

General Description

The DA14582 integrated circuit has a fully integrated radio transceiver, baseband processor for *Bluetooth® low energy*, and includes an audio CODEC.

The DA14582 is optimized for remote control units (RCU) requiring support for voice commands and motion/gesture recognition. Its integrated analog wide band audio CODEC provides native support for analog microphones thereby reducing the total number of components of the system while its optimized package enables designs using single-layer FR1 PCBs furthermore contributing to the reduction of the cost of the system.

The DA14582 supports a flexible memory architecture for storing Bluetooth profiles and custom application code, which can be updated over the air (OTA). The qualified Bluetooth low energy protocol stack is stored in a dedicated ROM. All software runs on the ARM® Cortex®-M0 processor via a simple scheduler.

The Bluetooth low energy firmware includes the L2CAP service layer protocols, Security Manager (SM), Attribute Protocol (ATT), the Generic Attribute Profile (GATT) and the Generic Access Profile (GAP). All profiles published by the Bluetooth SIG as well as custom profiles are supported.

The transceiver interfaces directly to the antenna and is fully compliant with the *Bluetooth 4.2* standard.

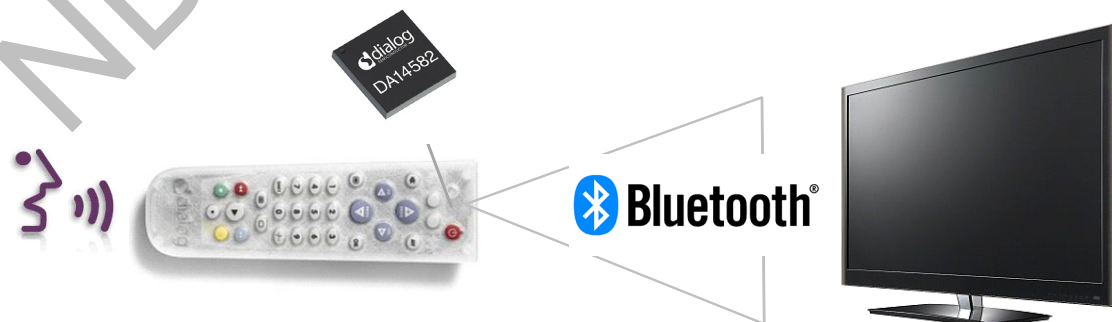
The DA14582 has dedicated hardware for the Link Layer implementation of *Bluetooth® low energy* and interface controllers for enhanced connectivity capabilities.

Features

- Complies with *Bluetooth V4.2*, ETSI EN 300 328 and EN 300 440 Class 2 (Europe), FCC CFR47 Part 15 (US) and ARIB STD-T66 (Japan)
- Processing power

- 16 MHz 32 bit ARM Cortex-M0 with SWD I/F
- Dedicated Link Layer Processor
- AES-128 bit encryption Processor
- Memories
 - 32 kB One-Time-Programmable (OTP) memory
 - 42 kB System SRAM
 - 84 kB ROM
 - 8 kB Retention SRAM
- Power management
 - Integrated Buck DC-DC converter
 - P0, P1, P2 and P3 ports with 3.3 V tolerance
 - Easy decoupling of only 4 supply pins
 - Supports coin (typ. 3.0 V) and alkaline (typ. 1.5 V) battery cells
 - 10-bit ADC for battery voltage measurement
- Digital controlled oscillators
 - 16 MHz crystal (± 20 ppm max) and RC oscillator
 - 32 kHz crystal (± 50 ppm, ± 500 ppm max) and RCX oscillator
- General purpose, Capture and Sleep timers
- Digital interfaces
 - 29 General purpose I/Os
 - 2 UARTs with hardware flow control up to 1 MBd
 - SPI+™ interface
 - I2C bus at 100 kHz, 400 kHz
 - 3-axis capable Quadrature Decoder
- Analog interfaces
 - 4-channel 10-bit ADC
 - 14 bits wide band CODEC with microphone and 28 Ω loudspeaker analog front-end
- Radio transceiver
 - Fully integrated 2.4 GHz CMOS transceiver
 - Single wire antenna: no RF matching or RX/TX switching required
 - Supply current at VBAT3V:
 - TX: 3.4 mA, RX: 3.7 mA (with ideal DC-DC)
 - 0 dBm transmit output power
 - -20 dBm output power in "Near Field Mode"
 - -93 dBm receiver sensitivity
- QFN 56 pins, 8 mm x 8 mm package

System Diagram



Contents

| | | | |
|---|-----------|---|-----------|
| 1 Block Diagram | 4 | 13 I2C Interface | 34 |
| 2 Pinout | 5 | 13.1 I2C BUS TERMS | 34 |
| 3 Ordering Information | 9 | 13.1.1 Bus Transfer Terms | 35 |
| 4 System Overview | 10 | 13.2 I2C BEHAVIOR | 35 |
| 4.1 INTERNAL BLOCKS | 10 | 13.2.1 START and STOP Generation | 36 |
| 4.2 FUNCTIONAL MODES | 10 | 13.2.2 Combined Formats | 36 |
| 4.3 OTP MEMORY LAYOUT | 11 | 13.3 I2C PROTOCOLS | 36 |
| 4.3.1 OTP Header | 11 | 13.3.1 START and STOP Conditions | 36 |
| 4.4 SYSTEM START PROCEDURE | 12 | 13.3.2 Addressing Slave Protocol | 36 |
| 4.4.1 Power/Wake-up Sequence | 13 | 13.3.3 Transmitting and Receiving Protocols | 37 |
| 4.4.2 OTP Mirroring | 14 | 13.4 MULTIPLE MASTER ARBITRATION | 39 |
| 4.4.3 BootROM sequence | 15 | 13.5 CLOCK SYNCHRONIZATION | 40 |
| 4.5 POWER SUPPLY CONFIGURATION | 17 | 13.6 OPERATION MODES | 41 |
| 4.5.1 Power Domains | 17 | 13.6.1 Slave Mode Operation | 41 |
| 4.5.2 Power Modes | 18 | 13.6.2 Master Mode Operation | 43 |
| 4.5.3 Retention Registers | 18 | 13.6.3 Disabling the I2C Controller | 43 |
| 5 Reset | 20 | 14 UART | 44 |
| 5.1 POR, HW AND SW RESET | 20 | 14.1 UART (RS232) SERIAL PROTOCOL | 45 |
| 6 ARM Cortex-M0 | 21 | 14.2 IRDA 1.0 SIR PROTOCOL | 45 |
| 6.1 INTERRUPTS | 22 | 14.3 CLOCK SUPPORT | 46 |
| 6.2 SYSTEM TIMER (SYSTICK) | 23 | 14.4 INTERRUPTS | 47 |
| 6.3 WAKE-UP INTERRUPT CONTROLLER | 23 | 14.5 PROGRAMMABLE THRE INTERRUPT | 47 |
| 6.4 REFERENCE | 23 | 14.6 SHADOW REGISTERS | 49 |
| 7 AMBA Bus Overview | 24 | 14.7 DIRECT TEST MODE | 49 |
| 8 Patch Block | 25 | 15 SPI+ Interface | 50 |
| 9 Memory Map | 26 | 15.1 OPERATION WITHOUT FIFOS | 50 |
| 10 Memory Controller | 28 | 15.2 9 BITS MODE | 51 |
| 10.1 ARBITRATION | 28 | 16 Quadrature Decoder | 54 |
| 11 Clock Generation | 29 | 17 Wake-Up Timer | 55 |
| 11.1 CRYSTAL OSCILLATORS | 29 | 18 General Purpose Timers | 56 |
| 11.1.1 Frequency Control (16 MHz Crystal) | 29 | 18.1 TIMER 0 | 56 |
| 11.1.2 Automated Trimming Mechanism | 29 | 18.2 TIMER 2 | 58 |
| 11.2 RC OSCILLATORS | 30 | 19 Watchdog Timer | 60 |
| 11.2.1 Frequency Calibration | 30 | 20 Keyboard Controller | 61 |
| 11.3 SYSTEM CLOCK GENERATION | 31 | 20.1 KEYBOARD SCANNER | 61 |
| 11.4 GENERAL CLOCK CONSTRAINTS | 32 | 20.2 GPIO INTERRUPT GENERATOR | 61 |
| 12 OTP Controller | 33 | 21 Input/Output Ports | 63 |
| 12.1 OPERATING MODES | 33 | 21.1 PROGRAMMABLE PIN ASSIGNMENT | 63 |
| 12.2 AHB MASTER INTERFACE | 33 | 21.2 GENERAL PURPOSE PORT REGISTERS | 63 |
| | | 21.2.1 Port Data Register | 64 |
| | | 21.2.2 Port Set Data Output Register | 64 |
| | | 21.2.3 Port Reset Data Output Register | 64 |
| | | 21.3 FIXED ASSIGNMENT FUNCTIONALITY | 64 |

