word size through the DSS bits. Bit phase and polarity, applicable to Motorola SPI format only, are programmed through the SPH and SPO bits.

To configure the SSP as a master, clear the SSPCR0 register master or slave selection bit, MS, to 0. This is the default value on reset. To enable the operation of the SSP, set the *Synchronous Serial Port Enable* (SSE) bit to 1.

**Bit rate generation**

The serial bit rate is derived by dividing down the input clock, **SSPCLK**. The clock is first divided by an even prescale value CPSDVSR in the range 2-254, and is programmed in SSPCPSR. The clock is divided again by a value in the range 1-256, that is 1 + SCR, where SCR is the value programmed in SSPCR0. The following equation defines the frequency of the output signal bit clock, **SSPCLKOUT**:

For example, if **SSPCLK** is 48MHz, and CPSDVSR = 2, then SSPCLKOUT has a frequency range from 93.75kHz-24MHz.

**Frame format**

Each data frame is between 4-8 bits long, depending on the size of data programmed, and is transmitted starting with the MSB. You can select the following basic frame types:

• Texas Instruments synchronous serial

• Motorola SPI

• National Semiconductor Microwire.

For all formats, the serial clock, **SSPCLKOUT**, is held inactive while the SSP is idle, and transitions at the programmed frequency only during active transmission or reception of data. The idle state of **SSPCLKOUT** is utilized to provide a receive timeout indication that occurs when the receive FIFO still contains data after a timeout period.

For Motorola SPI and National Semiconductor Microwire frame formats, the serial frame, **SSPFSSOUT**, pin is active-LOW, and is asserted, pulled-down, during the entire transmission of the frame.

For Texas Instruments synchronous serial frame format, the **SSPFSSOUT** pin is pulsed for one serial clock period, starting at its rising edge, prior to the transmission of each frame. For this frame format, both the SSP and the off-chip slave device drive their output data on the rising edge of **SSPCLKOUT**, and latch data from the other device on the falling edge.

Unlike the full-duplex transmission of the other two frame formats, the National Semiconductor Microwire format uses a special master-slave messaging technique, that operates at half-duplex. In this mode, when a frame begins, an 8-bit control message is transmitted to the off-chip slave. During this transmit, the SSP receives no incoming data. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the requested data.

**Motorola SPI frame format**

The Motorola SPI interface is a four-wire interface where the **SSPFSSOUT** signal behaves as a slave select. The main feature of the Motorola SPI format is that you can program the inactive state and phase of the **SSPCLKOUT** signal using the SPO and SPH bits of the SSPSCR0 control register.

**SPO, clock polarity**

When the SPO clock polarity control bit is LOW, it produces a steady state LOW value on the **SSPCLKOUT** pin. If the SPO clock polarity control bit is HIGH, a steady state HIGH value is placed on the **SSPCLKOUT** pin when data is not being transferred.

**SPH, clock phase**

The SPH control bit selects the clock edge that captures data and enables it to change state. It has the most impact on the first bit transmitted by either permitting or not permitting a clock transition before the first data capture edge.

When the SPH phase control bit is LOW, data is captured on the first clock edge transition.

When the SPH clock phase control bit is HIGH, data is captured on the second clock edge transition.

## Register

### Register summary table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Offset** | **Type** | **Reset** | **Width** | **Description** |
| SSPCR0 | 0x00 | RW | 0x0000 | 16 | Control register 0 |
| SSPDR | 0x08 | RW | 0x00 | 8 | Data register |
| SSPSR | 0x0c | RO | 0x03 | 5 | Status register |
| SSPCPSR | 0x10 | RW | 0x00 | 8 | Clock prescale register |
| SSPIMSC | 0x14 | RW | 0x00 | 4 | Interrupt mask set or clear register |
| SSPRIS | 0x18 | RO | 0x08 | 4 | Raw interrupt status register |
| SSPMIS | 0x1c | RO | 0x00 | 4 | Masked interrupt status register |
| SSPICR | 0x20 | WO | 0x00 | 2 | Interrupt clear register |

### Register description

**Control register –SSPCR (offset 00’h)**

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Function** |
| [15:8] | SCR | Serial clock rate. The value SCR is used to generate the transmit and receive bit rate of the SSP. The bit rate is:  where CPSDVSR is an even value from 2-254, SCR is a value from 0-255. |
| [7] | SPH | SSP\_CLK phase, applicable to Motorola SPI frame format only. |
| [6] | SPO | SSP\_CLK polarity, applicable to Motorola SPI frame format only. |
| [5:4] | FRF | Frame format:  00 Motorola SPI frame format.  01 TI synchronous serial frame format.  10 National Microwire frame format.  11 Reserved. |
| [3:0] | DSS | Data Size Select:  0000 Reserved.  0001 Reserved.  0010 Reserved.  0011 4-bit data.  0100 5-bit data.  0101 6-bit data.  0110 7-bit data.  0111 8-bit data.  Other, Reserved. |

**Data register –SSPDR (offset 08’h)**

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Function** |
| [15:0] | DATA | Transmit/Receive FIFO:  For a data size which is less than 16 bits.  Unused bits at the top are ignored by transmit logic.  Data is automatically right-justifies by receive logic |

**Status register –SSPSR (offset 0C’h)**

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Function** |
| [15:5] | - | Reserved. |
| [4] | BSY | SSP busy flag, RO:  0 SSP is idle.  1 SSP is currently transmitting or receiving a frame |
| [3] | RFF | Receive FIFO full, RO:  0 Receive FIFO is not full.  1 Receive FIFO is full. |
| [2] | RNE | Receive FIFO not empty, RO:  0 Receive FIFO is empty.  1 Receive FIFO is not empty. |
| [1] | TNF | Transmit FIFO not full, RO:  0 Transmit FIFO is full.  1 Transmit FIFO is not full. |
| [0] | TFE | Transmit FIFO empty, RO:  0 Transmit FIFO is not empty.  1 Transmit FIFO is empty. |

**Clock prescale register –SSPCPSR (offset 10’h)**

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Function** |
| [15:8] | - | Reserved. |
| [7:0] | CLK\_DVSR | Clock prescale divisor.  Must be an even number from 2-254, depending on the frequency of SSPCLK. |

**Interrupt mask set or clear register –SSPIMSC (offset 14’h)**

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Function** |
| [15:4] | - | Reserved, read as zero |
| [3] | TXIM | Transmit FIFO interrupt mask:  0 Transmit FIFO half empty or less condition interrupt is masked.  1 Transmit FIFO half empty or less condition interrupt is not masked. |
| [2] | RXIM | Receive FIFO interrupt mask:  0 Receive FIFO half full or less condition interrupt is masked.  1 Receive FIFO half full or less condition interrupt is not masked. |
| [1] | RTIM | Receive timeout interrupt mask:  0 Receive FIFO not empty and no cpu read timeout interrupt is masked.  1 Receive FIFO not empty and no cpu read timeout interrupt is not masked. |
| [0] | RORIM | Receive overrun interrupt mask:  0 Receive FIFO full condition interrupt is masked.  1 Receive FIFO full condition interrupt is not masked. |

**Raw interrupt status register –SSPRIS (offset 18’h)**

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Function** |
| [15:4] | - | Reserved, read as zero, do not modify |
| [3] | TXRIS | Gives the raw interrupt state, prior to masking, of the SSP\_TX\_INTR interrupt |
| [2] | RXRIS | Gives the raw interrupt state, prior to masking, of the SSP\_RX\_INTR interrupt |
| [1] | RTRIS | Gives the raw interrupt state, prior to masking, of the SSP\_RT\_INTR interrupt |
| [0] | RORRIS | Gives the raw interrupt state, prior to masking, of the SSP\_ROR\_INTR interrupt |

**Masked interrupt status register –SSPMIS (offset 1C’h)**

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Function** |
| [15:4] | - | Reserved, read as zero, do not modify |
| [3] | TXMIS | Transmit FIFO masked interrupt state, after masking of the SSP\_TX\_INTR |
| [2] | RXMIS | Receive FIFO masked interrupt state, after masking of the SSP\_RX\_INTR |
| [1] | RTMIS | Receive timeout masked interrupt state, after masking of the SSP\_RT\_INTR |
| [0] | RORMIS | Receive over run masked interrupt status, after masking of the SSP\_ROR\_INTR |

**Interrupt clear register –SSPICR (offset 20’h)**

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Function** |
| [15:2] | - | Reserved, read as zero |
| [1] | RTIC | Clears the SSP\_RT\_INTR interrupt |
| [0] | RORIC | Clears the SSP\_ROR\_INTR interrupt |

# IIC

## Overview

The IIC (Inter-Integrate Circuit) is a simple bi-directional 2-wire (SDA and SCL) bus. All IIC-bus compatible devices can communicate directly with each other via the IIC-bus.

## Feature list

* Support up to 2Mbps bus speed
* Support 7-bit and 10-bit address
* Support clock stretch
* Support clock divide
* Support master & slave mode
* Support interrupt mode

## Block diagram

APB

I2c

register

I2c

clk\_div

I2c

address

I2c

master

I2c

slave

I2c

rxfifo

I2c

txfifo

I2c

txfifo\_

slave

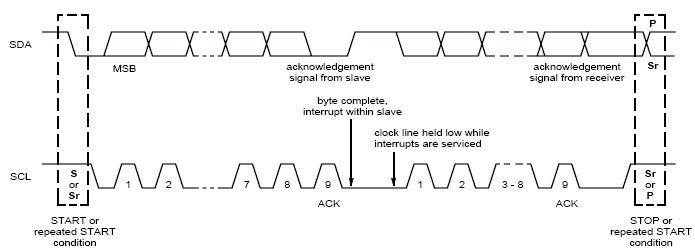
I2c

shift

## IIC protocols

### Byte format and timing

Every byte put on the SDA line must be 8-bits long. Each byte has to be followed by an acknowledge bit. Data is transferred with the most significant bit (MSB) first. Data transfer with acknowledge is obligatory. The acknowledge-related clock pulse is generated by the master. The transmitter releases the SDA line (HIGH)during the acknowledge clock pulse. The receiver must pull down the SDA line during the acknowledge clock pulse so that it remains stable LOW during the HIGH period of this clock pulse. When a slave doesn’t acknowledge the slave address, the data line must be left HIGH by the slave. The master can then generate either a STOP condition to abort the transfer, or a repeated START condition to start a new transfer.



data transfer on the I2C bus

### 7-bit Addressing

Data transfers follow the format shown in follow figure. After the START condition (S), a slave address is sent. This address is 7 bits long followed by an eighth bit which is a data direction bit (R/W) - a ‘zero’ indicates a transmission (WRITE), a ‘one’ indicates a request for data (READ). A data transfer is always terminated by a STOP condition (P)generated by the master. However, if a master still wishes to communicate on the bus, it can generate a repeated START condition (Sr) and address another slave without first generating a STOP condition.

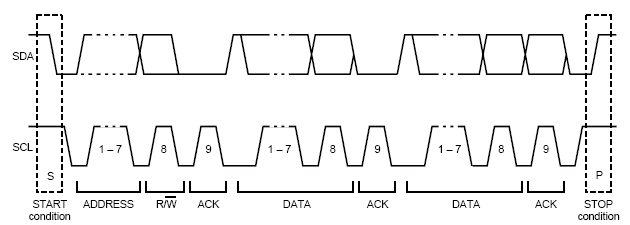


Figure: a complete data transfer

**Master transmit**

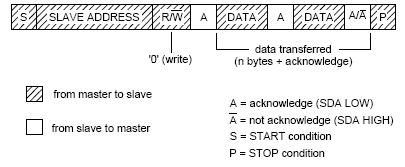


Figure: master transmit

**Master receive**

Master reads slave immediately after first byte. At the moment of the first acknowledge, the master- transmitter becomes a master- receiver and the slave-receiver becomes a slave-transmitter. This first acknowledge is still generated by the slave. The STOP condition is generated by the master, which has previously sent a not-acknowledge (A).

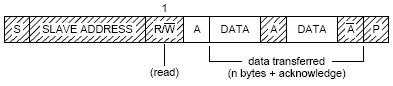


Figure: master receive

**Combined format**

During a change of direction within a transfer, the START condition and the slave address are both repeated, but with the R/W bit reversed. If a master receiver sends a repeated START condition, it has previously sent a not-acknowledge (A)

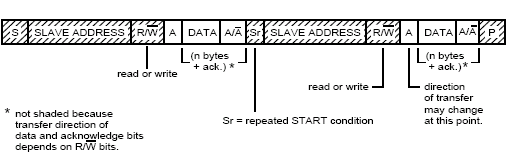


Figure: combined format

### 10-bit Addressing

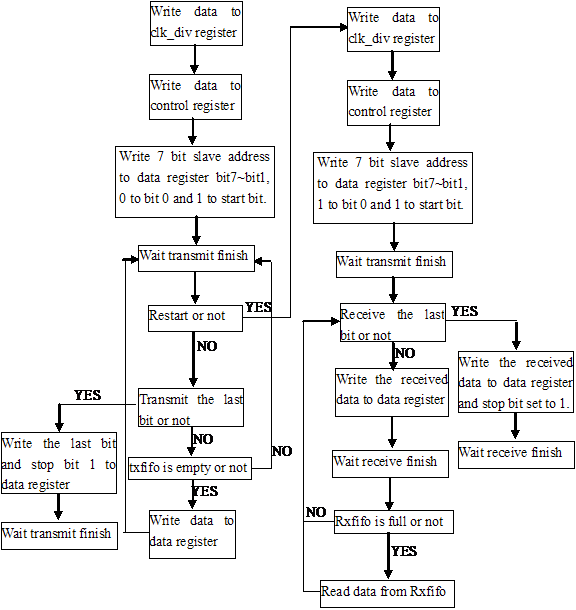
This section describes 10-bit addressing.10-bit addressing is compatible with, and can be combined with, 7-bit addressing. Using 10 bits for addressing exploits the reserved combination 1111XXX. The first seven bits of the first byte following a START (S) or repeated START (Sr) condition. The 10-bit addressing does not affect the existing7-bit addressing. Devices with 7-bit and 10-bit addresses can be connected to the same I2C-bus, and both 7-bit and10-bit addressing can be used in F/S-mode systems. The 10-bit slave address is formed from the first two bytes following a START condition (S) or a repeated START condition (Sr).

The first seven bits of the first byte are the combination11110XX of which the last two bits (XX) are the two most-significant bits (MSBs) of the 10-bit address; the eighth bit of the first byte is the R/W bit that determines the direction of the message. A ‘zero’ in the least significant position of the first byte means that the master will write information to a selected slave. A ‘one’ in this position means that the master will read information from the slave. If the R/W bit is ‘zero’, then the second byte contains the remaining 8 bits (XXXXXXXX) of the 10-bit address. If the R/W bit is ‘one’, then the next byte contains data transmitted from a slave to a master.