| | | | |
|--|-----------|--|------------|
| 22 General Purpose ADC | 65 | TIVE MODE | 88 |
| 22.1 INPUT CHANNELS AND INPUT SCALE ... | 65 | 25.4.1 Switching at Anchor Points | 88 |
| 22.2 STARTING THE ADC AND SAMPLING RATE | | 25.4.2 Switching Due to an External Event. . . | 90 |
| | 65 | | |
| 22.3 NON-IDEAL EFFECTS | 66 | 26 Radio | 91 |
| 22.4 CHOPPING | 66 | 26.1 RECEIVER | 91 |
| 22.5 OFFSET CALIBRATION | 66 | 26.2 SYNTHESIZER | 91 |
| 22.6 ZERO-SCALE ADJUSTMENT | 67 | 26.3 TRANSMITTER | 91 |
| 22.7 COMMON MODE ADJUSTMENT | 67 | 26.4 RFIO | 91 |
| 22.8 INPUT IMPEDANCE, INDUCTANCE AND IN- | | 26.5 BIASING | 91 |
| PUT SETTLING | 67 | 26.6 CONTROL | 91 |
| 22.9 DELAY COUNTER | 68 | 27 Registers | 92 |
| 23 Power Management | 69 | 28 CODEC Registers | 230 |
| 24 CODEC with Analog Front-End | 71 | 29 Specifications | 240 |
| 24.1 ARCHITECTURE | 72 | 30 CODEC Specifications | 251 |
| 24.1.1 Pin Assignment CODEC to BLE Core . | 72 | 30.1 CODEC DC CHARACTERISTICS | 251 |
| 24.1.2 Signal Flow | 72 | 30.2 ANALOG FRONTEND SPECIFICATIONS . | 253 |
| 24.2 SPI INTERFACE | 73 | 30.3 CODEC AC CHARACTERISTICS | 257 |
| 24.2.1 Architecture | 73 | 30.4 CODEC TIMING CHARACTERISTICS | 261 |
| 24.2.2 SPI Interface | 73 | | |
| 24.2.3 SPI Command Interpreter | 74 | 31 Package Information | 262 |
| 24.2.4 Bus Master | 75 | 31.1 MOISTURE SENSITIVITY LEVEL (MSL) . . | 262 |
| 24.3 CLOCK TREE | 76 | 31.2 SOLDERING INFORMATION | 262 |
| 24.3.1 PLL Clock Switching Procedure | 77 | 31.3 PACKAGE OUTLINES | 263 |
| 24.4 DMA CONTROLLER | 78 | | |
| 24.4.1 DMA Channel Operation | 78 | | |
| 24.4.2 DMA Arbitration | 79 | | |
| 24.5 BANDGAP REFERENCE CIRCUIT | 79 | | |
| 24.6 CODEC INPUT/OUTPUT PORTS (GPIO) . . | 80 | | |
| 24.6.1 Pull-Up/Pull-Down Resistors | 81 | | |
| 24.6.2 Port Direction | 81 | | |
| 24.6.3 Reference Voltage for Input Level | 81 | | |
| 24.7 CODEC AND ANALOG FRONTEND | 82 | | |
| 24.7.1 Loudspeaker Connection | 82 | | |
| 24.7.2 Microphone Connection | 82 | | |
| 24.7.3 Microphone Supply | 82 | | |
| 24.7.4 CODEC Side Tone | 82 | | |
| 24.7.5 CODEC Offset Compensation | 82 | | |
| 24.8 CODEC MEMORY MAP | 84 | | |
| 25 BLE Core | 85 | | |
| 25.1 EXCHANGE MEMORY | 85 | | |
| 25.2 PROGRAMMING BLE WAKE UP IRQ | 87 | | |
| 25.3 SWITCH FROM ACTIVE MODE TO DEEP | | | |
| SLEEP MODE | 87 | | |
| 25.4 SWITCH FROM DEEP SLEEP MODE TO AC- | | | |

1 Block Diagram

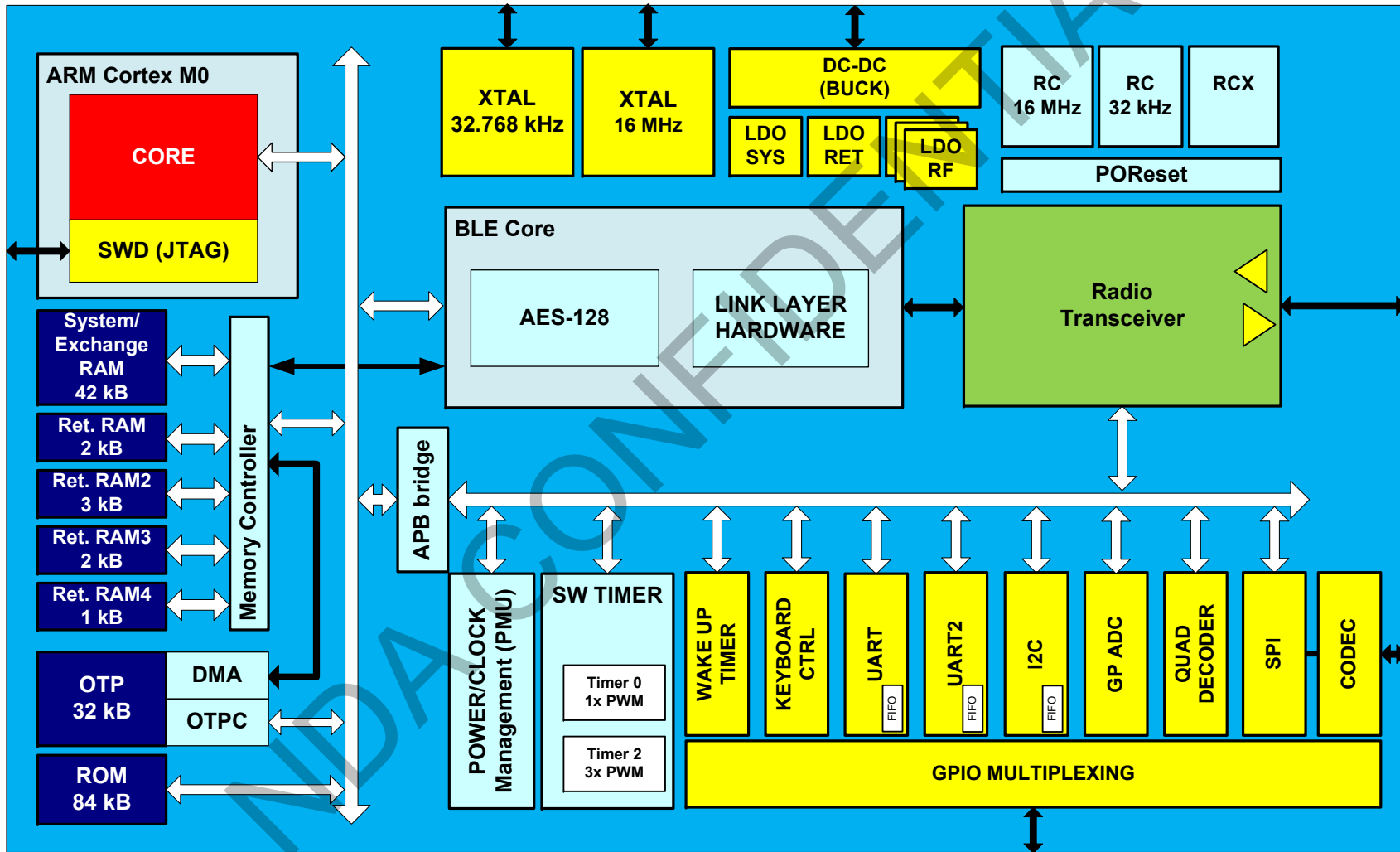


Figure 1: DA14582 Block Diagram

2 Pinout

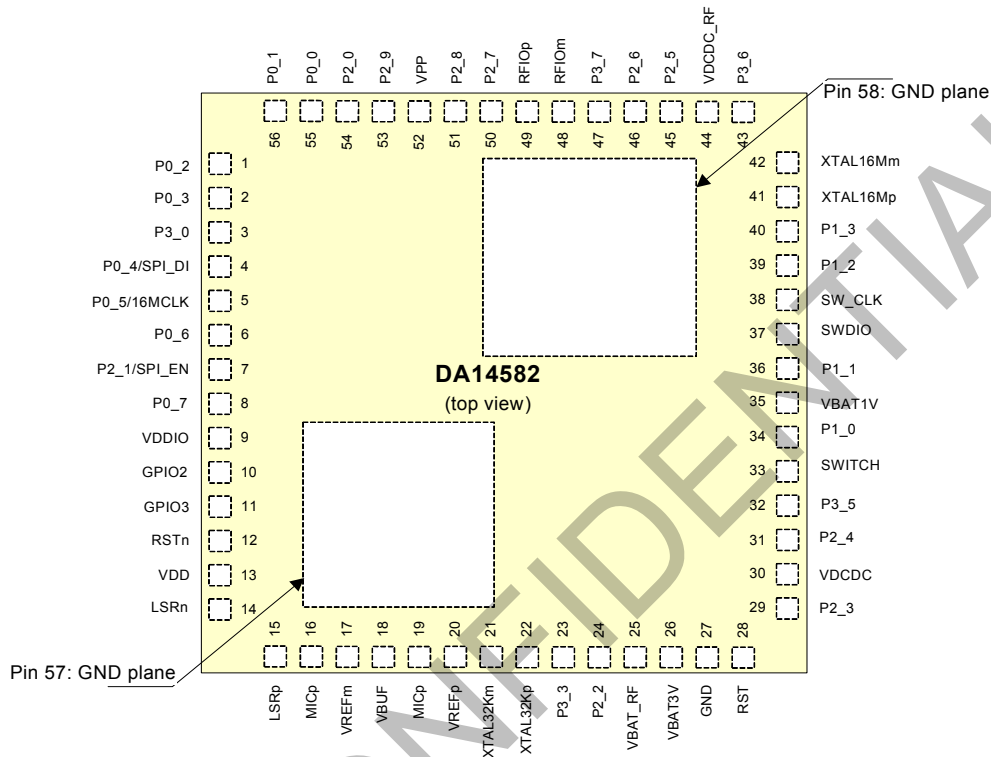


Figure 2: QFN56 Pin Assignment

Table 1: Pin Description

| Pin Name | Type | Drive (mA) | Reset State | Description |
|-----------------------------|------|------------|-------------|--|
| General Purpose I/Os | | | | |
| P0_0 | DIO | 4.8 | I-PD | INPUT/OUTPUT with selectable pull up/down resistor. Pull-down enabled during and after reset. General purpose I/O port bit or alternate function nodes. Contains state retention mechanism during power down. |
| P0_1 | DIO | | I-PD | |
| P0_2 | DIO | | I-PD | |
| P0_3/ | DIO | | I-PD | |
| P0_4/SPI_DI | DIO | | I-PD | |
| P0_5/16MCLK | DIO | | I-PD | |
| P0_6 | DIO | | I-PD | |
| P0_7 | DIO | | I-PD | |
| P1_0 | DIO | 4.8 | I-PD | INPUT/OUTPUT with selectable pull up/down resistor. Pull-down enabled during and after reset. General purpose I/O port bit or alternate function nodes. Contains state retention mechanism during power down. This signal is the JTAG clock by default This signal is the JTAG data I/O by default |
| P1_1 | DIO | | I-PD | |
| P1_2 | DIO | | I-PD | |
| P1_3 | DIO | | I-PD | |
| P1_4/SWCLK | DIO | | I-PD | |
| P1_5/SW_DIO | DIO | | I-PU | |
| P2_0 | DIO | 4.8 | I-PD | INPUT/OUTPUT with selectable pull up/down resistor. Pull-down enabled during and after reset. General purpose I/O port bit or alternate function nodes. Contains state retention mechanism during power down. |
| P2_1/SPI_EN | DIO | | I-PD | |
| P2_2 | DIO | | I-PD | |
| P2_3 | DIO | | I-PD | |
| P2_4 | DIO | | I-PD | |
| P2_5 | DIO | | I-PD | |
| P2_6 | DIO | | I-PD | |
| P2_7 | DIO | | I-PD | |
| P2_8 | DIO | | I-PD | |
| P2_9 | DIO | | I-PD | |
| P3_0 | DIO | 4.8 | I-PD | INPUT/OUTPUT with selectable pull up/down resistor. Pull-down enabled during and after reset. General purpose I/O port bit or alternate function nodes. Contains state retention mechanism during power down. |
| P3_1 (Table 2) | DIO | | I-PD | |
| P3_2 (Table 2) | DIO | | I-PD | |
| P3_3 | DIO | | I-PD | |
| P3_5 | DIO | | I-PD | |
| P3_6 | DIO | | I-PD | |
| P3_7 | DIO | | I-PD | |
| Debug Interface | | | | |
| SW_DIO/P1_5 | DIO | 4.8 | I-PU | INPUT/OUTPUT. JTAG Data input/output. Bidirectional data and control communication. Can also be used as a GPIO |
| SW_CLK/P1_4 | DIO | 4.8 | I-PD | INPUT JTAG clock signal. Can also be used as a GPIO |
| Clocks | | | | |
| XTAL16Mp | AI | | | INPUT. Crystal input for the 16 MHz XTAL |
| XTAL16Mm | AO | | | OUTPUT. Crystal output for the 16 MHz XTAL |
| XTAL32kp | AI | | | INPUT. Crystal input for the 32.768 kHz XTAL |
| XTAL32km | AO | | | OUTPUT. Crystal output for the 32.768 kHz XTAL |

Table 1: Pin Description

| Pin Name | Type | Drive (mA) | Reset State | Description |
|---|--------------|------------|-------------|--|
| 16MCLK/P0_5 | DIO | 4.8 | | OUTPUT. Buffered 16 MHz output This clock is used as reference clock for the on-chip CODEC which is enabled via a test register bit 0x500030F0[0]=1. However, when enabled the following pins also output certain clock signals. In this case these port pins can only be used as general purpose inputs by setting Pxy_MODE_REG[PID]=0 and Pxy_MODE_REG[PUPD] unequal to 3: P0_6: XTAL32K P0_7: RC16M P1_0: RC32K P1_1: RC32K_low_jitter |
| SPI Bus Interface (Refer to Table 2 for the fixed SPI pin assignment during CODEC operation) | | | | |
| SPI_CLK | DO | | | INPUT/OUTPUT. SPI Clock. Mapped on Px ports. |
| SPI_DI | DI | | | INPUT. SPI Data input. Mapped on Px ports. |
| SPI_DO | DO | | | OUTPUT. SPI Data output. Mapped on Px ports |
| SPI_EN | DI/DO | | | INPUT/OUTPUT. SPI Clock enable. Mapped on Px ports |
| I2C Bus Interface | | | | |
| SDA | DIO/ DIOD | | | INPUT/OUTPUT. I2C bus Data with open drain port. Mapped on Px ports |
| SCL | DIO/ DIOD | | | INPUT/OUTPUT. I2C bus Clock with open drain port. In open drain mode, SCL is monitored to support bit stretching by a slave. Mapped on Px ports. |
| UART Interface | | | | |
| UTX | DO | | | OUTPUT. UART transmit data. Mapped on Px ports |
| URX | DI | | | INPUT. UART receive data. Mapped on Px ports |
| URTS | DO | | | OUTPUT. UART Request to Send. Mapped on Px ports |
| UCTS | DI | | | INPUT. UART Clear to Send. Mapped on Px ports |
| UTX2 | DO | | | OUTPUT. UART 2 transmit data. Mapped on Px ports |
| URX2 | DI | | | INPUT. UART 2 receive data. Mapped on Px ports |
| URTS2 | DO | | | OUTPUT. UART 2 Request to Send. Mapped on Px ports |
| UCTS2 | DI | | | INPUT. UART 2 Clear to Send. Mapped on Px ports |
| Analog Interface | | | | |
| ADC[0] | AI | | | INPUT. Analog to Digital Converter input 0. Mapped on P0[0] |
| ADC[1] | AI | | | INPUT. Analog to Digital Converter input 1. Mapped on P0[1] |
| ADC[2] | AI | | | INPUT. Analog to Digital Converter input 2. Mapped on P0[2] |
| ADC[3] | AI | | | INPUT. Analog to Digital Converter input 3. Mapped on P0[3] |
| CODEC Interface | | | | |
| GPIO2, GPIO3 | DIO | 8 | I-PU | INPUT/OUTPUT. CODEC General Purpose I/O ports with selectable pull up/down resistor. Supplied from VDDIO. |
| RSTn | DI | | | INPUT. CODEC reset signal (active low). After startup this pin must be kept LOW for at least 1 us (while the clock is active) to guarantee synchronous release of the reset. |
| VREFp | A1 | - | Hi-Z | OUTPUT. Positive microphone reference voltage. |
| VREFm | A1 | - | - | Ground for CODEC AFE reference voltages and microphone. Must be connected to star-point of common ground. |
| LSRp, LSRn | A1 | - | Hi-Z | OUTPUT. Loudspeaker earpiece outputs positive and negative. |
| MICp, MICn | A2 | - | I | INPUT. Microphone inputs positive and negative. |

Table 1: Pin Description

| Pin Name | Type | Drive (mA) | Reset State | Description |
|--------------------------|------|------------|-------------|--|
| VBUF | A1 | - | Hi-Z | AFE Voltage reference 1.5 V output. Must be decoupled with a 100 nF capacitor |
| VDD | A1 | - | - | INPUT. CODEC supply voltage (1.8 V). |
| VDDIO | A1 | - | - | INPUT. Supply voltage for the CODEC digital I/Os (up to 3.45 V). |
| VSS | A1 | - | - | Digital core ground and digital I/O ground. |
| AVS | A1 | - | - | Ground for CODEC AFE main blocks. |
| Radio Transceiver | | | | |
| RFIOp | AIO | | | RF input/output. Impedance 50 Ω. |
| RFIOm | AIO | | | RF ground |
| Miscellaneous | | | | |
| RST | DI | | | INPUT. BLE reset signal (active HIGH). Must be connected to GND if not used. |
| VBAT_RF | AIO | | | Connect to VBAT3V on the PCB |
| VDCDC_RF | AIO | | | Connect to VDCDC on the PCB |
| VPP | AI | | | INPUT. This pin is used while OTP programming and testing. OTP programming: VPP = 6.7 V ± 0.1 V OTP Normal operation: leave VPP floating |
| Power Supply | | | | |
| VBAT3V | AIO | | | INPUT/OUTPUT. Battery connection. Used for a single coin cell battery (3 V). |
| VBAT1V | AI | | | INPUT. This pin must be connected to GND. |
| SWITCH | AIO | | | INPUT/OUTPUT. Connection for the external DC-DC converter inductor. |
| VDCDC | AO | | | Output of the DC-DC converter |
| GND | AIO | - | - | Ground |

Table 2: SPI Pin Assignment during CODEC Operation

| DA14482 Pin | CODEC Port | BLE Core Port |
|---------------|---------------|------------------|
| P2_1/SPI_EN | SPI_EN (I) | P2_1/SPI_EN (O) |
| Not available | SPI_CLK (I) | P3_1/SPI_CLK (O) |
| P0_4/SPI_DI | SPI_DI (I) | P0_4/SPI_DO (O) |
| Not available | SPI_DO (O) | P3_2/SPI_DI (I) |
| P0_5/16MCLK | CLK/16MHz (I) | P0_5/16MCLK (O) |

3 Ordering Information

Table 3: Ordering Information

| Part Number | Package | Size (mm) | Shipment Form | Pack Quantity |
|---------------|---------|-----------|---------------|-------------------|
| DA14582-01KA1 | QFN56 | 8 x 8 | Tray | 50 (Samples) |
| DA14582-01KA2 | QFN56 | 8 x 8 | Reel | 4000 (Production) |

NDA CONFIDENTIAL

4 System Overview

4.1 INTERNAL BLOCKS

The DA14582 contains the following blocks:

ARM Cortex M0 CPU with Wake-up Interrupt Controller (WIC). This processor provides 0.9 dMIPS/MHz and is used for assisting the Bluetooth low energy protocol implementation, providing processing power for calculations or data fetches required by the application and finally housekeeping, including controlling of the power scheme of the system.

BLE Core. This is the baseband hardware accelerator for the Bluetooth low energy protocol.

ROM. This is a 84 kB ROM containing the Bluetooth low energy protocol stack as well as the boot code sequence.

OTP. This is a 32 kB One-Time Programmable memory array, used to store the application code as well as Bluetooth low energy profiles. It also contains the system configuration and calibration data.

System SRAM. This is a 42 kB system SRAM (SysRAM) which is primarily used for mirroring the program code from the OTP when the system wakes/powers up. It also serves as Data RAM for intermediate variables and various data that the protocol requires. Optionally, it can be used as extra memory space for the BLE TX and RX data structures.

Retention RAMs. These 4 special low leakage SRAM cells (2 kB + 2 kB + 3 kB + 1 kB) are used to store various data of the Bluetooth low energy protocol as well as the system's global variables and processor stack when the system goes into Deep Sleep mode. Storage of this data ensures secure and quick configuration of the BLE Core after the system wakes up. Every cell can be powered on or off according to the application needs for retention area when in Deep Sleep mode.

UART and UART2. These asynchronous serial interfaces implement hardware flow control with FIFO depths of 16 bytes each.

SPI. This is the serial peripheral interface with master/slave capability and a FIFO of 2 16-bit words.

I2C. This is Master/Slave I2C interface used for sensors and/or host MCU communication. It comprises a 32 places 9-bits wide FIFO.