Figure: 10-bit address

### Operation description



Master transmits and receives combination

## Register

### Register summary table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Offset | Type | Default | Width | Description |
| I2CDR | 0x00 | RW | 0x0000 | 10 | I2C master data register |
| I2CSR | 0x04 | RO | 0x2ac0 | 14 | I2C status register |
| I2CCTR | 0x08 | RW | 0x00 | 11 | I2C control register |
| I2CCDR | 0x0c | RW | 0x1f4 | 9 | I2C clock divided register |
| I2CADR | 0x10 | RW | 0x6e | 10 | I2C address register |
| I2CRLR | 0x14 | RO | 0x00 | 3 | I2C RX level register |
| I2CTLR | 0x18 | RO | 0x08 | 3 | I2C TX level register |
| I2CRBCR | 0x1c | RO | 0x00 | 16 | I2C RX byte counter register |
| I2CTBCR | 0x20 | RO | 0x00 | 16 | I2C TX byte counter register |
| I2CSDR | 0x24 | RW | 0x00 | 8 | I2C slave data register |

### Register description

**IIC\_DATA REGISTER (I2CDR) Offset: 00’H**

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Description** |
| 31:10 | REV | Reserved |
| 9 | STOP | When this bit is set, IIC transition will stop |
| 8 | START | When this bit is set, IIC transition will start |
| 7:0 | MS\_DATA | When IIC is in master mode, data to be transition |

**IIC\_STATUS REGISTER (I2CSR) offset:04’H**

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Description** |
| 31:14 | REV | Reserved |
| 13 | SLV\_TX\_FFE | Slave TX FIFO EMPTY  In slave mode, if TX FIFO is empty after date is transmitted, this bit will be set  1 = Slave TX FIFO is empty  0= Not empty |
| 12 | SLV\_TX\_FFF | Slave TX FIFO FULL  In slave mode, if TX FIFO is full, this bit will be set  1= Slave TX FIFO is full  0= Not full |
| 11 | MS\_TX\_FFE | Master TX FIFO EMPTY  In master mode, if TX FIFO is empty after date is transmitted, this bit will be set  1= Master TX FIFO is empty  0= Not full |
| 10 | MS\_TX\_FFF | Master TX FIFO FULL  In master mode, if TX FIFO is full, this bit will be set  1= Master TX FIFO is full  0= Not full |
| 9 | MS\_RX\_FFE | MASTER RX FIFO EMPTY  In master mode, if RX FIFO is empty, this bit will be set  1= Master RX FIFO is empty  0= Not empty |
| 8 | MS\_RX\_FFF | MASTER RX FIFO FULL  In master mode, if RX FIFO is full, this bit will be set  1= Master RX FIFO is full  0= Not full |
| 7 | SDA\_ST | Current voltage state of IIC SDA line  1= High voltage  0= Low voltage |
| 6 | SCL ST | Current voltage state of IIC SCL line  1= High voltage  0= Low voltage |
| 5 | BUS\_ACT | IIC bus is active  When IIC detect start condition on IIC bus, this bit will be set. when IIC detect stop condition in I2C bus, this bit will be clear. |
| 4 | SLV\_DATA\_REQ | Data request for slave mode  IIC is in slave mode, when master read data from slave TX FIFO, but slave TX FIFO is empty, this bit will be set.  Writing data to slave TX FIFO will clear this bit. |
| 3 | MS\_DATA\_REQ | Data request for master mode  IIC is in master mode, if TX FIFO is empty, this bit will be set.  Writing data to TX FIFO will clear this bit. |
| 2 | NO\_ACK | No acknowledge interrupt  When IIC finished transiting 8 bit data or 7 bit address and r/w bit, it detect that SDA line is high, this bit will be set.  Writing data to TX FIFO will clear this bit. |
| 1 | ARB\_FAIL | Arbitration failure interrupt.  When IIC receive the last byte of data, it want to push high to SDA line, but it detect that SDA line is low, this bit will be set.  When IIC detect that the value in SDA is not equal to the value that I2C want to push, this bit will be set.  Writing 1 will clear this bit. |
| 0 | TRANS\_DONE | Transaction done interrupt  1= IIC transition is finished.  Writing 1 will clear this bit. |

**IIC\_CONTROL REGISTER (I2CCTR) offset:08’H**

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Description** |
| 31:11 | REV | Reserved |
| 10 | SLV\_TX\_FFNF\_IE | SLAVE TX FIFO NOT FULL IE  1= Enable slave TX FIFO not full interrupt  0= Disable |
| 9 | ADD\_TYP\_SEL | Address bit type selection  1= seven bit  0= ten bit |
| 8 | SW\_RST | Software reset  1= reset IIC block.  0= not reset |
| 7 | MS\_TX\_FFNF\_IE | MASTER TX FIFO NOT FULL IE  1= Enable master TX FIFO not full interrupt  0= Disable |
| 6 | MS\_RX\_FFNE\_IE | MASTER RX FIFO NOT EMPTY IE  1= Enable master RX FIFO not empty interrupt  0= Disable |
| 5 | MS\_RX\_FFF\_IE | MASTER RX FIFO FULL IE  1= Enable master RX FIFO full interrupt  0= Disable |
| 4 | SLV\_DATA\_REQ\_IE | SLAVE DATA REQ IE  1= Enable slave data request interrupt  0= Disable |
| 3 | MS\_DATA\_REQ\_IE | MASTER DATA REQ IE  1= Enable master data request interrupt  0= Disable |
| 2 | NO\_ACK\_IE | NO ACK IE  1= Enable no acknowledge interrupt  0= Disable |
| 1 | ARB\_FAIL\_IE | If there are multi master existing on the bus, there will be an arbitration failure  1= Enable arbitration failure interrupt  0= Disable |
| 0 | TRANS\_DONE\_IE | TRANS DONE IE  1= Enable transaction done interrupt  0= Disable |

**IIC\_CLKDIVREGISTER** **(I2CCDR) offset:0C’H**

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Description** |
| 31:9 | REV | Reserved |
| 8:0 | CLK\_DIV | Clock divide value, IIC bus frequency = pclk/(3 x clk\_div + 9).  Where,  clk\_div is value of bit[8:0]  pclk is cpu running clock |

**IIC\_ADDRESS REGISTER (I2CADR) offset:10’H**

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Description** |
| 31:10 | REV | Reserved |
| 9:0 | SLV\_ADDR | Slave device address, 10 bit or 7 bit. When use 7 bit address, the first 3 bit of this register should be ignored. |

**IIC\_RXLEVEL REGISTER (I2CRLR)offset:14’H**

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Description** |
| 31:3 | REV | Reserved |
| 2:0 | RX\_LVL | RX FIFO current level |

**IIC\_TXLEVEL REGISTER (I2CTLR) offset:18’H**

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Description** |
| 31:3 | REV | Reserved |
| 2:0 | TX\_LVL | TX FIFO current level |

**IIC\_RXBYTECNT REGISTER (I2CRBCR) offset:1C’H**

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Description** |
| 31:16 | REV | Reserved |
| 15:0 | RX\_BYTE\_CNT | Number of received bytes, counter self-add by 1 after one byte is received |

**IIC\_TXBTYECNT REGISTER (I2CTBCR) offset:20’H**

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Description** |
| 31:16 | REV | Reserved |
| 15:0 | TX\_BYTE\_CNT | Number of transmitted bytes, counter self-add by 1 after one byte is transmitted. |

**IIC\_SLAVEDATA REGISTER (I2CSDR) offset:24’H**

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Description** |
| 31:8 | REV | Reserved |
| 7:0 | SLV\_DATA | When IIC is in slave mode, write data to this register to transmit |

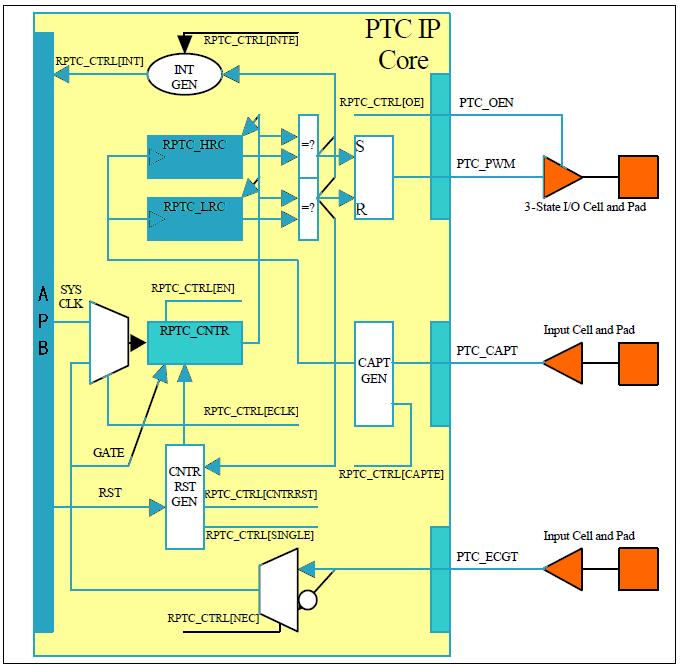
# PWM

The PWM module implements an up-and-down counter with six PWM channels that are assigned to Pins.

## Feature list

* Up to six PWM channels with individual polarity and duty cycle values
* Programmable PWM mode
* 32-bit counter facility
* Single-run or continues run of PWM counter
* Up to (1/48M) high resolution

## Block diagram



When operating in PWM mode, the core generates binary signal with user programmable low and high time.

In PWM modes, PWM\_CNT can run for a single cycle and it can automatically restart after each complete cycle. Cycle completes after reaching value in the PWM\_TRC register. These two modes are called single-run and continues run.

## Register

### Register summary table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Offset** | **Type** | **Default** | **Width** | **Description** |
| PWMn\_CNTR | 0x00+0x10\*n | RW | 0x0 | 32 | PWMn current counter |
| PWMn\_HRC | 0x04+0x10\*n | RW | 0x0 | 32 | PWMn High reference register |
| PWMn\_TRC | 0x08+0x10\*n | RW | 0x0 | 32 | PWMn Total reference register |
| PWMn\_CTRL | 0x0c+0x10\*n | RW | 0x0 | 9 | PWMn control register |

Note: n= 0,1,2,3,4,5

### Register description

**PWM0\_CNTR Offset: 00’H**

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Description** |
| 31:0 | CNT | PWM counter value  **PWM\_CNTR** register is the actual counter register. It is incremented at every counter clock cycle.  In order to count, **PWM\_CNTR** must first be enabled with **PWM\_CTRL[EN]**.  **PWM\_CNTR** can be reset by **PWM\_CTRL[RST]**.  **PWM\_CNTR** can operate in either single-run mode or free-run mode. Mode is selected by **PWM\_CTRL[SINGLE]**. |

**PWM0\_HRC Offset: 04’H**

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Description** |
| 31:0 | HRC | High Reference register  it is used to assert high PWM output  The **PWM\_HRC** should have lower value than **PWM\_TRC**. This is because PWM output goes first high and later low. |

**PWM0\_TRC Offset: 08’H**

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Description** |
| 31:0 | TRC | Total Reference register  it is used to assert low PWM output  The **PWM\_TRC** should have higher value than **PWM\_HRC**. This is because PWM output goes first high and later low. |

**PWM0\_CTRL Offset: 0C’H**

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Description** |
| 31:9 | - | Reserved |
| 8 | SW\_LOAD | 1= HRC/TRC value is updated immediately  0= HRC/TRC value will be updated when **PWM\_CNTR** reach |
| 7 | RST | 1= **PWM\_CNTR** is reset.  0= **PWM\_CNTR** is normal |
| 6:5 | - | Reserved |
| 4 | SINGLE | 1= **PWM\_CNTR** is not incremented after it reaches **PWM\_TRC** value.  0= **PWM\_CNTR** is restarted after it reaches value in the **PWM\_TRC** register. |
| 3 | OE | 1= Enable PWM output.  0= Disable PWM output. |
| 2 | LEVEL | 1= PWM output as reversed value  0= PWM output normal |
| 1 | HTRC\_SEL | 1= Enable bit8 defined function.  0= HRC/TRC value will be updated when **PWM\_CNTR** reach |
| 0 | EN | 1= **PWM\_CNTR** start run.  0= **PWM\_CNTR** stop run and reset |

**Note: PWM1 – PWM5 registers is same as PWM0.**

## PWM mode and operation

### PWM mode

To operate in PWM mode, **PWM\_HRC** and **PWM\_TRC** should be set with the value of high time and total period time of the PWM.

**PWM\_HRC** is number of clock cycles after reset of the **PTC\_CNTR** when PWM output should go high.

**PWM\_TRC** is number of total clock cycles during one pwm period.

PWM\_CNTR can be reset by **PWM\_CTRL[RST]** or periodically when **PWM\_CTRL[SINGLE]** bit is cleared.

To enable PWM output driver, **PWM\_CTRL[OE]** should be set.

To enable continues operation, **PWM\_CTRL[SINGLE]** should be cleared and **PWM\_CTRL[EN]** should be set.

# SARADC

The SARADC is a differential successive approximation register analog-to-digital converter. It supports four external analog input channels. ADC data FIFO depth is 32 bytes

## Feature list

The following lists the main features of the SARADC:

* 10-bit dynamic ADC with 1MHz sample rate.
* Battery monitoring function
* Eight channels for single-ended inputs
* Support 32 bytes FIFO depth
* 4-channels analog input
* Support fixed-channel mode and loop channel-scan mode

There two sample modes of ADC block, one is fixed mode, only one sample channel is chosen to sample analog value, and the sampled value will be stored to FIFO one by one.

The other is loop mode, several sample channel will be selected, in this mode, FIFO is not used, sample data can be accessed by reading **Dn\_REG@0x04 + 0x04 \* n(n=0,1,…3)**.

## Register

**CTRL\_REG@0x00**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 31:17 | REV | - | 00 | Reserved |
| 16 | INT\_CLR | R/W | 0 | Write 1 to clear error interrupt status  Set bit0 to 0,also clear err status |
| 15:12 | REV | - | 00 | Reserved |
| 11:8 | CHNL\_EN | R/W | 0 | Channel enable bits  1= Enable ADC sample channel number  0= Disable ADC sample channel number  Bit[8] --- channel[0]  Bit[9] --- channel[1]  Bit[10] --- channel[2]  Bit[11] --- channel[3] |
| 7 | REV | - | 0 | Reserved |
| 6:4 | ADC\_SEL | R/W | 0 | Channel number in fixed mode  3’b000 --- channel[0]  3’b001 --- channel[1]  … …  3’b111 --- channel[7] |
| 3 | DATA\_VALID | R/W | 0 | 0 = data invalid  1 = data valid |
| 2 | ADC\_MODE | R/W | 0 | work mode  0= channel loop mode  1= channel fixed mode |
| 1 | FIFO\_EN | R/W | 0 | 1= Enable FIFO mode  0= Disable FIFO mode |
| 0 | ADC\_EN | R/W | 0 | 1= Enable SARADC controller block  0= Disable SARADC controller block |

**Dn\_REG@0x04 + 0x04 \* n**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 31:10 | REV | - | 00 | Reserved |
| 9:0 | CHn\_DATA | R | 00 | Channel data |

**Note: n=0,1,2,3**

**DATA\_REG@0x24**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 31:10 | RSV | - | 00 | Reserved |
| 9:0 | ADC\_DATA | R | 00 | FIFO read data/ADC channel data access |

**INT\_REG@0x28**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 31:10 | REV | - | 00 | Reserved |
| 12 | ADC\_ERR | R | 00 | ADC error status |
| 11 | FF\_HE | R | 00 | FIFO half empty status |
| 10 | FF\_E | R | 00 | FIFO empty status |
| 9 | FF\_HF | R | 00 | FIFO half full status |
| 8 | FF\_F | R | 00 | FIFO full status |
| 7:5 | REV | - | 00 | Reserved |
| 4 | ADC\_ERR\_IE | R/W | 00 | 1= Enable ADC error interrupt  0= Disable |
| 3 | FF\_HE\_IE | R/W | 00 | 1= Enable half empty interrupt  0= Disable |
| 2 | FF\_E\_IE | R/W | 00 | 1= Enable FIFO empty interrupt  0= Disable |
| 1 | FF\_HF\_IE | R/W | 00 | 1= Enable FIFO half full interrupt  0= Disable |
| 0 | FF\_F\_IE | R/W | 00 | 1= Enable FIFO full interrupt  0= Disable |

**Example for ADC Program Flow**

1. Program adc mode, fifo\_en, chnl\_en to **CTRL\_REG@0x00**.

2. Write 1 to CTRL\_REG bit[0] to start ADC

3. Keep reading FIFO EMPTY @ **INT\_REG@0x28** util it is 0,which means ADC value is ready to read.

4. Read **Dn\_REG** to get ADC value

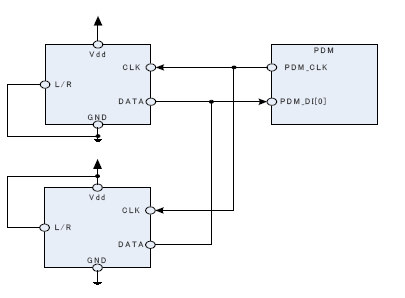
# PDM

## Overview

The pulse density modulation (PDM) module enables input of pulse density modulated signals from external audio frontends, for example, digital microphones. The PDM module generates the PDM clock and supports single-channel or dual-channel (Left and Right) data input.

## Feature list

* 2 PDM microphones configured as a Left/Right pair using the same data input
* Configurable output sample rate (16 kHz or 8kHz), 16-bit samples
* Flexible gain control with user-defined step
* Independent interrupt for data RX
* Support 1M/2M bus speed
* Support 64 16-bit FIFO depth



## Module operation

By default, bits from the left PDM microphone are sampled on PDM\_CLK falling edge, bits for the right are sampled on the rising edge of PDM\_CLK, resulting in two bitstreams. Each bitstream is fed into a digital filter which converts the PDM stream into 16-bit PCM samples, and filters and down-samples them to reach the appropriate sample rate.

The LEFTRISING field in the CTRL register allows swapping Left and Right, so that Left will be sampled on rising edge, and Right on falling.

Depending on the mode chosen in the MONO field in the CTRL register, memory either contains alternating left and right 16-bit samples (Stereo), or only left 16-bit samples (Mono).



## Register

**PDM\_CONTROL Register, Offset: 00’H**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 31:6 | REV | R/W | 00 | Reserved |
| 5 | ZERO\_DEC | R/W | 1 | 1= Enable zero detect when adjust volume  0= Disable |
| 4 | HPFEN | R/W | 0 | 1= Enable High Pass filter  0= Disable |
| 3 | SRMODE | R/W | 1 | 1= 8K sample rate  0= 16K sample rate |
| 2 | LEFTRISING | R/W | 1 | 1= Left channel sampled on PDM\_CLK rising edge  0= Left channel sampled on PDM\_CLK falling edge |
| 1 | MONO | R/W | 1 | MONO mode select, when mono selected, left channel is used  1= MONO  0= Stereo |
| 0 | PDMEN | R/W | 0 | 1= Enable PDM  0= Disable |

**PDM\_VOLGAIN Register, Offset: 04’H**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 31:28 | REV | - | 0 | Reserved |
| 27:16 | RIGHT\_GAIN | R/W | 0x172 | Gain level for right channel |
| 15:12 | REV | - | 0 | Reserved |
| 11:0 | LEFT\_GAIN | R/W | 0x172 | Gain level for left channel |

**PDM\_VOLSTEP Register, Offset: 08’H**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 31:17 | REV | - | 0 | Reserved |
| 16 | VOL\_DIRECT | R/W | 0 | Configure volume directly, not step by step,  usually be used before PDM enabled |
| 15:0 | REV | - | 0 | Reserved |

**PDM\_FFRXDATA Register, Offset: 0C’H**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 31:16 | RXDAT\_RD\_RIGHT | R | 0 | Right channel data |
| 15:0 | RXDAT\_RD\_LEFT | R | 0 | Left channel data |

**PDM\_FFCLR Register, Offset: 10’H**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 31:2 | REV | - | 0 | Reserved |
| 1 | RXFIFO\_RD\_CLR | R/W | 0 | 1= Clear RX FIFO read point  0= Release clear operation |
| 0 | RXFIFO\_WR\_CLR | R/W | 0 | 1= Clear RX FIFO write point  0= Release clear operation |

**PDM\_FFSTATUS Register, Offset: 14’H**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 31:3 | REV | - | 0 | Reserved |
| 2 | RX\_FF\_E | R/W | 0 | 1= PDM RX FIFO is empty  0= Not empty |
| 1 | RX\_FF\_HE | R/W | 0 | 1= PDM RX FIFO is half full  0= Not half full |
| 0 | RX\_FF\_F | R/W | 0 | 1= PDM RX FIFO is full  0= Not full |

**PDM\_INTEN Register, Offset: 18’H**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 31:3 | REV | - | 0 | Reserved |
| 2 | RX\_FF\_E\_IE | R/W | 0 | 1= Enable PDM RX FIFO empty interrupt  0= Disable |
| 1 | RX\_FF\_HE\_IE | R/W | 0 | 1= Enable PDM RX FIFO half-full interrupt  0= Disable |
| 0 | RX\_FF\_F\_IE | R/W | 0 | 1= Enable PDM RX FIFO full interrupt  0= Disable |

**PDM\_AFLR Register, Offset: 1C’H**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 31:6 | REV | - | 0 | Reserved |
| 5:0 | RX\_FF\_LVL | R/W | 0x20 | RX FIFO full level |

# I2S

## Overview

The I2S (Inter-IC Sound) module, support only one channel I2s interface.

## Feature list

* Support AMBA at full-duplex data TX and RX
* Support I2S master mode
* Support I2S left justified and right justified format
* Support max 24Mhz bus speed, and 16 kHz/8kHz, 16-bit sample rates
* 64 16-bit width FIFO depth
* Configurable system clock to audio interface clock division
* Interrupt for TX and RX data transfer

## Operations

### Mode

The I2S protocol specification defines one operation mode, Master.

The I2S mode decides which side (Master or Slave) shall provide the clock signals FRM and BCLK, and these signals are always supplied by the Master to the Slave.

### BCLK

The BCLK, often referred to as the serial bit clock, pulses once for each data bit being transferred on the serial data lines SDIN and SDOUT.

PCLK is the system clock(CPU running clock), and the BCLK\_DIV is set in the **I2S\_BCLK\_DIV** register.

### Frame Clock

The Frame Clock often referred to as "word clock", "sample clock" or "word select" in I2S context, is the clock defining the frames in the serial bit streams sent and received on SDOUT and SDIN, respectively.

In I2S mode, each frame contains one left and right sample pair.

The left sample being transferred during the low or high (configured by **I2S\_CONTROL[FRMINV]** register) half period of FRAME followed by the right sample being transferred during the high or low period of FRAME.

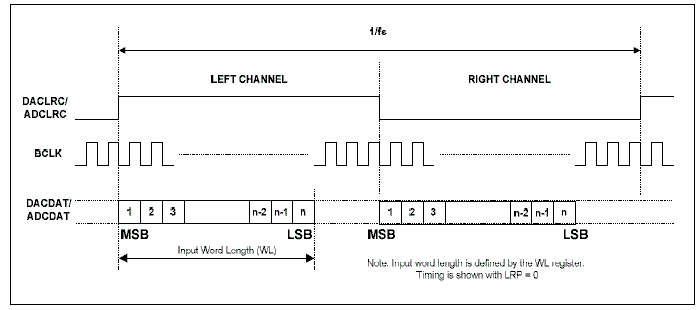
FRM\_DIV is set in the **I2S\_FRM\_DIV** register.

Consequently, the Frame frequency is equivalent to the audio sample rate. Frame Clock toggles at the falling or rising edge clock of the BCLK

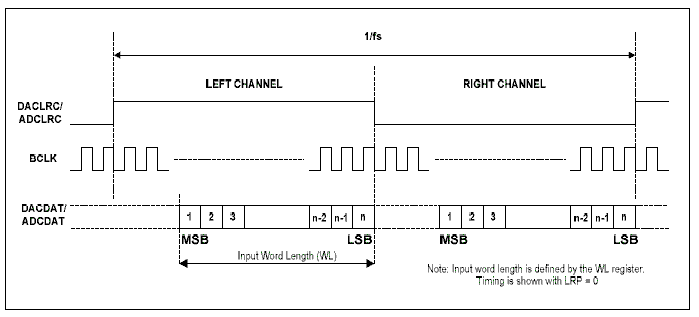
### Audio Interface Timing

The audio interface timing is shown below.

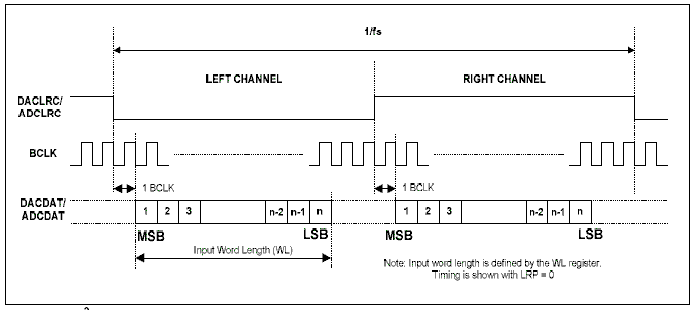
**I2SDLYEN = 0, I2SFMT = 0, I2SFRMINV = 1**



**I2SDLYEN = 0, I2SFMT = 1, I2SFRMINV = 1**



**I2SDLYEN = 1, I2SFMT =0, I2SFRMINV = 0**



## Register

**I2S\_CONTROL Register Offset: 00’H**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 31:10 | REV | R/W | 00 | Reserved |
| 9 | I2SINTEN | R/W | 0 | 1= Enable I2S Interrupt  0= Disable |
| 8 | I2SDMATXEN | R/W | 0 | 1= Enable TX interrupt in DMA mode  0= Disable |
| 7 | I2SDMARXEN | R/W | 0 | 1= Enable RX interrupt in DMA mode  0= Disable |
| 6 | I2SLP | R/W | 0 | Reserved for loop back test enable |
| 5 | I2SMODE | R/W | 0 | Should be set to 1  1= I2S master mode  0= I2S slave mode |
| 4 | I2SBCLKINV | R/W | 0 | BCLK inverse enable  1= Frame Clock toggles at the rising edge  0= Frame Clock toggles at the falling edge |
| 3 | I2SFRMINV | R/W | 0 | Frame clock inverse enable  1= Transfer left sample during the high half period  0= Transfer left sample during the low half period  \*refer to 11.3.4 timing for deep understand |
| 2 | I2SDLYEN | R/W | 0 | Enable delay 1 BCLK transferring frame data,  \*refer to 11.3.4 timing for deep understand |
| 1 | I2SFMT | R/W | 0 | 1= Frame data is Right-alignment  0= Frame data is Left-alignment  \*refer to 11.3.4 timing for deep understand |
| 0 | I2SEN | R/W | 0 | 1= Enable I2S module  0= Disable |

**I2S\_BCLK\_DIV REGISTER Offset: 04’H**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 31:16 | REV | R/W | 00 | Reserved |
| 15:0 | I2SBCLKDIV | R/W | 0h | I2S BCLK divider. |

**I2S\_FRM\_DIV REGISTER Offset: 08’H**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **names** | **Access** | **Default** | **Description** |
| 31:16 | REV | R/W | 00h | Reserved |
| 15:0 | I2SFRMDIV | R/W | 0h | I2S frame divider for frame generation clock |

**I2S\_DATA REGISTER Offset: 0C’H**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **names** | **Access** | **Default** | **Description** |
| 31:0 | I2SDATA | R/W | 0h | When write this register, data is written to transmit FIFO.  When read this register, receive FIFO data. |

**I2S\_STATUS REGISTER Offset: 10’H**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **names** | **Access** | **Default** | **Description** |
| 31:6 | REV | R/W | 00h | Reserved |
| 5 | TXFFEMPTY | R | 0h | 1= Transmit FIFO is empty  0= Transmit FIFO is not empty |
| 4 | TXFFHFULL | R | 0h | 1 = Transmit FIFO is half full  0 = Transmit FIFO is not half full |
| 3 | TXFFFULL | R | 0h | 1 = Transmit FIFO is full  0 = Transmit FIFO is not full |
| 2 | RXFFEMPTY | R | 0h | 1 = Receive FIFO is empty  0 = Receive FIFO is not empty |
| 1 | RXFFHFULL | R | 0h | 1 = Receive FIFO is half full  0 = Receive FIFO is not half full |
| 0 | RXFFFULL | R | 0h | 1 = Receive FIFO is full  0 = Receive FIFO is not full |

**I2S\_INTERUPT\_ENABLE REGISTER Offset: 14’H**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **names** | **Access** | **Default** | **Description** |
| 31:6 | REV | R/W | 00h | Reserved |
| 5 | TX\_FF\_E\_IE | R/W | 0h | 1= Enable transmit FIFO empty interrupt  0= Disable transmit FIFO empty interrupt |
| 4 | TX\_FF\_HF\_IE | R/W | 0h | 1= Enable transmit FIFO half full interrupt  0= Disable transmit FIFO half full interrupt |
| 3 | TX\_FF\_F\_IE | R/W | 0h | 1= Enable transmit FIFO full interrupt  0= Disable transmit FIFO full interrupt |
| 2 | RX\_FF\_E\_IE | R/W | 0h | 1= Enable receive FIFO empty interrupt  0= Disable receive FIFO empty interrupt |
| 1 | RX\_FF\_HF\_IE | R/W | 0h | 1= Enable receive FIFO half full interrupt  0= Disable receive FIFO half full interrupt |
| 0 | RX\_FF\_F\_IE | R/W | 0h | 1= Enable receive FIFO full interrupt  0= Disable receive FIFO full interrupt |

**Example for I2S Program Flow**

1. WritePCLK/(I2S\_BCLK\*2))-1 to reg **I2S\_BCLK\_DIV** to set **BCLK**.

Where:

PCLK= system\_clock, I2S\_BCLK=12M.

1. Write(I2S\_BCLK/(sample\_rate\*2))-1 to reg **I2S\_FRM\_DIV** to set **Frame Clock**

Where:

I2S\_BCLK=12M, sample\_rate=8K,16K… it is user defined value.

1. Write interrupt enable bits to reg **I2S\_INTERUPT\_ENABLE.**

For data sending, set “bit4|bit5|bit0” to reg **I2S\_INTERUPT\_ENABLE.**

For data receiving, set “bit1|bit0” to reg **I2S\_INTERUPT\_ENABLE.**

For data rx&tx, set “bit1|bit4|bit5|bit0” to reg **I2S\_INTERUPT\_ENABLE.**

1. Write “bit1|bit2|bit7|bit8|bit5” to reg **I2S\_CONTROL,** configure the I2s block
2. Write “bit0|bit1|bit2|bit7|bit8|bit5|bit9” to reg **I2S\_CONTROL,** start the I2s block
3. Write “bit1|bit7|bit8|bit5|bit9” to reg **I2S\_CONTROL,** stop the I2s block
4. In I2S ISR function,
5. Read reg **I2S\_STATUS REGISTER** and reg **I2S\_INTERUPT\_ENABLE**, do “and” operation with these two value go get actual interruption status.
6. If actual interruption status set RXFFHFULLINTEN bit, keep reading reg **I2S\_DATA REGISTER** for 32 times, and sent the read data to encode task.
7. If actual interruption status set TXFFHFULLINTEN bit, keep writing 16-bits wide data to reg **I2S\_DATA REGISTER** for 32 times.

# TIMER

## Overview

There are two separate 16-bit Free Running Decremented Counters with individual interrupt for digital timer function. And each 16-bit Counter is combined with 4 4-bit nibbles.

## Feature list

* Separated 16-bytes timer counter
* Support 2 timer block
* Support single-period and periodic run mode
* Up to (1/48Mhz) clock accuracy

## Timer Block Diagram



## Register

Below registers for timer 1 and timer 2.

n means 1 or 2 to represent timer module number.

**Timer FRCn Load Value Register ,Offset: 0x00 + 0x20 \*n**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Names** | **Access** | **Default** | **Description** |
| 15:0 | LOAD\_VAL | R/W | 0 | Initial value of timer in FRCn(Free-running counter n) |

**Timer FRCn Count Value Register Offset: 0x04 + 0x20 \*n**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Names** | **Access** | **Default** | **Description** |
| 15:0 | CNT\_VAL | R/W | 0xff | Indicates the 16-bit counter real-time value in FRCn |

**Timer FRCn Control Register Offset: 0x08 + 0x20 \*n**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Names** | **Access** | **Default** | Description |
| 15:8 | REV | R | 0 | Reserved |
| 7 | CNT\_EN | R/W | 0 | FRCn Counter Enable:  1= Start counting.  0= Stop counting. |
| 6 | CNT\_MOD | R/W | 0 | FRCn Count mode select:  1= Periodic count  0= Single period count |
| 5:4 | REV | R | 0 | Reserved |
| 3:2 | CNT\_CLK\_SEL | R/W | 00 | FRCn count clock setting:  00= count clock is pclk;  01= count clock is pclk/16  Others: count clock is pclk/256  pclk is cpu running clock |
| 1:0 | REV | R | 0 | Reserved |

**Timer FRCn Interrupt Clear Register Offset: 0x0C + 0x20 \*n**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Names** | **Access** | **Default** | **Description** |
| 31:16 | REV | R | 0 | Reserved |
| 15:0 | INT\_CLR | R/W | 0 | Write 1 to clear FRCn interrupt stauts  The interrupt is cleared (LOW) when the FRCn Interrupt Clear Register is written. |

## Operation Description

* Write to the control Register to select prescale and count mode
* Write to the load Register to set the initial value of counter
* Write to the control Register to set count enable bit to enable count
* Wait for interrupt.
* Write to the interrupt clear register to clear the interrupt

# EFUSE

EFUSE is a technology which allows for the dynamic real-time reprogramming of chips. The primary application of this technology is to provide in-chip performance tuning.