General purpose (GP) ADC. This is a 10-bits analog-to-digital converter with 4 external input channels.

Radio transceiver. This block implements the RF part of the Bluetooth low energy protocol.

Clock Generator. This block is responsible for the clocking of the system. It contains 2 XTAL oscillators: one running at 16 MHz (XTAL16M) which is used for the active mode of the system and one running at 32.768 kHz (XTAL32K) which is used for the sleep modes of the system. There are also three RC oscillators available: a 16 MHz and a 32 kHz oscillator (RC16M and RC32K) with low precision (> 500 ppm)

and an 10.5 kHz oscillator (RCX) with high precision (< 500 ppm). The RCX oscillator can be used as a sleep clock replacing the XTAL32K oscillator to further improve the power dissipation of the system while reducing the bill of materials of the system. The RC16M oscillator is used to provide a clock used for the mirroring of the OTP code into the SysRAM while the XTAL16M oscillator is settling directly after power/wake up.

Software Timer. This block contains a 16-bit general purpose timer (Timer0) with PWM capability as well as a 14-bits timer (Timer2) which controls 3 PWM signals with respect to frequency and duty cycle.

Wake-Up Timer. This is a timer for capturing external events and it can be used as a wake-up trigger based on a programmable number of external events on any of the GPIO ports.

Quadrature Decoder. This block decodes the pulse trains from a rotary encoder to provide the step and the direction of the movement of an external device. Three axes (X, Y, Z) are supported.

Keyboard Controller. This circuit enables the reading and debouncing of a programmable number of GPIOs and generates an interrupt upon a configurable action.

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

Power Management. A sophisticated power management circuit with a Buck/Boost DC-DC converter and several LDOs that can be turned on/off via software.

Note: The DA14582 does not support Boost mode.

CODEC. A bidirectional 14 bits Sigma-delta CODEC with fully differential analog front-end for microphones and loudspeakers down to 28 Ω is connected internally via SPI to the BLE core.

A more detailed description of each of the components of the DA14582 is presented in the following sections.

4.2 FUNCTIONAL MODES

The DA14582 is optimized for deeply embedded applications such as health monitoring, sports measuring, human interaction devices etc. Customers are able to develop and test their own applications. Upon completion of the development, the application code can be programmed into the OTP. In general, the system has three functional modes of operation:

A. Development Mode: During this phase application code is developed using the ARM Cortex-M0 SW environment. The compiled code is then downloaded into the System RAM or any Retention RAMs by means of SWD (JTAG) or any serial interface (e.g. UART). Address 0x00 is remapped to the physical memory that contains the code and the CPU is configured to reset and execute code from the remapped device. This mode is enabling application development, debugging and on-the-fly testing.

B. Normal Mode: After the application is ready and verified, the code can be burned into the OTP. When

the system boots/wakes up, the DMA of the OTP controller will automatically copy the program code from the OTP into the system RAM. Next, a SW reset or a jump to the System RAM occurs and code execution is started. Hence, in this mode, the system is autonomous, contains the required SW in OTP and is ready for integration into the final product.

C. Calibration Mode: Between Development and Normal mode, there is an intermediate stage where the chip needs to be calibrated with respect to two important features:

- Programming of the Bluetooth device address
- Programming of the trimming value for the external 16 MHz crystal.

This mode of operation applies to the final product and is performed by the customer. During this phase, certain fields in the OTP should be programmed as described in [section 4.3.1](#).

4.3 OTP MEMORY LAYOUT

The One Time Programmable memory has to be programmed according to a specific layout, which structures information to be easily accessible from the BootROM code as well as the actual application. An overview of the layout scheme is presented in the following figure:

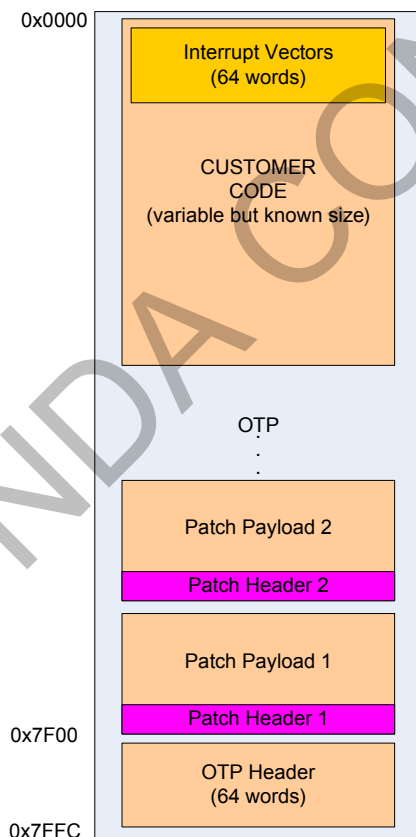


Figure 3: OTP Layout Scheme

The OTP memory comprises 8K of 32-bit words. The contents are described below:

- **Interrupt Vectors:** These are the vectors of the interrupt service routines **always** residing at address 0x0. This is part of the application (customer) code. The size of this vector list is 64 words.
- **Customer Code:** contains the application and the profiles that a customer has developed. The size is known and fixed before mass production and programming of the OTP.
- **Patch Area:** contains changes that have to be applied on the Customer Code. Each patch area comprises a Header and a Payload. Multiple patch areas are possible, however a single word must be left unprogrammed between the Customer Code area and the Patch Area. The patching mechanism is described in detail in Application Note AN-B-002.
- **OTP Header:** contains various information about the configuration of the system as well as Bluetooth low energy specific data.

4.3.1 OTP Header

The OTP header breakdown is presented in the following table:

Table 4: OTP Header

| Address | Description (single byte values apply to all 4 bytes) |
|------------------|--|
| 0x7FFC | SWD enable flag: 0x00 = JTAG is enabled 0xAA = JTAG is disabled |
| 0x7FF8 | OTP DMA length (number of 32-bit words) |
| 0x7FF4 | Reserved. Keep to 0x0 |
| 0x7FDC to 0x7FF0 | Customer Specific Field (6 32-bit words) |
| 0x7FD4 to 0x7FDB | Bluetooth Device Address (2 32-bit words). It is handled as a string of bytes. Leftmost number will be programmed at 0x7FD4 etc. |
| 0x7FD0 | Signature Algorithm Selector: 0x00 = None 0xAA = MD5 0x55 = SHA-1 0xFF = CRC32 |
| 0x7F94 to 0x7FCC | Signature of Customer Code (15 32-bit words) |
| 0x7F90 | Trim value for the VCO |
| 0x7F8C | Trim value for the XTAL16M oscillator |
| 0x7F88 | Trim value for the RC16M oscillator |
| 0x7F84 | Trim value for the BandGap |
| 0x7F80 | Trim value for the RFIO capacitance |
| 0x7F7C | Trim value for the LNA |

Table 4: OTP Header

| Address | Description (single byte values apply to all 4 bytes) |
|---------------------|---|
| 0x7F78 | Calibration Flags: Bit[31:16] - 0xA5A5 = at least 1 calibration was done 0x0000 = no calibration was done Bit[15:6] - Reserved Bit[5] - 1 = VCO trim value is valid Bit[4] - 1 = XTAL16M trim value is valid Bit[3] - 1 = RC16M trim value is valid Bit[2] - 1 = BandGap trim value is valid Bit[1] - 1 = RFIO trim value is valid Bit[0] - 1 = LNA trim value is valid |
| 0x7F74 | Sleep Clock Source Flag: 0x00 = External crystal (XTAL32K) 0xAA = Internal RCX oscillator (RCX) |
| 0x7F70 | Package Flag: 0x00 = WLCSP34 0xAA = QFN40 0x55 = QFN48 |
| 0x7F64 to 0x7F6C | Reserved |
| 0x7F10 to 0x7F60 | Customer Specific Field (21 32-bit words) |
| 0x7F0C | CRC for Trim and Calibration values |
| 0x7F08 | IQ Trim value |
| 0x7F04 | Application Programmed Flag #2 0xA5A51234 = Application is in OTP |
| 0x7F00 | Application Programmed Flag #1 0x1234A5A5 = Application is in OTP |

The first word (at address 0x7FFC) is a flag which defines whether SWD (JTAG) is mapped on pins or not. Default value is activating the SDW on the respective pins. The Length field (0x7FF8) specifies the number of 32-bit words to be copied to the SRAM.

The Bluetooth device address is stored at address 0x7FD4 (2 32-bit words).

The next field (0x7FD0) identifies the algorithm to be used for creating a signature on the OTP payload. The actual signature value is stored in the next fields (if a signature algorithm is selected). This is an optional feature to guarantee trusted OTP images.

The next memory locations contain the trim values for the XTAL16M and the RC16M oscillators. Since every crystal is different, an extra calibration step is required in the production line process, to identify the correct trimming value for the XTALs so that they provide the precision required by the Bluetooth low energy protocol. Since the crystal is an external component, this step has to be performed during the calibration phase.

A similar procedure is required for the trimming of the RC capacitance to keep the RC clock within a certain range of frequencies. However, this trimming is done

during production tests by Dialog Semiconductor.

Trimming values for the VCO, Bandgap reference, the RFIO capacitance and the LNA of the Radio are also stored in the OTP header. These values are generated and programmed during production testing by Dialog Semiconductor.

The Calibration Flags defines whether the chip has been already calibrated and if so, which trim values are valid. The 32 kHz Source Flag indicates to the application software whether an external 32 kHz crystal is used or not.

The Package Flag defines which package is used: the 34 balls WLCSP, the 40 pins QFN or the 48 pins QFN.

The IQ Trim value contains the value of respective radio configuration fields and is delivered at production testing.

Two more flags are indicating if the application code has indeed been programmed (burned) into the OTP. Both flags are read by the BootROM software designating that the system is now in Normal mode and not in Development mode, as explained in the next section.

4.4 SYSTEM START PROCEDURE

The actual start procedure consists of the following distinct stages that are sequentially combined to form the preferred system start sequence:

1. The Power/Wake-up sequence
2. The OTP mirroring sequence
3. The BootROM booting sequence

The Power/Wake-up sequence is a hardwired state machine that enables the LDOs and prepares the system's internal voltages.

The OTP mirroring is a hardwired state machine which instructs the copying of the OTP contents into the SysRAM even before the ARM CPU starts.