## Register

**CTRL Offset: 00’H**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 31:4 | REV | - | 0 | Reserved |
| 3 | TRIG | R/W | 0 | AVDD voltage for EFUSE active trigger  0= High  1= Low |
| 2 | BEN | R/W | 0 | For read operation, this bit represent EFUSE burn operation is ready, cleared by HW  1= EFUSE burn operation is not done  0= EFUSE burn operation is done  For write operation, this bit represent EFUSE enable burn  1= EFUSE is enable for burn  0= EFUSE can’t be burn |
| 1 | REN | R/W | 0 | For read operation, this bit represent EFUSE data ready  1= EFUSE data is not ready for read, cleared by HW  0= EFUSE data is ready for read  For write operation, this bit represent EFUSE enable data read  1= EFUSE data is enable for read  0= EFUSE data can’t be read |
| 0 | EN | R/W | 0 | EFUSE enable & go done status  For read operation, this bit represent that action(read/burn) is prepared.  1= Prepared  0= Not prepared  For write operation, this bit represent control  1= EFUSE action(read/burn) enable  0= Not enable |

**DATA0 Offset: 04’H**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 31:0 | DATA | R/W | 0 | Data[31:0] bit for EFUSE burn or read |

**DATA1 Offset: 08’H**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 31:0 | DATA | R/W | 0 | Data[63:32] bit for EFUSE burn or read |

**DATA2 Offset: 0C’H**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 31:0 | DATA | R/W | 0 | Data[95:64] bit for EFUSE burn or read |

**TIM\_CFG Offset: 10’H**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 31:14 | REV | - | 0 | Reserved |
| 13:8 | HIGH\_TIME | R/W | 0x28 | EFUSE burn high time clock counter |
| 7:6 | REV | - | 0 | Reserved |
| 5:0 | HOLD\_TIME | R/W | 0x14 | EFUSE AVDD signal setup/hold time clock counter |

**Operating instructions**

**Burn mode:**

* Set the TIM\_CFG register to 0x2814 typically.
* Write the 96bit data to DATA2~DATA0 register.
* Set the CTRL register to 0x05(EFUSE enable and EFUSE burn enable).
* Wait for the BEN bit in the CTRL register being cleared by the HW.
* Wait for the EN bit in the CTRL register setting to 1

**Read mode:**

* Set the CTRL register to 0x03(EFUSE enable and EFUSE read enable).
* Wait for the REN bit in the CTRL register being cleared by the HW.
* Wait for the EN bit in the CTRL register setting to 1
* Set the TIM\_CFG register to 0x2814 typically.
* Get the 96-bit data from DATA2~DATA0 register.

# FRSPIM(freq-chip designed spi master controller)

The SPI master can communicate with multiple SPI slaves using individual chip select signals for each slave.

Listed here are the main features for the SPIM

* Support Modem/RF SPI register access
* Support PMU/CODEC SPI register access
* Burst read and write (up to 4bytes)
* Configurable serial clock frequency

This interface is for pmu reg & codec reg access, user don’t need to know how to configure this interface.

PMU regs is described in chapter 19, and Codec regs is described in chapter 20.

To write pmu regs, by ool\_write(addr, data), ool\_write16(addr,data) and ool\_write32(addr,data).

To read pmu regs, by ool\_read(addr, data), ool\_read16(addr,data) and ool\_read32(addr,data).

To write codec regs, by codec\_write(addr, data).

To read codec regs, by codec \_read(addr, data).

These access function is listed in header file “driver\_frspim.h”.

# PMU (Power management unit)

This block describes the registers about power management unit. The registers include PIN, clock&reset, interrupt, debounce, watchdog, RTC, matrix keyboard, Qdec(quadrature decoder), calibration, PWM, sleep and wake-up setting. The pmu block will be power on during deep sleep, while digital block may be power down during deep sleep.

PMU regs should be access through FRSPIM interface(refer to chapter 17), which is a specially interface only for pmu.

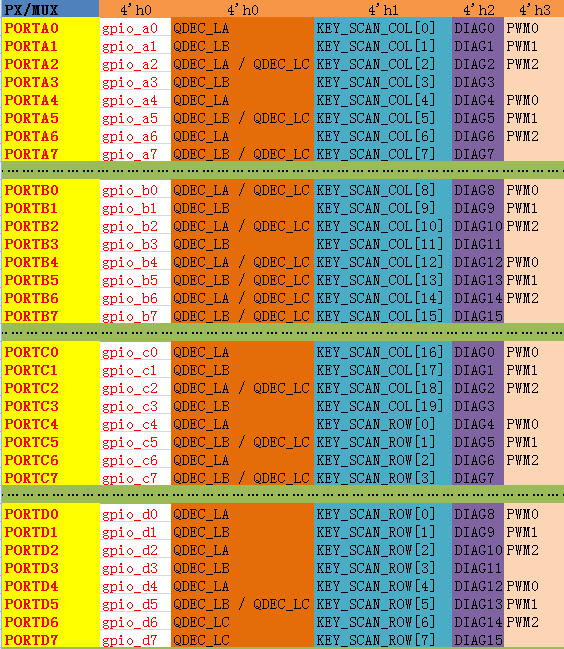
User can access pmu regs by function ool\_write(addr, data)/ool\_read(addr, data), defined in “driver\_frspim.h”

## PIN Configuration

The FR801xH normally have 32 pins. Each pin can be controlled by Digital Logic (not available in deep sleep mode) or PMU (always on module). Following tables show the pmu register details.

The pin IO mux configuration with pmu module please refer to below picture.

User can write **PMU porta mux setting@0xA8, PMU portb mux setting@0xC0, PMU portc mux setting@0xC2, PMU portd mux setting@0xC4** to configure pin to different pmu modules.



**PMU pin selection@0x58**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| pin selection | 31:0 | R/W | 32’h00 | 1: Pin is controlled by digital logic  0: Pin is controlled by PMU  Each bit maps to {PD[7:0], PC[7:0], PB[7:0], PA[7:0]}.  For example bits[1] is for PA1 |

**PMU pin output enable setting@0x60**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Pin output setting | 31:0 | R/W | 32’h00 | 1: Pin is in input mode  0: Pin is in output mode  Each bit maps to {PD[7:0], PC[7:0], PB[7:0], PA[7:0]}  For example bits[1] is for PA1 |

**PMU pin pull-up setting@0x64**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| pin pull-up setting | 31:0 | R/W | 32’h00 | 1: Pin is floating  0: Pin is pulled up  Each bit maps to {PD[7:0], PC[7:0], PB[7:0], PA[7:0]}  For example bits[1] is for PA1 |

**PMU porta mux setting@0xA8**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Porta mux setting | 15:0 | R/W | 16’h00 | 00: GPIO or Qdec  01: matrix keyboard  10: diagnosis port  11: PWM  Each pin takes two bits for mux configuration.  For example bits[1:0] is for PA0 |

**PMU portb mux setting@0xC0**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Portb mux setting | 15:0 | R/W | 16’h00 | 00: GPIO or Qdec  01: matrix keyboard  10: diagnosis port  11: PWM  Each pin takes two bits for mux configuration.  For example bits[1:0] is for PB0 |

**PMU portc mux setting@0xC2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Portc mux setting | 15:0 | R/W | 16’h00 | 00: GPIO or Qdec  01: matrix keyboard  10: diagnosis port  11: PWM  Each pin takes two bits for mux configuration.  For example bits[1:0] is for PC0 |

**PMU portd mux setting@0xC4**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| portd mux setting | 15:0 | R/W | 16’h00 | 00: GPIO or Qdec  01: matrix keyboard  10: diagnosis port  11: PWM  Each pin takes two bits for mux configuration.  For example bits[1:0] is for PD0 |

**PMU diagnosis output selection@0xAA**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Diagnosis selection | 7:0 | R/W | 8’h00 | Used to select which signals will be routed to diagnosis ports, for inner debug use. |

## Clock and Reset

**PMU clock configuration@0x37**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| SAR-ADC clock divisor from 24M OSC | 3:0 | R/W | 4’h00 | Divisor value for ADC clock configuration.  If ADC source clock is 24M OSC, |
| SAR-ADC clock divisor from low power RC | 5:4 | R/W | 2’h00 | Divisor value for ADC clock configuration.  If SAR-ADC source clock is low power RC(62.5K) |
| PMU system clock source | 7:6 | R/W | 2’h00 | Set PMU system clock for all pmu modules  00: low power RC ( 62.5K )  01: low power RC/2  10: external crystal ( 32768 mostly )  11: 32K ( divided from 24M OSC ) |

**PMU clock controller@0x57**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| SAR-ADC OSC clock enable | 0 | R/W | 1’h00 | 1: enable 24M OSC as SAR-ADC clock source  0: disable 24M OSC as SAR-ADC clock source |
| SAR-ADC low power clock enable | 1 | R/W | 1’h00 | 1: enable low power RC as SAR-ADC clock source  0: disable low power RC as SAR-ADC clock source |
| Keyscan clock enable | 2 | R/W | 1’h00 | 1: enable keyscan clock  0: disable keyscan clock |
| PWM clock enable | 3 | R/W | 1’h00 | 1: enable PWM clock  0: disable PWM clock |
| calibration clock enable | 4 | R/W | 1’h00 | 1: enable calibration clock  0: disable calibration clock |
| RTC clock enable | 5 | R/W | 1’h00 | 1: enable RTC clock  0: disable RTC clock |
| BLE sleep timer clock enable | 6 | R/W | 1’h00 | 1: enable BLE sleep timer clock  0: disable BLE sleep timer clock |
| Debounce clock enable | 7 | R/W | 1’h00 | 1: enable debounce clock  0: disable debounce clock |

**PMU debounce clock setting@0x54**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Debounce clock settings | 7:3 | R/W | 5’h00 | Debounce clock = PMU system clock / ( ( N + 1 ) \* 2 ).  Where, N is the value of bits[7:3].  This will generate the clock source for all pmu module debounce.  For example:  QDEC debounce,  Low voltage detecting debounce,  Battery full debounce,  Onkey switch debounce,  Charge detect debounce. |

**PMU reset controller@0x56**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| External pin reset disable | 0 | R/W | 1’h01 | 1: disable external pin reset PMU  0: enable external pin reset PMU |
| WDT reset disable | 1 | R/W | 1’h01 | 1: disable watchdog reset PMU after timeout;  0: enable |
| External pin reset digital core disable | 2 | R/W | 1’h01 | 1: disable external pin only reset digital core  0: enable external pin only reset digital core |
| BT sleep timer reset | 3 | R/W | 1’h00 | 1: release BLE sleep timer module reset  0: reset BLE sleep timer module |
| RTC reset | 4 | R/W | 1’h00 | 1: release RTC module reset  0: reset RTC module |
| Matrix keyboard reset | 5 | R/W | 1’h00 | 1: release Matrix keyboard module reset  0: reset Matrix keyboard module |
| PWM reset | 6 | R/W | 1’h00 | 1: release PWM module reset  0: reset PWM module |
| calibration reset | 7 | R/W | 1’h00 | 1: release calibration module reset  0: reset calibration module |

Note: Keep unused module in reset mode to save power.

## Interrupt

**PMU interrupt enable @0x3A**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Charge full interrupt enable | 0 | R/W | 1’h00 | 1: enable charge full interrupt  0: disable charge full interrupt |
| Ultra-low voltage interrupt enable | 1 | R/W | 1’h00 | 1: enable ultra-low voltage interrupt  0: disable ultra-low voltage interrupt |
| Low voltage interrupt enable | 2 | R/W | 1’h00 | 1: enable low voltage interrupt  0: disable low voltage interrupt |
| Over temperature interrupt enable | 3 | R/W | 1’h00 | 1: enable over temperature interrupt  0: disable over temperature interrupt |
| Charge plug in interrupt enable | 4 | R/W | 1’h00 | 1: enable charge plug in interrupt  0: disable charge plug in interrupt |
| Calibration end interrupt enable | 5 | R/W | 1’h00 | 1: enable calibration end interrupt  0: disable calibration end interrupt |
| Charge plug out interrupt enable | 6 | R/W | 1’h00 | 1: enable charge plug out interrupt  0: disable charge plug out interrupt |
| Onkey power off interrupt enable | 7 | R/W | 1’h00 | 1: enable onkey power off interrupt  0: disable onkey power off interrupt |

**PMU interrupt enable 1@0x7A**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Onkey low level interrupt enable | 0 | R/W | 1’h00 | 1: enable onkey low level trigger interrupt  0: disable onkey low level trigger interrupt |
| Onkey high level interrupt enable | 1 | R/W | 1’h00 | 1: enable onkey high level trigger interrupt  0: disable onkey high level trigger interrupt |
| ADKEY0 pressed interrupt enable | 2 | R/W | 1’h00 | 1: enable ADKEY0 low level trigger interrupt  0: disable ADKEY0 low level trigger interrupt |
| ADKEY0 pressed interrupt enable | 3 | R/W | 1’h00 | 1: enable ADKEY0 high level trigger interrupt  0: disable ADKEY0 high level trigger interrupt |
| ADKEY1 pressed interrupt enable | 4 | R/W | 1’h00 | 1: enable ADKEY1 low level trigger interrupt  0: disable ADKEY1 low level trigger interrupt |
| ADKEY1 pressed interrupt enable | 5 | R/W | 1’h00 | 1: enable ADKEY1 high level trigger interrupt  0: disable ADKEY1 high level trigger interrupt |

**PMU interrupt status@0x70**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Charge full status | 0 | R/W | 1’h00 | 1: Charge full interrupt is pending  0: Charge full interrupt is clear |
| Power off status | 1 | R/W | 1’h00 | 1: power off interrupt is pending  0: power off interrupt is clear  Both ultra-low volage and onkey power off will affect this bit. |
| Low voltage status | 2 | R/W | 1’h00 | 1: Low voltage interrupt is pending  0: Low voltage interrupt is clear |
| Over temperature status | 3 | R/W | 1’h00 | 1: Over temperature interrupt is pending  0: Over temperature interrupt is clear |
| Charge plug in status | 4 | R/W | 1’h00 | 1: Charge plug in interrupt is pending  0: Charge plug in interrupt is clear |
| Calibration end status | 5 | R/W | 1’h00 | 1: Calibration end interrupt is pending  0: Calibration end interrupt is clear |
| Charge plug out status | 6 | R/W | 1’h00 | 1: Charge plug out interrupt is pending  0: Charge plug out interrupt is clear |
| Matrix keyboard status | 7 | R/W | 1’h00 | 1: Matrix keyboard interrupt is pending  0: Matrix keyboard interrupt is clear |
| RTC alarm A status | 8 | R/W | 1’h00 | 1: RTC alarm A interrupt is pending  0: RTC alarm A interrupt is clear |
| RTC alarm B status | 9 | R/W | 1’h00 | 1: RTC alarm B interrupt is pending  0: RTC alarm B interrupt is clear |
| watchdog status | 10 | R/W | 1’h00 | 1: watchdog interrupt is pending  0: watchdog interrupt is clear |
| ADKEY0 status | 11 | R/W | 1’h00 | 1: ADKEY0 interrupt is pending  0: ADKEY0 interrupt is clear |
| ADKEY1 status | 12 | R/W | 1’h00 | 1: ADKEY1 interrupt is pending  0: ADKEY1 interrupt is clear |
| Onkey status | 13 | R/W | 1’h00 | 1: Onkey interrupt is pending  0: Onkey interrupt is clear |
| GPIO status | 14 | R/W | 1’h00 | 1: GPIO exti interrupt is pending  0: GPIO exti interrupt is clear |
| QDEC status | 15 | R/W | 1’h00 | 1: QDEC interrupt is pending  0: QDEC interrupt is clear |

**PMU interrupt clear@0x3B**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Charge full clear | 0 | R/W | 1’h00 | 1: clear charge full interrupt  0: stop clear operation |
| Power off clear | 1 | R/W | 1’h00 | 1: clear power off interrupt  0: stop clear operation |
| Low voltage clear | 2 | R/W | 1’h00 | 1: clear Low voltage interrupt  0: stop clear operation |
| Over temperature clear | 3 | R/W | 1’h00 | 1: clear Over temperature interrupt  0: stop clear operation |
| Charge plug in clear | 4 | R/W | 1’h00 | 1: clear charge plug in interrupt  0: stop clear operation |
| Calibration end clear | 5 | R/W | 1’h00 | 1: clear Calibration end interrupt  0: stop clear operation |
| Charge plug out clear | 6 | R/W | 1’h00 | 1: clear charge plug out interrupt  0: stop clear operation |
| Matrix keyboard clear | 7 | R/W | 1’h00 | 1: clear Matrix keyboard interrupt  0: stop clear operation |
| RTC alarm A clear | 8 | R/W | 1’h00 | 1: clear RTC alarm A interrupt  0: stop clear operation |
| RTC alarm B clear | 9 | R/W | 1’h00 | 1: clear RTC alarm B interrupt  0: stop clear operation |
| watchdog clear | 10 | R/W | 1’h00 | 1: clear watchdog interrupt  0: stop clear operation  Clear watchdog interrupt will reset watchdog counter value. |
| ADKEY0 clear | 11 | R/W | 1’h00 | 1: clear ADKEY0 interrupt  0: stop clear operation |
| ADKEY1 clear | 12 | R/W | 1’h00 | 1: clear ADKEY1 interrupt  0: stop clear operation |
| Onkey clear | 13 | R/W | 1’h00 | 1: clear Onkey interrupt  0: stop clear operation |
| GPIO clear | 14 | R/W | 1’h00 | 1: clear GPIO interrupt  0: stop clear operation |
| QDEC clear | 15 | R/W | 1’h00 | 1: clear QDEC interrupt  0: stop clear operation |

Note: this register cannot reset automatically, so at least 2 PMU system clock cycles should be wait after clear operation.

For example, below code will clear QDEC interrupt:

ool\_write16(PMU\_REG\_ISR\_CLR, (1<<15));

cpu\_delay\_100us(1);

ool\_write16(PMU\_REG\_ISR\_CLR, 0);

## Debounce(Anti-Shake)

**PMU debounce clock configuration @0x54**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Debounce clock settings | 7:3 | R/W | 5’h00 | Debounce clock = PMU system clock / ( ( N + 1 ) \* 2 ).  Where, N is the value of bits[7:3].  This will generate the clock source for all pmu module debounce.  For example:  QDEC debounce,  Low voltage detecting debounce,  Battery full debounce,  Onkey switch debounce,  Charger detect debounce. |

**PMU Onkey&ADkey debounce configuration @0x79**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Onkey debounce settings | 3:0 | R/W | 4’h00 | When onkey is configured as level trigger interrupt, this setting is used for anti-shaking.  Debounce time = (N<<5 +9) \* debounce clock period  Where,  N is the value of bit[3:0].  debounce clock period is decided by **PMU debounce clock setting@0x54** |
| Adkey debounce settings | 7:4 | R/W | 4’h00 | When adkey is configured as interrupt, this setting is used for anti-shaking.  Debounce time = (N<<5 + 9) \* debounce clock period  Where,  N is the value of bit[7:4].  debounce clock period is decided by **PMU debounce clock setting@0x54** |

**PMU QDEC debounce configuration @0x7B**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| QDEC debounce setting | 7:0 | R/W | 8’h00 | QDEC anti-shake values.  Debounce time = ( 2 + N ) \* debounce clock period  Where,  N is the value of bit[7:0].  debounce clock period is decided by **PMU debounce clock setting@0x54** |

**PMU charge full detect debounce configuration @0xCE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Charge full detect debounce setting | 7:0 | R/W | 8’h00 | **charge** full detect anti-shake values.  Debounce time = (N<<5 + 9) \* debounce clock period  Where,  N is the value of bit[7:0].  debounce clock period is decided by **PMU debounce clock setting@0x54** |

**PMU low voltage detect debounce configuration @0xCF**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Low voltage detect debounce setting | 7:0 | R/W | 8’h00 | low voltage detect anti-shake values.  Debounce time = (N<<5 + 9) \* debounce clock period  Where,  N is the value of bit[7:0].  debounce clock period is decided by **PMU debounce clock setting@0x54** |

**PMU debounce selection@0x68**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Charge full detect debounce selection | 6 | R/W | 1’h00 | Should be fixed to 1 |
| Low voltage detect debounce selection | 7 | R/W | 1’h00 | Should be fixed to 1 |

**PMU charge detect debounce configuration @0x6D**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Charge plug in&out detect debounce setting | 5:0 | R/W | 6’h00 | Charge plug in&out detect anti-shake values.  Debounce time = (N<<5 + 9) \* debounce clock period  Where,  N is the value of bit[5:0].  debounce clock period is decided by **PMU debounce clock setting@0x54** |
| Charge detect debounce selection | 6 | R/W | 1’h00 | Should be fixed to 1 |

## Watchdog

**WDT time configuration@0xB0**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| WDT Time | 23:0 | R/W | 23’h00 | Watchdog timeout counter value.  unit : pmu system clock,  refer to bit[7:6] of **PMU clock configuration@0x37** |

**WDT controller@0xB3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| WDT enable | 0 | R/W | 1’h00 | 1: start watchdog counter;  0: stop watchdog counter; |
| WDT IRQ enable | 1 | R/W | 1’h00 | 1: watchdog occur will generate wdt interrupt  0:watchdog occur won’t generate wdt interrupt |

**PMU reset controller@0x56**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| WDT reset disable | 1 | R/W | 1’h00 | 1: watchdog occur will not reset PMU  0: watchdog occur will reset PMU |

Note: Feed watchdog operation can be done by clear watchdog interruption bit.

Refer to bit 10 of **PMU interrupt clear@0x3B**

## RTC

Built-in RTC supports two alarms( alarm A & alarm B ). Each alarm have threshold value and have independent interrupt.

**RTC controller@0x3D**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| RTC enable | 0 | R/W | 1’h00 | 1: start RTC counter  0: stop RTC counter.  RTC counter is a 32-bit value, after be started,it will automatically self-add at each pmu system period, and will wrap to 0 when reach 0xffffffff. |
| RTC counter update enable | 1 | R/W | 1’h00 | 1: update RTC initial counter  0: release update operation  Update rtc counter value steps:   1. Write 32-bit rtc counter value to **RTC update counter\_0 @0x3E** 2. Setting 1 to this bit 3. wait 2 PMU system clock 4. Setting 0 to this bit. |
| RTC alarm A enable | 2 | R/W | 1’h00 | 1: enable alarm A  0: disable alarm A |
| RTC alarm B enable | 3 | R/W | 1’h00 | 1: enable alarm B  0: disable alarm B |
| RTC alarm A interrupt enable | 4 | R/W | 1’h00 | 1: enable alarm A interrupt  0: disable alarm A interrupt |
| RTC alarm B interrupt enable | 5 | R/W | 1’h00 | 1: enable alarm B interrupt  0: disable alarm B interrupt |

**RTC alarmA counter @(0x42,0x43,0x44,0x45)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Alarm A threshold value | 31:0 | R/W | 31’h00 | RTC alarm A threshold counter value.  When rtc counter, which is 32-bit value from **RTC counter value @0x72,** reach to this threshold, RTC will generate a isr if isr is enabled. |

**RTC alarmB counter @(0x46,0x47,0x48,0x49)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Alarm B threshold value | 31:0 | R/W | 31’h00 | RTC alarm B threshold counter value.  When rtc counter which is 32-bit value from **RTC counter value @0x72,** reach to this threshold, RTC will generate a isr if isr is enabled. |

**RTC update counter @(0x3E,0x3F,0x40,0x41)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | Bits | Access | Default | Description |
| RTC counter update | 31:0 | R/W | 31’h00 | Value to be loaded as rtc current counter.  Unit : pmu system period, refer to bit[7:6] of **PMU clock configuration@0x37** |

**RTC counter value @(0x72,0x73,0x74,0x75)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| RTC counter value | 31:0 | R/W | 31’h00 | RTC current counter value.  Unit : pmu system period, refer to bit[7:6] of **PMU clock configuration@0x37** |

**PMU reset controller@0x56**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| RTC reset | 4 | R/W | 1’h00 | 1: release RTC reset  0: reset RTC |

**PMU clock controller@0x57**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| RTC clock setting | 5 | R/W | 1’h00 | 1: enable RTC clock  0: disable RTC clock |

## Matrix Keyboard

Matrix keyboard block supports 8\*20 keyboard matrix scanning.

Row[7:0] is corresponded for PD[7:0] or {PC[7:4], PD[3:0]}, depend on bit[2] of **Keyscan controller@0xC6.**

Col[21:0] is corresponded for {PC[3:0], PB[7:0], PA[7:0]}.

The IO shall be controlled by PMU when it is used for keyscan.

**Keyscan controller@0xC6**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Keyscan enable | 0 | R/W | 4’h00 | 1: enable keyscan module  0: disable keyscan module |
| Keyscan low power mode | 1 | R/W | 4’h00 | 1: enable keyscan low power mode  0: disable keyscan low power mode  Low power mode means: keyscan module don’t run when no key is pressed, and start scanning when some key is pressed. |
| Keyscan row mapping | 2 | R/W | 4’h00 | Select which pins as row[7:4]  1: row[7:4] = PC[7:4]  0: row[7:4] = PD[7:4] |
| Keyscan interrupt enable | 3 | R/W | 4’h00 | 1: enable keyscan interrupt  0: disable keyscan interrupt  Whenever there is key being pressed or released, an interrupt will be generated  Read 160-bit values from **Keyscan status regs @0xCC** to get current key status. |
| Keyscan column enable 2 | 7:4 | R/W | 4’h00 | Each bit represent column[19:16] selection .  1: enable corresponding column[19:16]  0: disable corresponding column[19:16]  Work together with **Keyscan column enable 0@0xC9** & **Keyscan column enable 1@0xCA** |

**Keyscan safety setting@0xC7**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Keyscan debounce time | 3:0 | R/W | 4’h00 | Used for anti-shaking. The controller will generate a new key-changed interrupt after key status has been kept longer than configured time.  Unit: keyscan interval. (Refer to **Keyscan interval setting@0xCB**) |
| Keyscan debounce enable | 4 | R/W | 1’h00 | 1: Enable anti-shaking.  0: disable anti-shaking |
| Keyscan ghost key detect | 5 | R/W | 1’h00 | 1: Enable ghost key detect  0: Disable ghost key detect  When two keys named A and B belong to the same row, then the keys in the same column with A or B means ghost key. After enable this bit, the ghost keys will not be masked to avoid mistakenly identified. |
| Keyscan ghost key mask 0 | 6 | R/W | 1’h00 | Keep as 1 after bit5 is enabled. |
| Keyscan ghost key mask 1 | 7 | R/W | 1’h00 | Keep as 1 after bit5 is enabled. |

**Keyscan row enable@0xC8**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Bits | Access | Default | Description |
| Keyscan row enable | 7:0 | R/W | 8’h00 | Each bit represent row [7:0] selection  1: enable corresponding row[7:0]  0: disable corresponding row[7:0] |

**Keyscan column enable 0@0xC9**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Keyscan column enable 0 | 7:0 | R/W | 8’h00 | Each bit represent column [7:0] selection  1: enable corresponding column[7:0]  0: disable corresponding column[7:0] |

**Keyscan column enable 1@0xCA**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Keyscan column enable 1 | 7:0 | R/W | 8’h00 | Each bit represent column [15:8] selection  1: enable corresponding column[15:8]  0: disable corresponding column[15:8] |

**Keyscan interval setting@0xCB**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Keyscan interval | 7:0 | R/W | 8’h00 | Interval between two continuous scanning, unit is RC clock/4. RC clock is 62.5KHz |

**Keyscan status regs@0xCC**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Keys status | 159:0 | R/W | 159’h00 | Each bit represent key [159:0] press status  1: key is pressed  0: keys is released |

**PMU reset controller@0x56**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Matrix keyboard reset | 5 | R/W | 1’h00 | 1: release Matrix keyboard reset  0: reset Matrix keyboard |

**PMU clock controller@0x57**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Keyscan clock setting | 2 | R/W | 1’h00 | 1: enable keyscan clock  0: disable keyscan clock |

## Quadrature decoder

Quadrature decoder module can detect two direction rotation counter, when this counter reaches threshold value, a qdec pmu interruption will be generated to inform user.

The pins which are used as quadrature decoder function shall be controlled by PMU and configured as input.

Quadrature decoder module need 3 pins, which are assigned as LA, LB,LC. And pin mux selection please refer to pmu reg **PMU QDEC pin mux@0xB8** and **PMU QDEC LC pin mux@0xB9**.

**PMU QDEC pin mux@0xB8**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| QDEC LA pin mux | 3:0 | W | 4’h00 | Assign which pin as LA input.  0000: reserved 0001: PA0  0010: PA2 0011: PA4  0100: PA6 0101: PB0  0110: PB2 0111: PB4  1000: PB6 1001: PC0  1010: PC2 1011: PC4  1100: PC6 1101: PD0  1110: PD2 1111: PD4 |
| QDEC LB pin mux | 7:4 | W | 4’h00 | Assign which pin as LB input.  0000: reserved 0001: PA1  0010: PA3 0011: PA5  0100: PA7 0101: PB1  0110: PB3 0111: PB5  1000: PB7 1001: PC1  1010: PC3 1011: PC5  1100: PC7 1101: PD1  1110: PD3 1111: PD5 |

**PMU QDEC LC pin mux@0xB9**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| QDEC LC pin mux | 3:0 | W | 4’h00 | Assign which pin as LC input.  0000: reserved 0001: PA2  0010: PA5 0011: PA7  0100: PB2 0101: PB5  0110: PB7 0111: PC2  1000: PC5 1001: PC7  1010: PB0 1011: PB4  1100: PB6 1101: PD5  1110: PD6 1111: PD7 |

**PMU QDEC direction A counter value@0xB8**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| QDEC direction A counter value | 7:0 | R | 8’h00 | Direction A rotation counter number.  When this number reach threshold(defined in **PMU QDEC multi-step rotation interrupt threshold @0x76**), interrupt will be generated.  In interruption function, user can compare this value with direction B counter to judge real rotation direction. |

**PMU QDEC direction B counter value@0xB9**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| QDEC direction B counter value | 7:0 | R | 8’h00 | Direction B rotation counter number.  When this number reach threshold(defined in **PMU QDEC multi-step rotation interrupt threshold @0x76**), interrupt will be generated.  In interruption function, user can compare this value with direction A counter to judge real rotation direction. |

**PMU QDEC counter confiuration@0x68**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| QDEC counter auto reset enable | 0 | W | 1’h00 | Direction A&B rotation counter number auto reset.  1: direction A and B counter value will be auto reset after read  0: disable auto reset after read operation  Read below pmu regs to get two direction counter numbers.  **PMU QDEC direction A counter value@0xB8**  **PMU QDEC direction B counter value@0xB9** |

**PMU QDEC multi-step rotation interrupt threshold@0x76**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| QDEC multi-step rotation interrupt threshold | 7:0 | R/W | 8’h00 | Multi-step rotation interrupt generation threshold.  This value can be compared with direction A&B rotation counter number, if any rotation number is over the threshold, interruption will be generated. |

**PMU QDEC controller@0x78**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| QDEC first edge detect disable | 0 | R/W | 1’h00 | 1: disable first edge detect  0: enable first edge detect  First edge detect means no matter LA or LB level has changed, corresponding direction counter will be increased. |
| QDEC second edge detect disable | 1 | R/W | 1’h00 | 1: disable second edge detect  0: enable second edge detect  Second edge detect means only both LA and LB level has changed, corresponding direction counter will be increased |
| QDEC single-step interrupt enable | 2 | R/W | 1’h00 | 1: enable QDEC single-step interrupt  0: disable QDEC single-step interrupt  If this bit is set, once any direction A &B  rotation counter number is changed, an interrupt will generated. |
| QDEC multi-step interrupt enable | 3 | R/W | 1’h00 | 1: enable QDEC multi-step interrupt  0: disable QDEC multi-step interrupt  If this bit is set, direction A / B rotation counter number is larger than configured threshold, an interrupt will generated.  Bit[0] of **PMU QDEC counter confiuration@0x68**  should be enable to avoid continuous interrupt generation. |
| QDEC LA debounce enable | 4 | R/W | 1’h00 | 1: enable LA pin debounce function  0: disable LA pin debounce function |
| QDEC LB debounce enable | 5 | R/W | 1’h00 | 1: enable LB pin debounce function  0: disable LB pin debounce function |
| QDEC LC debounce enable | 6 | R/W | 1’h00 | 1: enable LC pin debounce function  0: disable LC pin debounce function |
| QDEC LC reset function enable | 7 | R/W | 1’h00 | 1: enable counter reset function from LC pin  0: disable counter reset function from LC pin  If this bit is 1, press LC pin will reset direction A &B  rotation counter numbers. |

**PMU QDEC debounce setting@0x7B**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| QDEC debounce setting | 7:0 | R/W | 8’h00 | QDEC anti-shake values.  Debounce time = ( 2 + N ) \* debounce clock period  Where,  N is the value of bit[7:0].  debounce clock period is decided by **PMU debounce clock setting@0x54** |

## Calibration

This block is used to calibrate low-power clock source(same as pmu system clock) based on the 24MHz system clock.