The BootROM code will only be executed when the system is powered up or a hardware reset occurs.

The relation between the aforementioned sequences is presented in the following figure:

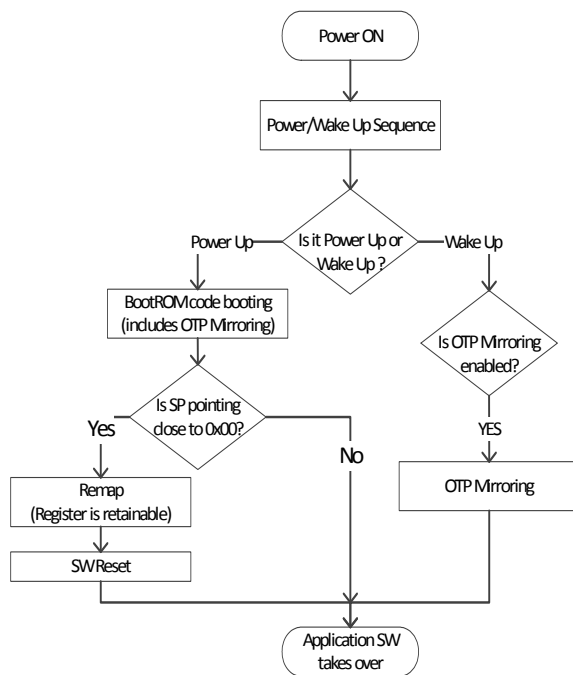


Figure 4: General Startup Flow

The system will enter the Power/Wake-Up sequence after cold power up or after waking up from one of the Sleep modes, which are described in [section 4.5.2](#). The distinction is done by means of a Flip-Flop which is set to '1' to indicate that the system has been just powered up (cold start). The whole process from power on up to the point that the system starts advertising (considering a typical application) takes around 145 ms.

4.4.1 Power/Wake-up Sequence

The Power/Wake-up sequence is a hardwired state machine that is activated every time the system is powered up (cold start) or woken up from one of the Sleep modes. This state machine uses information from internal analog comparators to reliably identify whether the system is set up in BUCK ($V_{BAT3V} > 2\text{ V}$) or BOOST ($V_{BAT1B} > 0.5\text{ V}$) mode automatically.

Initially, it checks whether the voltage at the VBAT1V pin is above the threshold indicating that the system is in BOOST mode and if so, it activates the DC-DC converter accordingly. If not, it continues enabling the Bandgap, the digital LDO and the OTP LDO to provide a stable 1.2 V for the core and 2.4 V for the OTP cell. Next, the XTAL16M oscillator is started while a second check overwrites the programming of the DC-DC converter in the correct mode.

Since LDOs as well as the XTAL16M oscillator take sometime to settle, the state machine is polling on signals indicating that the startup of these blocks has occurred successfully. However, if these signals are not set in a timely manner, the state machine ignores

the status of these blocks and continues with the power-up sequence. The timeout can be disabled by software via `SYS_CTRL_REG[TIMEOUT_DISABLE]`.

When no timeout occurs, the time required from the Power/Wake-up until the state that the chip is powered correctly and either the BootROM (in the case of a cold start) or the OTP mirroring (in the case of a programmed wake up) takes over, is ranging from 1.2 ms up to 1.5 ms depending on the supply voltage.

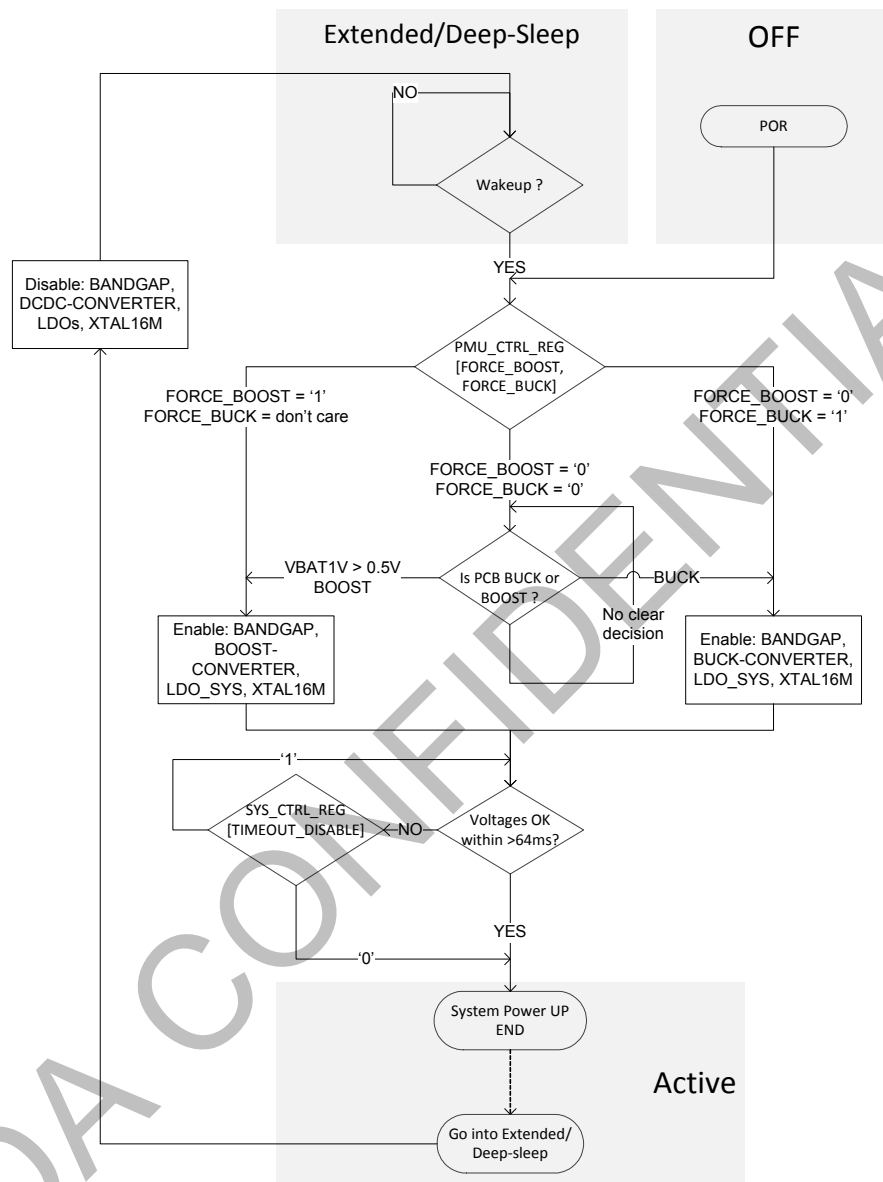


Figure 5: Power/Wake-Up Sequence

4.4.2 OTP Mirroring

This is one of the two branches of the decision regarding Power-up or Wake-up as illustrated in Figure 4. During the OTP mirroring process, the contents of the OTP memory are copied into the SysRAM, so that the ARM CPU can start executing code from there instead of the power hungry OTP memory. This task is programmable (via SYS_CTRL_REG[OTP_COPY]). In case of a Power Up, this task is performed by SW in the BootROM, i.e. the ARM CPU takes care of the mirroring of the application code into the SysRAM. However, in case of a Wake-up, the BootROM code is not executed and a small hardware state machine performs the mirroring as shown in the following figure.

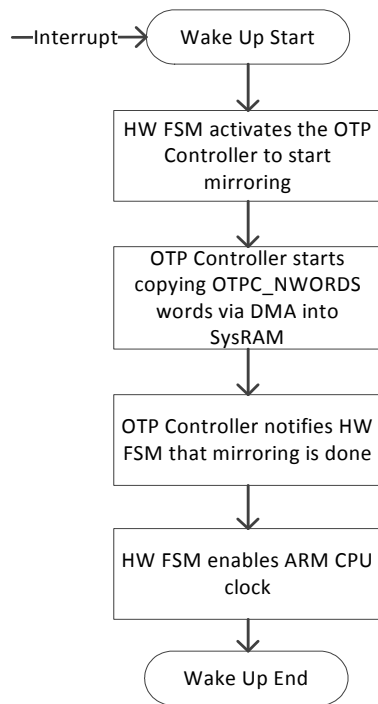


Figure 6: Wake-Up OTP Mirroring

The flow chart of [Figure 6](#) assumes that the OTP controller is aware of the number of words that need to be copied from the OTP memory. This value resides in the OTP Header (see [Table 4](#)) and at power up will be copied by the BootROM code (i.e. the ARM CPU) into the OTPC_NWORDS register. This is done only once, since the OTPC_NWORDS register retains its value even when the system goes into any of the Sleep modes. In this way, the number of words to be transferred from the OTP memory into the SysRAM by the OTP controller DMA engine is always available.

The typical time required for a full OTP mirroring (all 32 kB) using a trimmed RC16M clock is close to 1.2 ms.

4.4.3 BootROM sequence

The BootROM code identifies whether the chip is in Development mode or Normal mode by reading the “Application Programmed” flags from the OTP header (see [Table 4](#)). The OTP contains all zeros when it is not programmed.

If the predefined value is identified this ensures that the OTP is functional and that the application code has been programmed. However, if the predefined value is not identified, either the OTP is not programmed (all zeros) or the OTP memory is not operational (random data).

In the first case, the system enters Development mode where the application can be developed and values

can be calibrated. In the second case, the BootROM code recognizes the OTP to be malfunctioning due to power issues (e.g. battery life is ending and thus the OTP LDO cannot generate the required voltages) and continues to activate the peripherals so that the system is still usable and can be debugged.