**PMU calibration time@0x6E**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Calibration time | 15:0 | R/W | 16’h00 | Cycle numbers should be cost to do calibration.  Uint: pmu system clock period.  Refer to bit[7:6] of **PMU clock configuration @0x37**  More cycles bring more precise result, but take more time to finish calibration. |

**PMU calibration result@0x7C**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Calibration result | 31:0 | R/W | 32’h00 | How many 24M clock cycles have been token during calibration.  Frequency = Calibration result \* 24000000 / Calibration time.  Frequency is the calibrated pmu system clock. |

**PMU calibration controller@0x55**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| Calibration enable | 0 | R/W | 0’h00 | 1: start calibration  0: stop calibration |
| Calibration clock selection | 3:1 | R/W | 3’h00 | Decide which clock will be calibrated. Normally, low power RC will be selected.  000: low power RC  001: 2M divided from 2.4G  010: 31.25K divided from 24M OSC  011: external low power 32768  100: signal from pin LED3  101: signal from pin PD5  110: signal from pin PD6  111: signal from pin PD7 |

**PMU reset controller@0x56**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| calibration reset | 7 | R/W | 1’h00 | 1: release calibration reset  0: reset calibration |

**PMU clock controller@0x57**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| calibration clock setting | 4 | R/W | 1’h00 | 1: enable calibration clock  0: disable calibration clock |

## PWM

PMU supports 3 PWM. It can be configured to following GPIO in PMU IO MUX register.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **PWM0** | PA0 | PA4 | PB0 | PB4 | PC0 | PC4 | PD0 | PD4 |
| **PWM1** | PA1 | PA5 | PB1 | PB5 | PC1 | PC5 | PD1 | PD5 |
| **PWM2** | PA2 | PA6 | PB2 | PB6 | PC2 | PC6 | PD2 | PD6 |

IO MUX setting please refer to **porta mux setting@0xA8, portb mux setting@0xC0, portc mux setting@0xC2**

**, portd mux setting@0xC4.**

PMU pwm should set high level counter, and low level counter, unit is pmu system clock

**PMU PWM0 high level counter@0x90**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| PWM0 high level counter | 16 | R/W | 16’h00 | How many cycles PWM0 will output high level.  Unit : pmu system period, refer to bit[7:6] of **PMU clock configuration@0x37** |

**PMU PWM0 low level counter@0x92**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| PWM0 low level counter | 16 | R/W | 16’h00 | How many cycles PWM0 will output low level.  Unit : pmu system period, refer to bit[7:6] of **PMU clock configuration@0x37** |

**PMU PWM1 high level counter@0x94**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| PWM1 high level counter | 16 | R/W | 16’h00 | How many cycles PWM1 will output high level.  Unit : pmu system period, refer to bit[7:6] of **PMU clock configuration@0x37** |

PMU PWM1 low level counter@0x96

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| PWM1 low level counter | 16 | R/W | 16’h00 | How many cycles PWM1 will output low level.  Unit : pmu system period, refer to bit[7:6] of **PMU clock configuration@0x37** |

**PMU PWM2 high level counter@0x98**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| PWM2 high level counter | 16 | R/W | 16’h00 | How many cycles PWM2 will output high level.  Unit : pmu system period, refer to bit[7:6] of **PMU clock configuration@0x37** |

**PMU PWM2 low level counter@0x9A**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| PWM2 low level counter | 16 | R/W | 16’h00 | How many cycles PWM2 will output low level.  Unit : pmu system period, refer to bit[7:6] of **PMU clock configuration@0x37** |

**PMU PWM0 controller@0xA4**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| PWM0 start | 0 | R/W | 1’h00 | 1: start PWM0 counter  0: stop PWM0 counter |
| PWM0 output enable | 1 | R/W | 1’h00 | 1: enable signal route to configured pad  0: disable signal route to configured pad |
| PWM0 output single mode | 2 | R/W | 1’h00 | 1: only output one period after enabled  0: output configured PWM wave continuously |
| PWM0 counter reset | 3 | R/W | 1’h00 | 1: reset PWM0 counter  0: stop reset PWM0 counter operation  Two low power RC cycles should be wait between reset and clear. |
| PWM0 output reverse | 4 | R/W | 1’h00 | 1: output level is reversed.  0: output level is no changed. |

**PMU PWM1 controller@0xA5**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| PWM1 start | 0 | R/W | 1’h00 | 1: start PWM1 counter  0: stop PWM1 counter |
| PWM1 output enable | 1 | R/W | 1’h00 | 1: enable signal route to configured pad  0: disable signal route to configured pad |
| PWM1 output single mode | 2 | R/W | 1’h00 | 1: only output one period after enabled  0: output configured PWM wave continuously |
| PWM1 counter reset | 3 | R/W | 1’h00 | 1: reset PWM1 counter  0: stop reset PWM1 counter operation  Two low power RC cycles should be wait between reset and clear. |
| PWM1 output reverse | 4 | R/W | 1’h00 | 1: output level is reversed.  0: output level is no changed. |

**PMU PWM2 controller@0xA4**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| PWM2 start | 0 | R/W | 1’h00 | 1: start PWM2 counter  0: stop PWM2 counter |
| PWM2 output enable | 1 | R/W | 1’h00 | 1: enable signal route to configured pad  0: disable signal route to configured pad |
| PWM2 output single mode | 2 | R/W | 1’h00 | 1: only output one period after enabled  0: output configured PWM wave continuously |
| PWM2 counter reset | 3 | R/W | 1’h00 | 1: reset PWM2 counter  0: stop reset PWM2 counter operation  Two low power RC cycles should be wait between reset and clear. |
| PWM2 output reverse | 4 | R/W | 1’h00 | 1: output level is reversed.  0: output level is no changed. |

**PMU reset controller@0x56**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| PWM reset | 6 | R/W | 1’h00 | 1: release PWM reset  0: reset PWM |

**PMU clock controller@0x57**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Bits** | **Access** | **Default** | **Description** |
| PWM clock setting | 3 | R/W | 1’h00 | 1: enable PWM clock  0: disable PWM clock |

# CODEC

## Overview

This block consists of 1-ch ADC, 1-ch DAC, microphone amplifier, speaker amplifier and analog mixing and gain functions. It uses advanced multi-bit delta-sigma modulation technique to convert data between digital and analog. The multi-bit delta-sigma modulators make the device with low sensitivity to clock jetter and low out of band noise. Different audio sample rates such as 48kHz, 44.1kHz and 8kHz are generated directly from the master clock without the need for an external PLL.

The Codec register is configured by the FRSPIM. Please refer to **FRSPIM** (chapter 17).

## Feature list

* 1-ch 16bit  ∑ΔADC, 1-ch 16bit ∑ΔDAC,
* Samples rate is up to 48KHz32-Byte depth FIFOs for both Rx/Tx Independently Receiver Clock Input
* Internal microphone bias equal to 0.9\*CODEC power voltage
* Input PGA amplifier -17.25dB - 30dB gain range
* Output earphone PA with 50mW output power

## Register

Reg00@0x00

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **SW** |
| 7 | EN\_DEC | R | R/W | 1’b0 | The clock enable or data valid signal. When “1”, the decimation filter is enabled |
| 6 | EN\_INT | R | R/W | 1’b0 | The enable signal for the interpolating filter and sigma-delta modulator. When “1”, the interpolate filter is enabled. |
| 5 | DECRST | R | R/W | 1’b1 | nrst of dec data path |
| 4 | INTRST | R | R/W | 1’b1 | nrst of int data path |
| 3 | DITHER\_EN | R | R/W | 1’b0 | Digital dsm dither enable;  1=enable |
| 2 | SCRAMBLE\_EN | R | R/W | 1’b1 | Digital dsm out DEM module enable;  1=enable |
| 1 | DSM\_MODE | R | R/W | 1’b0 | Digital dsm module;  1=1+z^(-3)  0=1+z^(-1) |
| 0 | HPF\_EN | R | R/W | 1’b1 | Digital ADC highass filter enable  1= enable |

Reg01@0x01

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 7 | REV | R | R/W | 1’b0 | REV |
| 6:0 | DITHEROW[22:16] | R | R/W | 7’b0000100 | Dither power |

Reg02@0x02

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 7:0 | DITHEROW[15:8] | R | R/W | 8’b1111\_1101 | Dither power |

Reg03@0x03

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 7:0 | DITHEROW[7:0] | R | R/W | 8’hF3 | Dither power |

Reg04@0x04

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 7:5 | SDTG\_R | R | R/W | 3’b000 | Gain control of the right channel side tone. |
| 4:2 | RSV0 | - | - | - | Reserved |
| 1 | INT\_MUTE\_R | R | R/W | 1’b0 | Right interpolate filter mute |
| 0 | RSV1 | - | - | - | Reserved |

Reg05@0x05

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 7:4 | REV |  |  | 4’h0 | Reserved |
| 3:0 | INT\_VOL\_R[11:8] | R | R/W | 4’h1 | The volume control for the right interpolate filter. |

Reg06@0x06

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 7:0 | INT\_VOL\_R[7:0] | R | R/W | 8’hc8 | The volume control for the right interpolate filter. The int\_vol range from [0,4095], the gain is calculated by , eg. default value is about 0.89 |

Reg09@0x09

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| HW | RW |
| 7 | BCLKINV | R | R/W | 1’b0 | 0 = BCLK not inverted |
| 1 = BCLK inverted |
| 6 | RSV | - | - | 1’b0 | Reserved |
| 5 | LRSWAP | R | R/W | 1’b1 | 1 = swap left and right DAC data in audio interface |
| 0 = output left and right data as normal |
| 4 | LRP | R | R/W | 1’b0 | Right, Left and I2S modes – LRCLK |
| 1 = invert LRCLK polarity |
| 0 = normal LRCLK polarity |
| DSP Mode – mode A/B select |
| 1 = MSB is available on 1st BCLK rising edge after LRC rising edge (mode B) |
| 0 = MSB is available on 2nd BCLK rising edge after LRC rising edge (mode A) |
| 3:2 | DCI\_WL | R | R/W | 2’b00 | 11 = 32 bits |
| 10 = 24 bits |
| 01 = 20 bits |
| 00 = 16 bits |
| 1:0 | FMT[1:0] | R | R/W | 2’b10 | 11 = DSP Mode |
| 10 = I2S Format |
| 01 = Left justified |
| 00 = Right justified |

Reg0a@0x0a

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 7:4 | DEC\_SR | R | R/W | 4’b0 | DEC Sample Rate Control  4’b0000: 8K  4’b0001: 12K  4’b0010: 16K  4’b0011: 24K  4’b0100: 48K  4’b1000: 8.0214  4’b1001: 11.0259  4’b1010: 22.0588  4’b1011: 44.1 |
| 3:0 | INT\_SR | R | R/W | 4’b0 | INT Sample Rate Control  the same as above |

Reg0b@0x0b

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 7:3 | TEST\_MOD | R | R/W | 4’b0000 | Change digital in/out for test |
| 2 | TEST\_RES\_MOD | R | R/W | 1’b0 | tst\_resin [23:0] can be used as test control for analog (whenenable , DITHEROW[22:0] has no use) |
| 1:0 | RSV | - | - | 2’b00 | Reserved |

Reg0c@0x0c

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 7 | CODEC\_BIAS\_MODE | R | R/W | 1’b0 | 1: codec bias mode  0: normal mode |
| 6 | LVVDDA\_EN | R | R/W | 1’b0 | 1: Low analog voltage enable  0: Low analog voltage disable |
| 5 | MICBIAS\_CTL | R | R/W | 1’b0 | 1: MIC bias control enable  0: MIC bias control disable |
| 4:2 | BIAS\_IB\_IS[2:0] | R | R/W | 3’b000 | BIAS current control  00: default  01: 20%  10: 40%  11: -20% |
| 1:0 | VMID\_CTL[1:0] | R | R/W | 2’b11 | The ramp up control for VMID  00: slowest  01: Slow  10: Fast  11: Fastest |

Reg0e@0x0e

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 7 | - |  |  | 1’b0 | Reserved |
| 6 | ADC\_ENCODE\_MODE |  |  | 1’b0 | ADC output encoding mode |
| 5:4 | ADC\_DITHER\_AMP |  |  | 2’b11 | The amplitude setting of the dithering feed |
| 3 | ADC\_DEM\_EN |  |  | 1’b1 | ADC DEM enable |
| 2 | ADC\_DITHER\_EN |  |  | 1’b1 | ADC dither enable |
| 1 | ADC\_DITHER\_IN |  |  | 1’b0 | ADC dither input |
| 0 | ADC\_ICTL |  |  | 1’b1 | ADC current control |

Reg0f@0x0f

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 7:6 | MIX\_CAP\_CTL |  |  | 2’b11 | Mixer cap control  00: min cap setting  01: min + 25% setting  10: max - 25% setting  11: max cap setting |
| 5:4 | PA\_IS |  |  | xxxx | PA current setting |
| 3:0 | DAC\_IS |  |  | 4’h0 | DAC current settings |

Reg10@0x10

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 7:6 | REV | R | R/W | 2’b00 | Reserved |
| 5:0 | LPAVOL[5:0] | R | R/W | 6’b111111 | Left channel power amplifier setting. More reference follow tables. |

Reg11@0x11

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 7:6 | REV | R | R/W | 2’b00 | Reserved |
| 5:0 | RPAVOL[5:0] | R | R/W | 6’b111111 | Right channel power amplifier setting. More reference follow tables. |

Reg12@0x12

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 7 | PARES | R | R/W | 1’b0 | Reserved |
| 6 | PGA\_MUTE | R | R/W | 1’b1 | 1: PGA mute enable  0: PAG mute disable |
| 5 | RDAC\_MUTE | R | R/W | 1’b1 | 1: Right DAC mute enable  0: Right DAC mute disable |
| 4 | RSV | - | - | 1’b1 | Reserved |
| 3 | PADD | R | R/W | 1’b1 | 1: Power amplifier power done  0: Power amplifier power on |
| 2 | ADCD | R | R/W | 1’b1 | 1: ADC channel power done  0: ADC channel power on |
| 1 | RDACD | R | R/W | 1’b1 | 1: Right DAC channel power done  0: Right DAC channel power on |
| 0 | RSV | - | - | 1’b1 | Reserved |

Reg13@0x13

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 7:6 | RMIX\_MUTE | R | R/W | 2’b11 | 1: Right mixer mute enable  0: Right mixer mute disable |
| 5:4 | RSV | - | - | 2’b11 | Reserved |
| 3:2 | RPA\_MUTE | R | R/W | 2’b11 | 1: Right PA mute enable  0: Right PA mute disable |
| 1:0 | RSV | - | - | 2’b11 | Reserved |

Reg14@0x14

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 7:6 | RMIXD | R | R/W | 2’b11 | 1: Right mixer power done  0: Right mixer power on |
| 5:4 | RSV | - | - | 2’b11 | Reserved |
| 3:2 | RPAD | R | R/W | 2’b11 | 1: Right power amplifier power done  0: Right power amplifier power on |
| 1:0 | RSV | - | - | 2’b11 | Reserved |

Reg15@0x15

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 7 | Rev |  |  |  | Reserved |
| 6 | DAC\_CLK\_EDGE | R | R/W | 1’b0 | DAC channel clock select  1: posedge  0: negedge |
| 5 | PA\_CZEN | R | R/W | 1’b1 | Cross-zero volume update enable for power amplifier |
| 4 | BIASD | R | R/W | 1’b1 | 1: BIAS power done  0: BIAS power on |
| 3 | MICBIASD | R | R/W | 1’b1 | 1: MICBIAS power done  0: MICBIAS power on |
| 2 | PGA\_STG2\_INVD | R | R/W | 1’b1 | 1: PGA stage2 inverter power done  0: PGA stage2 inverter power on |
| 1 | PGA\_STG2D | R | R/W | 1’b1 | 1: PGA stage2 power done  0: PGA stage2 power on |
| 0 | PGA\_STG1D | R | R/W | 2’b1 | 1: PGA stage1 power done  0: PGA stage1 power on |

Reg17@0x17

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 7:5 | REV | R | R/W | 3’b000 | Reserved |
| 4 | DAC2MIX\_GAIN | R | R/W | 1’b0 | DAC to mixer gain |
| 3:2 | MIX\_IS | R | R/W | 2’b00 | Mixer current settings |
| 1 | RDAC2MIX\_EN | R | R/W | 1’b0 | Right DAC to mixer enable |
| 0 | RSV | - | - | 1’b0 | Reserved |

Reg18@0x18

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 7 | Rev | - | - | 1’b0 | Reserved |
| 6 | VMIDD | R | R/W | 1’b0 | For internal use |
| 5 | VMID\_RAMPD | R | R/W | 1’b0 | For internal use |
| 4 | PGA\_INN\_INP\_EXCHG | R | R/W | 1’b0 | PGA input P/N exchange |
| 3 | PGA\_INN\_EN | R | R/W | 1’b0 | PGA input N enable |
| 2 | PGA\_INP\_EN | R | R/W | 1’b0 | PGA input P enable |
| 1:0 | PGA\_ICTL | R | R/W | 2’b00 | PGA current settings |

Reg19@0x19

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 7:6 | REV | - | - | 2’b00 | Reserved |
| 5:4 | PGA\_STG2\_VOL[1:0] | R | R/W | 2’b00 | PGA boost gain setting |
| 3:0 | PGA\_STG1\_VOL[3:0] | R | R/W | 6’b110111 | PGA gain detail setting, details refer to following tables |

Reg25@0x25

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 7:3 | REV | - | - | 4’b00000 | Reserved |
| 2 | DEC\_CP | R | R/W | 1’b0 | 0: left channel has no data  1: copy the right channel to left channel |
| 1 | REV | - | - | 1’b0 | Reserved |
| 0 | CHANNEL\_CP | R | R/W | 1’b0 | 0: select right channel of I2S  1: select left channel of I2S |

# SYSTEM REGS

## Register

The system regs provide functions to configure pin io mux, system clock selection. Registers are shown as below.

**CLK SET Offset: 04’h**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 31:2 | REV |  |  | ‘d0 | Reserved |
| 1:0 | SYS\_CLK\_SEL | R | R/W | 2’b10 | System Clock (h-clk/p-clk) Setting  2’b00: 6MHz  2’b01: 12MHz  2’b10: 24MHz  2’b11: 48MHz |

**CLK Enable Offset: 08’h**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 31:9 | REV |  |  | ‘d0 | Reserved |
| 8 | EFUSE\_CLK\_EN | R | R/W | 1’b1 | 1= Efuse module Clock enable  0= disable |
| 7 | QSPI\_CLK\_EN | R | R/W | 1’b1 | 1= Qspi module Clock enable  0= disable |
| 6 | CODEC\_CLK\_EN | R | R/W | 1’b1 | 1= Codec module Clock enable  0= disable |
| 5 | OUT\_CLK\_EN | R | R/W | 1’b1 | 1= Ouput module Clock enable  0= disable |
| 4 | GPIO\_CLK\_EN | R | R/W | 1’b1 | 1= GPIO module Clock enable  0= disable |
| 3 | TRNG\_CLK\_EN | R | R/W | 1’b1 | 1= TRNG module Clock enable  0= disable |
| 2 | MM\_CLK\_EN | R | R/W | 1’b1 | 1= Modem module Clock enable  0= disable |
| 1 | UART1\_CLK\_EN | R | R/W | 1’b1 | 1= Uart1 module Clock enable  0= disable |
| 0 | UART0\_CLK\_EN | R | R/W | 1’b1 | 1= Uart0 module Clock enable  0= disable |

**Module Reset Offset: 0C’h**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 31:10 | REV |  |  | ‘d0 | Reserved |
| 9 | EFUSE\_RST | R | R/W | 1’b1 | 1= Release efuse module reset  0= reset efuse module |
| 8 | TRNG\_RST |  |  |  | 1= Release trng module reset  0= reset trng module |
| 7 | QSPI\_RST | R | R/W | 1’b1 | 1= Release qspi module reset  0= reset qspi module |
| 6 | PDM\_RST | R | R/W | 1’b1 | 1= Release pdm module reset  0= reset pdm module |
| 5 | CODEC\_RST | R | R/W | 1’b1 | 1= Release codec module reset  0= reset codec module |
| 4 | ADC\_RST | R | R/W | 1’b1 | 1= Release adc module reset  0= reset adc module |
| 3 | MM\_REG\_RST | R | R/W | 1’b1 | 1= Release modem register reset  0= reset modem register |
| 2 | MM\_RST | R | R/W | 1’b1 | 1= Release modem module reset  0= reset modem module |
| 1 | BB\_CRY\_RST | R | R/W | 1’b1 | 1= Release Baseband crypt module reset  0= reset Baseband crypt module |
| 0 | BB\_MS\_RST | R | R/W | 1’b1 | 1= Release Baseband master module reset  0= reset Baseband master module |

**PORTX PULL Offset: 20’h**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 31:24 | PDULL | R | R/W | 8’h00 | 1= port\_d pull disable  0= port\_d pull enable |
| 23:16 | PCULL | R | R/W | 8’h00 | 1= port\_c pull disable  0= port\_c pull enable |
| 15:8 | PBULL | R | R/W | 8’h00 | 1= port\_b pull disable  0= port\_b pull enable |
| 7:0 | PAULL | R | R/W | 8’h00 | 1= port\_a pull disable  0= port\_a pull enable |

**QSPIULL Offset: 24’h**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 31:5 | REV | R | R/W | ’h00 | Reserved |
| 5:0 | QSPIULL | R | R/W | 6’h00 | 1= QSPI pad pull disable  0= QSPI pad pull enable |

**PORTA FUN\_MUX Offset: 28’h**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 31:28 | PA7\_MUX | R | R/W | 4’h0 | Pin PA7 function mux define. |
| 27:24 | PA6\_MUX | R | R/W | 4’h0 | Pin PA6 function mux define |
| 23:20 | PA5\_MUX | R | R/W | 4’h0 | Pin PA5 function mux define |
| 19:16 | PA4\_MUX | R | R/W | 4’h0 | Pin PA4 function mux define |
| 15:12 | PA3\_MUX | R | R/W | 4’h0 | Pin PA3 function mux define |
| 11:8 | PA2\_MUX | R | R/W | 4’h0 | Pin PA2 function mux define |
| 7:4 | PA1\_MUX | R | R/W | 4’h0 | Pin PA1 function mux define |
| 3:0 | PA0\_MUX | R | R/W | 4’h0 | Pin PA0 function mux define |

Note: Refer to 21.2 to know detail defines

**PORTB FUN\_MUX Offset: 2C’h**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 31:28 | PB7\_MUX | R | R/W | 4’h0 | Pin PB7 function mux define. |
| 27:24 | PB6\_MUX | R | R/W | 4’h0 | Pin PB6 function mux define |
| 23:20 | PB5\_MUX | R | R/W | 4’h0 | Pin PB5 function mux define |
| 19:16 | PB4\_MUX | R | R/W | 4’h0 | Pin PB4 function mux define |
| 15:12 | PB3\_MUX | R | R/W | 4’h0 | Pin PB3 function mux define |
| 11:8 | PB2\_MUX | R | R/W | 4’h0 | Pin PB2 function mux define |
| 7:4 | PB1\_MUX | R | R/W | 4’h0 | Pin PB1 function mux define |
| 3:0 | PB0\_MUX | R | R/W | 4’h0 | Pin PB0 function mux define |

Note: Refer to 21.2 to know detail defines

**PORTC FUN\_MUX Offset: 30’h**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 31:28 | PC7\_MUX | R | R/W | 4’h0 | Pin PC7 function mux define. |
| 27:24 | PC6\_MUX | R | R/W | 4’h0 | Pin PC6 function mux define |
| 23:20 | PC5\_MUX | R | R/W | 4’h0 | Pin PC5 function mux define |
| 19:16 | PC4\_MUX | R | R/W | 4’h0 | Pin PC4 function mux define |
| 15:12 | PC3\_MUX | R | R/W | 4’h0 | Pin PC3 function mux define |
| 11:8 | PC2\_MUX | R | R/W | 4’h0 | Pin PC2 function mux define |
| 7:4 | PC1\_MUX | R | R/W | 4’h0 | Pin PC1 function mux define |
| 3:0 | PC0\_MUX | R | R/W | 4’h0 | Pin PC0 function mux define |

Note: Refer to 21.2 to know detail defines

**PORTD FUN\_MUX Offset: 34’h**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 31:28 | PD7\_MUX | R | R/W | 4’h0 | Pin PD7 function mux define. |
| 27:24 | PD6\_MUX | R | R/W | 4’h0 | Pin PD6 function mux define |
| 23:20 | PD5\_MUX | R | R/W | 4’h0 | Pin PD5 function mux define |
| 19:16 | PD4\_MUX | R | R/W | 4’h0 | Pin PD4 function mux define |
| 15:12 | PD3\_MUX | R | R/W | 4’h0 | Pin PD3 function mux define |
| 11:8 | PD2\_MUX | R | R/W | 4’h0 | Pin PD2 function mux define |
| 7:4 | PD1\_MUX | R | R/W | 4’h0 | Pin PD1 function mux define |
| 3:0 | PD0\_MUX | R | R/W | 4’h0 | Pin PD0 function mux define |

Note: Refer to 21.2 to know detail defines

**EXTI Interruption Mux, Offset: 38’h**

This reg map 15 exti interruption source to different pins, for interruption source is only 16,but pins number is 32.

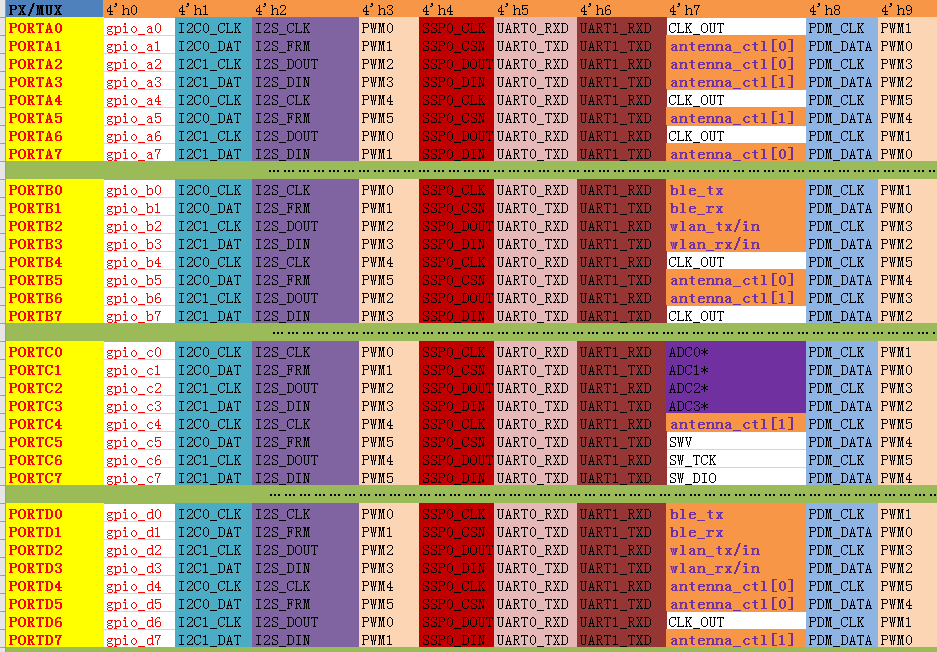
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BIT** | **Name** | **Access** | | **DEFAULT** | **DESCRIPTION** |
| **HW** | **RW** |
| 31:30 | EXT\_INT15\_MUX | R | R/W | 2’b00 | Set which pin to assigned for exti15 isr source.  00: PB7  01: PD7  10: PD0  11: ONKEY |
| 2928 | EXT\_INT14\_MUX | R | R/W | 2’b00 | Set which pin to assigned for exti14 isr source.  00: PB6  01: PD6  10: PD1  11: ONKEY |
| 27:26 | EXT\_INT13\_MUX | R | R/W | 2’b00 | Set which pin to assigned for exti13 isr source.  00: PB5  01: PD5  10: PD2  11: ONKEY |
| 25:24 | EXT\_INT12\_MUX | R | R/W | 2’b00 | Set which pin to assigned for exti12 isr source.  00: PB4  01: PD4  10: PD3  11: ONKEY |
| 23:22 | EXT\_INT11\_MUX | R | R/W | 2’b00 | Set which pin to assigned for exti11 isr source.  00: PB3  01: PD3  10: PD4  11: ONKEY |
| 21:20 | EXT\_INT10\_MUX | R | R/W | 2’b00 | Set which pin to assigned for exti10 isr source.  00: PB2  01: PD2  10: PD5  11: ONKEY |
| 19:18 | EXT\_INT9\_MUX | R | R/W | 2’b00 | Set which pin to assigned for exti9 isr source.  00: PB1  01: PD1  10: PD6  11: ONKEY |
| 17:16 | EXT\_INT8\_MUX | R | R/W | 2’b00 | Set which pin to assigned for exti8 isr source.  00: PB0  01: PD0  10: PD7  11: ONKEY |
| 15:14 | EXT\_INT7\_MUX | R | R/W | 2’b00 | Set which pin to assigned for exti7 isr source.  00: PA7  01: PC7  10: PC0  11: ONKEY |
| 13:12 | EXT\_INT6\_MUX | R | R/W | 2’b00 | Set which pin to assigned for exti6 isr source.  00: PA6  01: PC6  10: PC1  11: ONKEY |
| 11:10 | EXT\_INT5\_MUX | R | R/W | 2’b00 | Set which pin to assigned for exti5 isr source.  00: PA5  01: PC5  10: PC2  11: ONKEY |
| 9:8 | EXT\_INT4\_MUX | R | R/W | 2’b00 | Set which pin to assigned for exti4 isr source.  00: PA4  01: PC4  10: PC3  11: ONKEY |
| 7:6 | EXT\_INT3\_MUX | R | R/W | 2’b00 | Set which pin to assigned for exti3 isr source.  00: PA3  01: PC3  10: PC4  11: ONKEY |
| 5:4 | EXT\_INT2\_MUX | R | R/W | 2’b00 | Set which pin to assigned for exti2 isr source.  00: PA2  01: PC2  10: PC5  11: ONKEY |
| 3:2 | EXT\_INT1\_MUX | R | R/W | 2’b00 | Set which pin to assigned for exti1 isr source.  00: PA1  01: PC1  10: PC6  11: ONKEY |
| 1:0 | EXT\_INT0\_MUX | R | R/W | 2’b00 | Set which pin to assigned for exti0 isr source.  00: PA0  01: PC0  10: PC7  11: ONKEY |

## Port IO MUX

Chip pin {PD[7:0], PC[7:0], PB[7:0], PA[7:0]} can be configured according to below picture as different digital functions.

Please refer to system reg **PORTA FUN\_MUX Offset: 28’h, PORTB FUN\_MUX Offset: 2C’h**

**PORTC FUN\_MUX Offset: 30’h, PORTD FUN\_MUX Offset: 34’h** to know how to configure pins.



# GPIO

## General purpose ports

The number of ports and GPIOs per port might vary with product variant and package.