The booting process of the DA14582 is presented in the following figure:

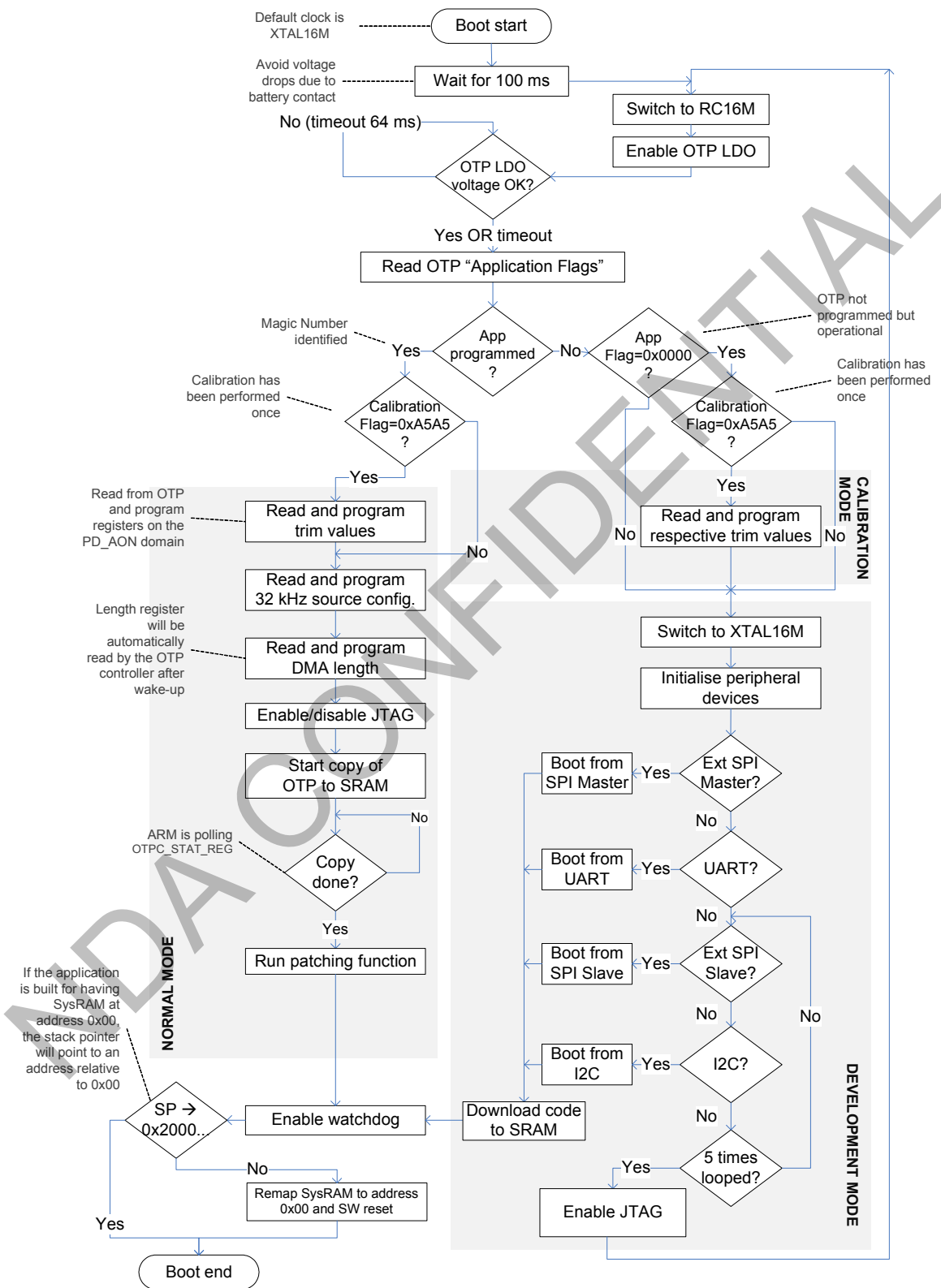


Figure 7: BootROM Sequence

When in **Development Mode**, the BootROM code initializes all peripheral devices from which the DA14582 might download code: UART, SPI (both Master and Slave) and I2C. The code searches for a valid response on various combinations of I/Os one after the other. The step sequence and GPIO mapping is presented in the following tables:

Table 5: Development Mode Peripheral Pin Mapping

| | Signal | Step A | Step B | Step C | Step D |
|------------|--------|--------|--------|--------|--------|
| SPI Master | SCK | P0_0 | P0_0 | | |
| | CS | P0_3 | P0_1 | | |
| | MISO | P0_6 | P0_2 | | |
| | MOSI | P0_5 | P0_3 | | |
| UART | TX | P0_0 | P0_2 | P0_4 | P0_6 |
| | RX | P0_1 | P0_3 | P0_5 | P0_7 |
| SPI Slave | SCK | P0_0 | | | |
| | CS | P0_3 | | | |
| | MOSI | P0_6 | | | |
| | MISO | P0_5 | | | |
| I2C | SCL | P0_0 | P0_2 | P0_4 | P0_6 |
| | SDA | P0_1 | P0_3 | P0_5 | P0_7 |

Table 6: Development Mode Peripheral Search Sequence

| Sequence | Action |
|----------|-------------------|
| 0 | SPI Master Step A |
| 1 | SPI Master Step B |
| 2 | UART Step A |
| 3 | UART Step B |
| 4 | UART Step C |
| 5 | UART Step D |
| 6 | SPI Slave Step A |
| 7 | I2C Step A |
| 8 | I2C Step B |
| 9 | I2C Step C |
| 10 | I2C Step D |

At each step, the BootROM code sends a certain character to the peripheral controller waiting for an answer. If no answer is received within a certain amount of time, a timeout instructs the code to continue to the next step. If a response is detected, a specific protocol is executed and communication between the DA14582 and the external device is established. The protocol and the timeout values are described in detail in Application Note AN-B-001.

When in **Normal Mode**, the ARM CPU programs OTP header configuration information into actual registers, some of them retaining this value even if the system enters one of the Sleep modes. Next, the actual Application Code is mirrored into the System RAM and the patching function is executed to ensure that SW updates are applied. The patching mechanism is described in detail in Application Note AN-B-002. Finally, the Remap register (SYS_CTRL_REG[REMAP_ADR0]) is programmed to ensure that address zero is now pointing to the System RAM.

A SW reset as a last step, instructs the ARM CPU to reboot running code from the SRAM.

4.5 POWER SUPPLY CONFIGURATION

4.5.1 Power Domains

The DA14582 comprises several different power domains, that are controlled by power switching elements, thus eliminating leakage currents by totally powering them down. The partitioning of the DA14582's resources with respect to the various power domains is presented in the following table:

Table 7: Power Domains

| Power Domain | Description |
|--------------|---|
| PD_AON | Always ON: This power line connects to all the resources that must be powered constantly: the ARM/WIC, the LLP/Timer, the Retention SRAM, the PMU/CRG, the Capture Timer, the Quadrature Decoder, the padding and various registers required for the Wake Up sequence. |
| PD_SYS | System: This power line connects to all the resources that should be powered only when the ARM M0 is running: the AHB bus, the OTP cell and controllers, the ROM, the System RAM the Watchdog, the SW Timer and the GPIO port multiplexing. |
| PD_PER | Peripherals: This power line connects to the peripherals that can be switched off after completing their operation: the UARTs, the SPI, the I2C the Keyboard controller and the ADC. |
| PD_DBG | Debug: Powers the debug part of the ARM Cortex-M0 processor. |
| PD_RAD | Radio: This is the power domain that includes the digital part of the Radio (DPHY) and the BLE Core. The power management of the Radio (RF) subsystem is controlled via several dedicated LDOs. |

Table 7: Power Domains

| Power Domain | Description |
|--------------|--|
| PD_RR | Retention_RAM: This is a separate power line that only controls the 2 kB Retention SRAM. If this memory cell is not needed, it should always be OFF. |
| PD_RR2 | Retention_RAM2: This is a separate power line that only controls the 3 kB Retention SRAM. If this memory cell is not needed, it should always be OFF |
| PD_RR3 | Retention_RAM3: This is a separate power line that only controls the second 2 kB Retention SRAM. If this memory cell is not needed, it should always be OFF |
| PD_RR4 | Retention_RAM4: This is a separate power line that only controls the 1 kB Retention SRAM. If this memory cell is not needed, it should always be OFF |
| PD_SR | System_RAM: This is a separate power line that only powers the 42 kB SysRAM. This power line keeps the SysRAM's contents retained but not accessible. |

4.5.2 Power Modes

There are four different power modes in the DA14582:

1. **Active Mode:** System is active and operates at full speed.
2. **Sleep Mode:** No power gating has been programmed, the ARM CPU is idle, waiting for an interrupt. PD_SYS is on. PD_PER and PED_RAD depending on the programmed enabled value.
3. **Extended Sleep Mode:** All power domains are

off except for the PD_AON, the programmed PD_RRx and the PD_SR. Since the SysRAM retains its data, not OTP mirroring is required upon waking up the system.

4. **Deep Sleep mode:** All power domains are off except for the PD_AON and the programmed PD_RRx. This mode dissipates the minimum leakage power. However, since the SysRAM has not retained its data, an OTP mirror action is required upon waking up the system.

The first two power modes do not include any special power gating or manipulation of power domains. The Extended Sleep and Deep Sleep modes are activated in almost the same way with a small difference. The entering into the Extended Sleep and Deep Sleep modes is summarized in the following table:

Table 8: Activating Power Modes

| Power Mode | Activation Steps |
|----------------|---|
| Extended Sleep | SYS_CTRL_REG[RET_SYSRAM] = 1; SCB->SCR = 1<<2; enables the SLEEPDEEP bit on the System Control Register of the ARM CPU |
| Deep Sleep | SCB->SCR = 1<<2; enables the SLEEPDEEP bit on the System Control Register of the ARM CPU |

However, the above Power modes which involve power gating do not turn off all power domains automatically. Certain power domains need to be powered off prior to activating the Extended Sleep or Deep Sleep modes. The following table summarizes the effect of the Extended/Deep Sleep mode activation with regards to specific power domains.

Table 9: Power Domain Manipulation when Activating Extended Sleep or Deep Sleep Mode

| Power Mode | PD_SYS | PD_PER | PD_DBG | PD_RAD | PD_RRx | PD_SR | Analog |
|----------------|-------------------|--------------------|-------------------|--------------------|--------------------|-------------------|-------------------|
| Extended Sleep | Automatically OFF | Must be programmed | Automatically OFF | Must be programmed | Must be programmed | Automatically ON | Automatically OFF |
| Deep Sleep | Automatically OFF | Must be programmed | Automatically OFF | Must be programmed | Must be programmed | Automatically OFF | Automatically OFF |

The "Analog" column in Table 9 refers to the Bandgap, the DC-DC converter, the XTAL16M oscillator, the RC oscillators, the ADC and the respective LDOs. These blocks are not part of a single power domain but the HW will turn them off upon activation of either the Extended Sleep or the Deep Sleep mode.