The GPIO block comprises eight programmable input/output lines. When the software control mode is enabled, data and control for these lines are provided by a data register and a data direction register. On reads, the data register contains the current status of the GPIO pins, whether they are configured as input or output. Writing to the data register only affects the pins that are configured as outputs.

There are maximum 32 pad pins can work as GPIO function.

## Register

**GPIOORTA\_DATA, Addr: GPIOAB\_BASE + 00’h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 7:0 | PA\_VAL | RW | 0x00 | GPIO PortA data register, each bit represent each pin from PA7 to PA0.  Write value to this reg to set pin output value.  Read this reg to get pin input value. |

**GPIOORTB\_DATA, Addr: GPIOAB\_BASE +04’h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 7:0 | PB\_VAL | RW | 0x00 | GPIO PortB data register, each bit represent each pin from PB7 to PB0.  Write value to this reg to set pin output value.  Read this reg to get pin input value. |

**GPIOORTA\_DIR, Addr: GPIOAB\_BASE +08’h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 7:0 | PA\_DIR | RW | 0x00 | GPIO PortA direction register, each bit represent each pin IO direction from PA7 to PA0.  Write value to this reg to set pin i/o direction  1= input  0= output |

**GPIOORTB\_DIR, Addr: GPIOAB\_BASE +0C’h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 7:0 | PB\_DIR | RW | 0x00 | GPIO PortB direction register, each bit represent each pin IO direction from PB7 to PB0.  Write value to this reg to set pin i/o direction  1= input  0= output |

**GPIOORTC\_DATA, Addr: GPIOCD\_BASE +00’h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 7:0 | PC\_VAL | RW | 0x00 | GPIO PortC data register, each bit represent each pin from PC7 to PC0.  Write value to this reg to set pin output value.  Read this reg to get pin input value. |

**GPIOORTD\_DATA, Addr: GPIOCD\_BASE+04’h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 7:0 | PD\_VAL | RW | 0x00 | GPIO PortD data register, each bit represent each pin from PD7 to PD0.  Write value to this reg to set pin output value.  Read this reg to get pin input value. |

**GPIOORTC\_DIR, Addr: GPIOCD\_BASE+08’h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 7:0 | PC\_DIR | RW | 0x00 | GPIO PortC direction register, each bit represent each pin IO direction from PC7 to PC0.  Write value to this reg to set pin i/o direction  1= input  0= output |

**GPIOORTD\_DIR, Addr: GPIOCD\_BASE+0C’h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 7:0 | PD\_DIR | RW | 0x00 | GPIO PortD direction register, each bit represent each pin IO direction from PD7 to PD0.  Write value to this reg to set pin i/o direction  1= input  0= output |

## GPIO output

If the corresponding bit in GPIOORT\_DIR register is set, the corresponding GPIO is used for output function. In this configuration, writing to the GPIOORT\_DATA register changes the output value.

## GPIO input

If the corresponding bit in GPIOORT\_DIR register is clear, the corresponding GPIO is used for input function.

# EXTI(external interrupt)

## Register

**EXT\_INT\_EN, Addr: EXT\_BASE+00’h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 15:0 | EXTI\_INT\_EN | RW | 0x0000 | External interrupt source enable  There are totally 15 exti interruption source. Bit[15:0] represent one interrupt source number from 15 to 0. |

**EXT\_INT\_STATUS, Addr: EXT\_BASE+04’h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 15:0 | EXTI\_INT\_ST | RW | 0x0000 | External interrupt status  Bit[15:0] represent one interrupt source number from 15 to 0.  Write this register is to clear corresponding interrupt |

**EXT\_INT\_TYPE, Addr: EXT\_BASE+08’h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 31:30 | EXTI15\_INT\_TYP | RW | 00 | External interrupt trigger type for index 15  00: low level trigger  01: high level trigger  10: positive edge trigger  11: negative edge trigger |
| 29:28 | EXTI14\_INT\_TYP | RW | 00 | External interrupt trigger type for index 14  00: low level trigger  01: high level trigger  10: positive edge trigger  11: negative edge trigger |
| 27:26 | EXTI13\_INT\_TYP | RW | 00 | External interrupt trigger type for index 13  00: low level trigger  01: high level trigger  10: positive edge trigger  11: negative edge trigger |
| 25:24 | EXTI12\_INT\_TYP | RW | 00 | External interrupt trigger type for index 12  00: low level trigger  01: high level trigger  10: positive edge trigger  11: negative edge trigger |
| 23:22 | EXTI11\_INT\_TYP | RW | 00 | External interrupt trigger type for index 11  00: low level trigger  01: high level trigger  10: positive edge trigger  11: negative edge trigger |
| 21:20 | EXTI10\_INT\_TYP | RW | 00 | External interrupt trigger type for index 10  00: low level trigger  01: high level trigger  10: positive edge trigger  11: negative edge trigger |
| 19:18 | EXTI9\_INT\_TYP | RW | 00 | External interrupt trigger type for index 9  00: low level trigger  01: high level trigger  10: positive edge trigger  11: negative edge trigger |
| 17:16 | EXTI8\_INT\_TYP | RW | 00 | External interrupt trigger type for index 8  00: low level trigger  01: high level trigger  10: positive edge trigger  11: negative edge trigger |
| 15:14 | EXTI7\_INT\_TYP | RW | 00 | External interrupt trigger type for index 7  00: low level trigger  01: high level trigger  10: positive edge trigger  11: negative edge trigger |
| 13:12 | EXTI6\_INT\_TYP | RW | 00 | External interrupt trigger type for index 6  00: low level trigger  01: high level trigger  10: positive edge trigger  11: negative edge trigger |
| 11:10 | EXTI5\_INT\_TYP | RW | 00 | External interrupt trigger type for index 5  00: low level trigger  01: high level trigger  10: positive edge trigger  11: negative edge trigger |
| 9:8 | EXTI4\_INT\_TYP | RW | 00 | External interrupt trigger type for index 4  00: low level trigger  01: high level trigger  10: positive edge trigger  11: negative edge trigger |
| 7:6 | EXTI3\_INT\_TYP | RW | 00 | External interrupt trigger type for index 3  00: low level trigger  01: high level trigger  10: positive edge trigger  11: negative edge trigger |
| 5:4 | EXTI2\_INT\_TYP | RW | 00 | External interrupt trigger type for index 2  00: low level trigger  01: high level trigger  10: positive edge trigger  11: negative edge trigger |
| 3:2 | EXTI1\_INT\_TYP | RW | 00 | External interrupt trigger type for index 1  00: low level trigger  01: high level trigger  10: positive edge trigger  11: negative edge trigger |
| 1:0 | EXTI0\_INT\_TYP | RW | 00 | External interrupt trigger type for index 0  00: low level trigger  01: high level trigger  10: positive edge trigger  11: negative edge trigger |

**EXT\_INT\_CONTROL, Addr: EXT\_BASE+ 0C’h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default** | **Description** |
| 31:20 | REV | R | 00 | Reserved |
| 19:4 | DEB\_CLK\_DIV | RW | 00 | Debunce clock prescaler value.  Debunce clock = PCLK/(1+ Prescaler value)  Where,  Prescaler value is bit[19:4] value  PCLK is system clock, refer to **CLK SET Offset: 04’h**(chapter 21.2) |
| 3:0 | DEB\_CNT | RW | 00 | Debunce counter value.  Unit: Debunce clock period, decided by bit[19:4]  If the external interrupt source last longer than the value set in this register, the corresponding external interrupt status would set. (only valid for level trigger mode) |

Usually the GPIO input function is the precondition for interrupt operation.

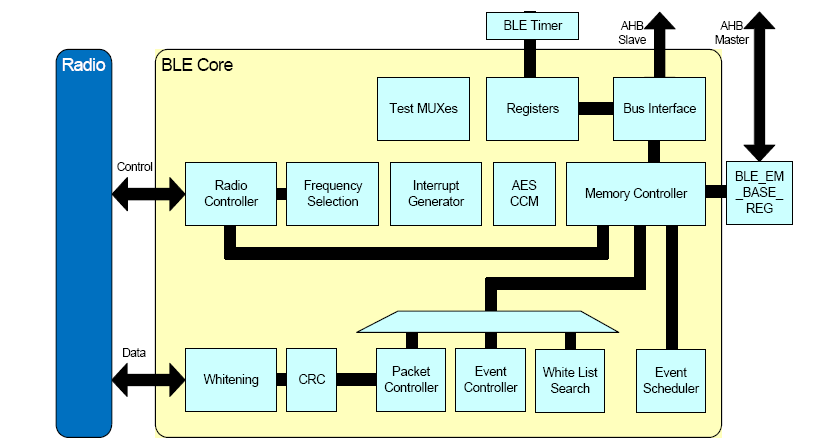
The interrupt section of the GPIO is controlled by a set of registers in the 22 chapter. You can select the source of the interrupt, its polarity and edge properties.

For example, if you want to generate edge-triggered interrupts you must perform the following initialization sequence:

* Choose GPIO port to as GPIO function to generate interrupt, refer to chapter 19.1 **System Registe**r
* Set corresponding GPIO direction to input function
* Configure the debounce time in the **GPIO\_EXT\_INT\_CONTROL Register**
* Choose the trigger type in the **GPIO\_EXT\_INT\_TYPE Register**
* Enable corresponding GPIO interrupt in the **GPIO\_EXT\_INT\_EN Register**

# BLE Core

The BLE (Bluetooth Low Energy) core is a qualified Bluetooth 5.0 baseband controller compatible with Bluetooth Low Energy specification and it is in charge of packet encoding/decoding and frame scheduling.



# Memory Map

This section contains a detailed view of the FR801xH memory map.

|  |  |  |
| --- | --- | --- |
| **Address** | **Description** | **AMBA** |
| 0x00000000 | ROM | AHB |
| 0x01000000 | QSPI FLASH | AHB |
| 0x20000000 | DATA RAM | AHB |
| 0x40000000 | BLE Baseband Register | AHB |
| 0x40002000 | BLE Exchange memory | AHB |
| 0x50000000 | System Registers | APB |
| 0x50001000 | MODEM | APB |
| 0x50002000 | TIMER | APB |
| 0x50003000 | I2C0 | APB |
| 0x50003800 | I2C1 | APB |
| 0x50004000 | SSP | APB |
| 0x50005000 | UART0 | APB |
| 0x50005800 | UART1 | APB |
| 0x50006000 | GPIOAB | APB |
| 0x50006400 | GPIOCD | APB |
| 0x50006800 | EXTI | APB |
| 0x50007000 | I2S | APB |
| 0x50008000 | EFUSE | APB |
| 0x50009000 | SAR\_ADC | APB |
| 0x5000a000 | CACHE | APB |
| 0x5000b000 | QSPI\_APB | APB |
| 0x5000c000 | TRNG | APB |
| 0x5000d000 | PDM | APB |
| 0x5000e000 | PWM | APB |
| 0x5000f000 | FRSPIM | APB |

The RAM comprises 5 physical RAM cells, all with content retaining capability. It includes three data RAM, one cache RAM and one exchange RAM. Each of the RAM sections have separate power control for system ON and system OFF mode operation, which is configured via PMU register.

The following table shows the details of the retention data RAM.

|  |  |  |
| --- | --- | --- |
| **Index** | **Address** | **Size** |
| 1 | 0x20000000 ~ 0x2000BFFF | 48K |

# Electrical Characteristics

## **Absolute Maximum Ratings**

Continuous operation at or beyond these conditions may permanently damage the device

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rating | | Min | Max | Unit |
| Storage Temperature | | -40 | 125 | ℃ |
| Core Supply Voltage | | 0.9 | 1.3 | V |
| I/O Voltage | ALDO\_OUT | 2.1 | 3.5 | V |
| Supply Voltage | VBAT | 1.8 | 4.3 | V |
| VCHG | 4.75 | 5.25 | V |

Note: the ALDO\_OUT will be same with VBAT if configured ALDO\_OUT value is larger than VBAT.

## **Recommended Operating Conditions**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operating Condition** | | **Min** | **Typ** | **Max** | **Unit** |
| Operating Temperature Range | | -40 | 20 | 125 | ℃ |
| Core Supply Voltage | | 0.9 | 1.2 | 1.3 | V |
| I/O Voltage | ALDO\_OUT | 2.1 | 2.9 | 3.5 | V |
| Supply Voltage | VBAT | 1.8 | 3.3 | 4.3 | V |
| VCHG | 4.75 | 5 | 5.25 | V |

## **IO Input/Output Electrical Logical Characteristics**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **IO input characteristics** | | | | | | |
| **Symbol** | **Parameter** | **Min** | **Typ** | **Max** | **Unit** | **Test Conditions** |
| VIL | Low-Level Input Voltage | -0.3 | \_ | 0.3\* VDDIO | V | VDDIO = 3.3V |
| VIH | High-Level Input Voltage | 0.7\*VDDIO | \_ | VDDIO+0.3 | V | VDDIO = 3.3V |
| **IO output characteristics** | | | | | | |
| VOL | Low-Level Output Voltage | \_ | \_ | 0.33 | V | VDDIO = 3.3V |
| VOH | High-Level Output Voltage | 1.8 | \_ | \_ | V | VDDIO = 3.3V |

## Internal Resistor Characteristics

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Port** | **General Output** | **High Drive** | **Internal Pull-Up Resistor** | **Internal Pull-Down Resistor** | **Comment** |
| PA PB PC PD | 8mA | - | 50K | - | internal pull-up resistance accuracy ±20% |

## Audio CODEC

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Digital to Analogue Converter(Mono) | | | | | |
| PARAMETER | CONDITIONS | MIN | TYP | MAX | UNIT |
| Resolution | - | - | - | 20 | Bits |
| Sampling frequency (Fs)\* | The synchronized clock signal | 8 |  | 48 | kHz |
| SNR  (Signal to Noise Ratio) | Fin=1kHz B/W=20Hz—20KHz  A-Weighted THD<0.01%  Fs(8K,16K,32K,44.1K,48K) |  | 92 |  | dB |
| Digital Gain | Digital Gain Resolution=1/48dB | -48 |  | 32 | dB |
| Analogue Gain | Analog Gain Resolution = 3dB | 0 |  | -30 | dB |
| Output voltage fulls-cale | VDDA=2.9V |  | 1500 |  | mV |
| Stopband attenuation |  | 65 |  |  | dB |
|  | | | | | |
| Analog to Digital Converter(Mono) | | | | | |
| PARAMETER | CONDITIONS | MIN | TYP | MAX | UNIT |
| Resolution | - | - | - | 16 | Bits |
| Sampling frequency (Fs)\* | The synchronized clock signal | 8 |  | 48 | kHz |
| Signal to Noise Ratio | A-weighted |  | 79 |  | dBFS |
| W/O weighting |  | 79 |  | dBFS |
| Digital Gain | Digital Gain Resolution=1/48dB | -48 |  | 32 | dB |
| Analogue Gain | Analog Gain Resolution = 3dB | 0 |  | 30 | dB |

## Crystal oscillator

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CLOCK SOURCE | Min | Typ | Max | Unit |
| Main Crystal OSC(24Mhz) for Bluetooth RF application | | | | |
| Clock Frequency | 24 | 24 | 24 | MHz |
| Digital rim range |  | 7.5 |  | pf |
| Trim step size |  | 0.1 |  | pf |
| Tolerance |  | +-10 |  | ppm |
| Note: XTAL Load capacitance = 7.5pf | | | | |

## BT Characteristics

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter** | **Min** | **Typ** | **Max** | **Unit** | **Test Conditions** |
| RF frequency range | 2402 |  | 2480 | MHz | 25 , ℃ Power Supply Voltage=3.3V 2440MHz |
| RF Transmit Power | -20 | 0 | 10 | dBm |
| Receiver sensitivity |  | -98 |  | dBm |
| Receiver sensitivity(1Mbps) |  | -94 |  | dBm |

## Power Consumption

|  |  |  |  |
| --- | --- | --- | --- |
| Operation Mode | Average | Maximum | Unit |
| TX peek current (0dB) |  | 8 | mA |
| RX peek current |  | 9.7 | mA |
| Deep sleep current (include 48K retention RAM) | 6.1 |  | μA |
| Power off | 2.7 |  | μA |

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Description** |
| 1.0 | 2019-12-31 | Freqchip R&D | First test vision |