The use of the Extended Sleep vs. the Deep Sleep mode depends heavily on the Bluetooth low energy application and its specific parameters such as the required connection interval, the duty cycle and the

amount of data transmitted or received per connection. Based on these parameters a time threshold for the connection interval can be calculated, above which Deep Sleep mode performs better than Extended Sleep mode. This is elaborated upon in Application Note AN-B-008.

4.5.3 Retention Registers

A number of the registers in the DA14582 needs to retain their values when the system enters one of the

Sleep modes described in the previous section. These registers and their power domains are described in the following table.

Table 10: Retention Registers And Power Domains

| Power Domain | Address Range | | Retention Register | Notes |
|--------------|---------------|-------------|------------------------------|--|
| | Start | End | | |
| PD_RAD | 0x4000.0000 | 0x4000.7FFF | BLE_CNTL2_REG | Retained bit fields are: BLE_RSSI_SEL SW_RPL_SPI BB_ONLY RADIO_ONLY BLE_CLK_SEL RADIO_PWRDN_ALLOW MON_LP_CLK DIAGPORT_REVERSE DIAGPORT_SEL EMACCERRMSK EMACCERRSTAT |
| | | | DEBUG_REG | Retained bit fields are: DEBUG_FREEZE_EN |
| PD_SYS | 0x4000.8000 | 0x4000.83FF | OTPC_NWORDS_REG | Contains the number of words to be copied from the OTP memory into the Sys-RAM. |
| | 0x5000.3000 | 0x5000.3FFF | GP_CONTROL_REG | Retained bit fields are: EM_MAP |
| PD_AON | 0x5000.0000 | 0x5000.0FFF | All registers are retainable | |
| PD_PER | 0x5000.1000 | 0x5000.1FFF | No registers are retainable | |

5 Reset

The DA14582 comprises an RST pad which is active high. It contains an RC filter for spikes suppression with 400 k Ω and 2.8 pF for the resistor and the capacitor respectively. It also contains a 25 k Ω pull-down resistor. This pad should be connected to ground if not needed by the application. The response is illustrated in the following figure which displays the voltage (V) on the vertical axis and the time (μ s) on the horizontal axis:

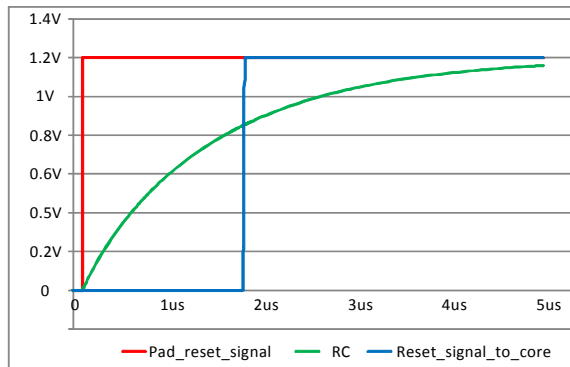


Figure 8: RST Pad Latency

The typical latency of the RST pad is in the range of 2 μ s.

5.1 POR, HW AND SW RESET

The Power On Reset (POR) signal is generated internally and will release the system's flip flops as soon as the VDD voltage crosses the minimum threshold value. There are two main reset signals in the DA14582, both active high. The HW reset which is basically triggered by the RST pad and the SW reset which is triggered by writing the SYS_CTRL_REG[SW_RESET] bit.

The HW reset can also be automatically activated upon waking up of the system from an Extended Sleep or Deep Sleep mode by programming bit PMU_CTRL_REG[RESET_ON_WAKEUP]. The HW reset will basically run the cold start-up sequence and the BootROM code will be executed.

The SW reset is the logical OR of a signal from the ARM CPU (triggered by writing SCB->AIRCR = 0x05FA0004) and the SYS_CTRL_REG[SW_RESET] bit. This is mainly used to reboot the system after the base address has been remapped.

Certain registers are reset by POR only or by POR and the HW reset signal, but not by the SW reset. These registers are listed in the following table:

Table 11: Reset Signals and Registers

| Reset by POR Only | Reset by POR or HW Reset | Reset by POR, HW or SW Reset |
|--|------------------------------|-------------------------------|
| OTPC_NWORDS_REG | CLK_FREQ_TRIM_REG | The rest of the Register File |
| BANDGAP_REG | CLK_RADIO_REG | |
| RF registers containing the trimming values for the VCO, the LNA and the I/O capacitance | All RF calibration registers | |
| | BLE_CNTL2_REG | |
| | CLK_CTRL_REG | |
| | PMU_CTRL_REG | |
| | SYS_CTRL_REG | |
| | CLK_32K_REG_I | |
| | CLK_16M_REG_I | |
| | CLK_RCX32K_REG | |
| | TRIM_CTRL_REG | |
| DEBUG_REG[DEBUGS_FREEZE_EN] | | |
| GP_CONTROL_REG[EM_MAP] | | |

6 ARM Cortex-M0

The Cortex-M0 processor is a 32-bit Reduced Instruction Set Computing (RISC) processor with a von Neumann architecture (single bus interface). It uses an instruction set called Thumb, which was first supported in the ARM7TDMI processor; however, several newer instructions from the ARMv6 architecture and a few instructions from the Thumb-2 technology are also included. Thumb-2 technology extended the previous Thumb instruction set to allow all operations to be carried out in one CPU state. The instruction set in Thumb-2 includes both 16-bit and 32-bit instructions; most instructions generated by the C compiler use the 16-bit instructions, and the 32-bit instructions are used when the 16-bit version cannot carry out the required

operations. This results in high code density and avoids the overhead of switching between two instruction sets.

In total, the Cortex-M0 processor supports only 56 base instructions, although some instructions can have more than one form. Although the instruction set is small, the Cortex-M0 processor is highly capable because the Thumb instruction set is highly optimized.

Academically, the Cortex-M0 processor is classified as load-store architecture, as it has separate instructions for reading and writing to memory, and instructions for arithmetic or logical operations that use registers.

A simplified block diagram of the Cortex-M0 is shown in Figure 9.

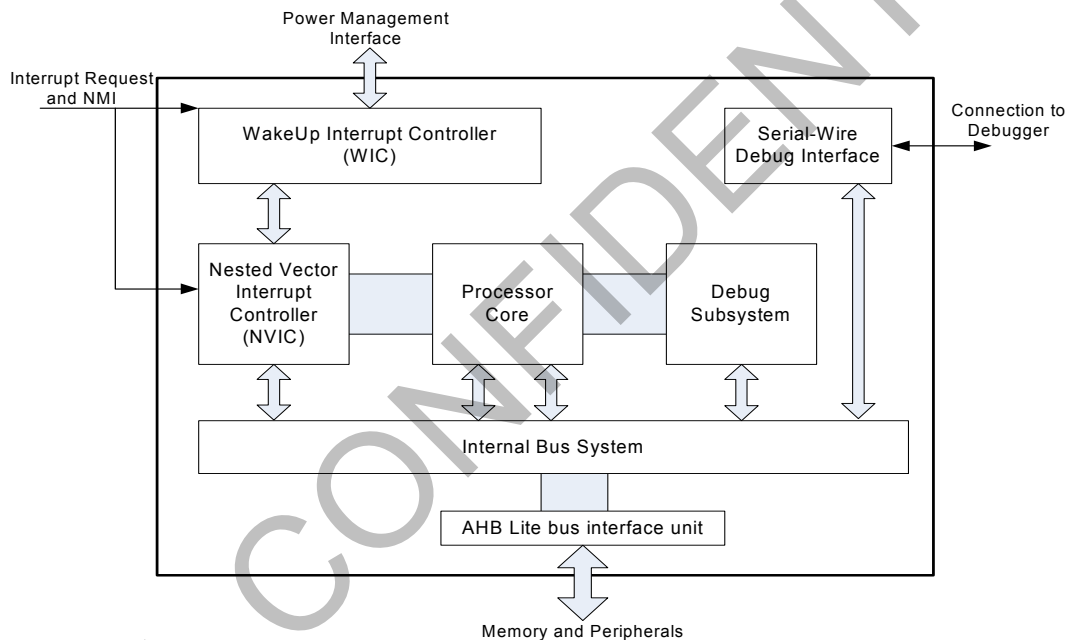


Figure 9: ARM Cortex-M0 Block Diagram

Features

- Thumb instruction set. Highly efficient, high code density and able to execute all Thumb instructions from the ARM7TDMI processor.
- High performance. Up to 0.9 DMIPS/MHz (Dhrystone 2.1) with fast multiplier.
- Built-in Nested Vectored Interrupt Controller (NVIC). This makes interrupt configuration and coding of exception handlers easy. When an interrupt request is taken, the corresponding interrupt handler is executed automatically without the need to determine the exception vector in software.
- Interrupts can have four different programmable priority levels. The NVIC automatically handles nested interrupts.
- The design is configured to respond to exceptions (e.g. interrupts) as soon as possible (minimum 16 clock cycles).
- Non maskable interrupt (NMI) input for safety critical systems.
- Easy to use and C friendly. There are only two modes (Thread mode and Handler mode). The whole application, including exception handlers, can be written in C without any assembler.
- Built-in System Tick timer for OS support. A 24-bit timer with a dedicated exception type is included in the architecture, which the OS can use as a tick timer or as a general timer in other applications without an OS.
- SuperVisor Call (SVC) instruction with a dedicated SVC exception and PendSV (Pendable SuperVisor service) to support various operations in an embedded OS.

- Architecturally defined sleep modes and instructions to enter sleep. The sleep features allow power consumption to be reduced dramatically. Defining sleep modes as an architectural feature makes porting of software easier because sleep is entered by a specific instruction rather than implementation defined control registers.
- Fault handling exception to catch various sources of errors in the system.
- Support for 24 interrupts.
- Little endian memory support.
- Wake up Interrupt Controller (WIC) to allow the processor to be powered down during sleep, while still allowing interrupt sources to wake up the system.
- Halt mode debug. Allows the processor activity to stop completely so that register values can be accessed and modified. No overhead in code size and stack memory size.
- CoreSight technology. Allows memories and peripherals to be accessed from the debugger without halting the processor.
- Supports Serial Wire Debug (SWD) connections. The serial wire debug protocol can handle the same debug features as the JTAG, but it only requires two wires and is already supported by a number of debug solutions from various tools vendors.
- Four (4) hardware breakpoints and two (2) watch points.
- Breakpoint instruction support for an unlimited number of software breakpoints.
- Programmer's model similar to the ARM7TDMI processor. Most existing Thumb code for the ARM7TDMI processor can be reused. This also makes it easy for ARM7TDMI users, as there is no need to learn a new instruction set.

6.1 INTERRUPTS

This section lists all 24 interrupt lines, except the NMI interrupt, and describes their source and functionality. The overview of the interrupts is illustrated in the following table:

Table 12: Interrupt List

| IRQ Number (Inherent Priority) | IRQ Name | Description |
|--------------------------------|---------------------|---|
| 0 | BLE_WAKEUP_LP_IRQn | Wake-up from Low Power (Deep Sleep) interrupt from BLE. |
| 1 | BLE_FINETGTIM_IRQn | Fine Target Timer interrupt generated when Fine Target timer expired. Timer resolution is 625 μ s base time reference. |
| 2 | BLE_GROSSTGTIM_IRQn | Gross Target Timer interrupt generated when Gross Target timer expired. Timer resolution is 16 times 625 μ s base time reference. |
| 3 | BLE_CSCNT_IRQn | 625 μ s base time reference interrupt, available in active modes. |
| 4 | BLE_SLP_IRQn | End of Sleep mode interrupt. |
| 5 | BLE_ERROR_IRQn | Error interrupt, generated when undesired behavior or bad programming occurs in the BLE Core. |
| 6 | BLE_RX_IRQn | Receipt interrupt at the end of each received packets. |
| 7 | BLE_EVENT_IRQn | End of Advertising / Scanning / Connection events interrupt. |
| 8 | SWTIM_IRQn | Software Timer interrupt. |
| 9 | WKUP_QUADEC_IRQn | Combines the Wake up Capture Timer interrupt, the GPIO interrupt and the QuadDecoder interrupt |
| 10 | BLE_RF_DIAG_IRQn | Baseband or Radio Diagnostics Interrupt. Triggered by internal events of the Radio or Baseband selected by the BLE_RF_DIAGIRQ_REG. For Debug purposes only. |
| 11 | BLE_CRYPT_IRQn | Encryption / Decryption interrupt, generated either when AES and/or CCM processing is finished. |
| 12 | UART_IRQn | UART interrupt. |
| 13 | UART2_IRQn | UART2 interrupt. |
| 14 | I2C_IRQn | I2C interrupt. |
| 15 | SPI_IRQn | SPI interrupt. |
| 16 | ADC_IRQn | Analog-Digital Converter interrupt. |
| 17 | KEYBRD_IRQn | Keyboard interrupt. |

Table 12: Interrupt List

| IRQ Number (Inherent Priority) | IRQ Name | Description |
|--------------------------------|------------|----------------------------------|
| 18 | RFCAL_IRQn | RF Calibration Interrupt. |
| 19 | GPIO0_IRQn | GPIO interrupt through debounce. |
| 20 | GPIO1_IRQn | GPIO interrupt through debounce. |
| 21 | GPIO2_IRQn | GPIO interrupt through debounce. |
| 22 | GPIO3_IRQn | GPIO interrupt through debounce. |
| 23 | GPIO4_IRQn | GPIO interrupt through debounce. |

Interrupt priorities are programmable by the ARM Cortex-M0. The lower the priority number, the higher the priority level. The priority level is stored in a byte-wide register, which is set to 0x0 at reset. Interrupts with the same priority level follow a fixed priority order using the interrupt number listed in Table 12 (lower interrupt number has higher priority level).

To access the Cortex-M0 NVIC registers, CMSIS functions can be used. The input parameter IRQn of the CMSIS NVIC access functions is the IRQ number. This can be the IRQ number or (more convenient) the corresponding IRQ name listed in Table 12. The corresponding interrupt handler name in the vector table for IRQ#15 is e.g. SPI_Handler. For more information on the ARM Cortex-M0 interrupts and the corresponding CMSIS functions, see a.o. section 4.2 Nested Vectored Interrupt Controller on pag. 4-3 in the Cortex™-M0 User Guide Reference Material.

The Watchdog interrupt is connected to the NMI input of the processor.

6.2 SYSTEM TIMER (SYSTICK)

The Cortex-M0 System Timer (SysTick) can be configured for using 2 different clocks. The SysTick Control & Status (STCSR) register specifies which clock should be used by the counter.

STCSR[CLKSOURCE]=0; use the (fixed) external reference clock STCLKEN of 1 MHz.

STCSR[CLKSOURCE]=1; use the (HCLK_DIV dependent) processor clock SCLK (e.g. 2, 4, 8 or 16 MHz).

Table 13: ARM Documents List

| | Document Title | ARM Document Number |
|---|--|---|
| 1 | Cortex™-M0 User Guide Reference Material | ARM DUI 0467B (available on the ARM website) |
| 2 | Cortex™-M0 r0p0 Technical Reference Manual | ARM DDI 0432C (available on the ARM website) |
| 3 | ARMv6-M Architecture Reference Manual | ARM DDI 0419C (can be downloaded by registered ARM customers) |

The default SysTick Timer configuration will be using the (fixed) external reference clock STCLKEN (STCSR[CLKSOURCE]=0). When necessary, higher clock frequencies can be used with STCSR[CLKSOURCE]=1 but the software should take the HCLK_DIV dependent core clock SCLK into account w.r.t. the timing.

6.3 WAKE-UP INTERRUPT CONTROLLER

The Wake-up Interrupt Controller (WIC) is a peripheral that can detect an interrupt and wake the processor from deep sleep mode. The WIC is enabled only when the DEEPSLEEP bit in the SCR is set to 1 (see System Control Register on page 4-16 of the Cortex-M0 User Guide Reference Material).

The WIC is not programmable, and does not have any registers or user interface. It operates entirely from hardware signals. When the WIC is enabled and the processor enters deep sleep mode, the power management unit in the system can power down most of the Cortex-M0 processor. This has the side effect of stopping the SysTick timer. When the WIC receives an interrupt, it takes a number of clock cycles to wake-up the processor and restore its state, before it can process the interrupt. This means interrupt latency is increased in Deep Sleep mode.

6.4 REFERENCE

For more information on the ARM Cortex-M0, see the ARM documents listed in Table 13.

7 AMBA Bus Overview

The DA14582 is based on an AMBA 2.0 AHB and APB components. The AHB is an AMBA Lite version which requires a single master on the system, i.e. in the DA14582 the ARM CPU.

The APB interface implements 3 different decoding slaves which are grouped according to the power domain structure of the chip.

The AMBA bus organization is presented in the following figure:

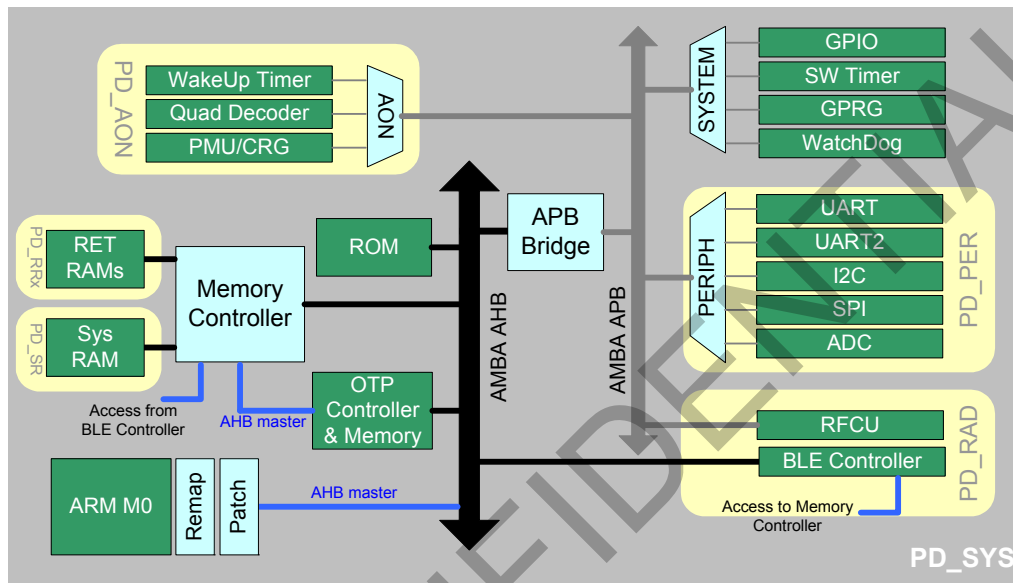


Figure 10: AMBA Bus Architecture and Power Domains

Since the DA14582 consists of several different power domains that are digitally controlled and can be shut down completely, various slave resources especially on the APB bus are grouped together to reduce signal isolation requirements.

On the AHB Lite bus, the CPU is the master while OTP, BLE Core, Memory and ROM controllers are the slaves. Furthermore, on the APB subsystem, the Always On power domain consists of the following blocks:

- Wake-up Timer
- Quadrature Decoder
- Power Management Unit / Clock Reset Generator
- Retention RAMs (not mapped on APB but accessible through the Memory Controller)

The APB peripherals power domain comprises:

- UART and UART2
- SPI
- I2C
- GP ADC

The radio power domain contains both AHB and APB peripherals i.e. the Radio Frequency Control Unit (RFCU) and the Bluetooth Low Energy controller.

The rest of the system is mapped on the system power domain and contains APB blocks (the GPIO multiplexing, the Software Timer, the General Purpose Registers and the WatchDog timer) as well as AHB blocks (OTP, Memory and ROM controller).

8 Patch Block

The patch block provides a multiplexer on the databus of the ROM Controller able to modify (patch) the contents of the data read from ROM. This feature provides a repair mechanism of a ROM image.

Features

- Direct patching on ROM databus
- Up to 8x 32 bit patch entries
- Enable/disable patch entries
- Priority scheme allows off-loads software
- Patch data can be stored internally in OTP or externally in EEPROM or QSPI

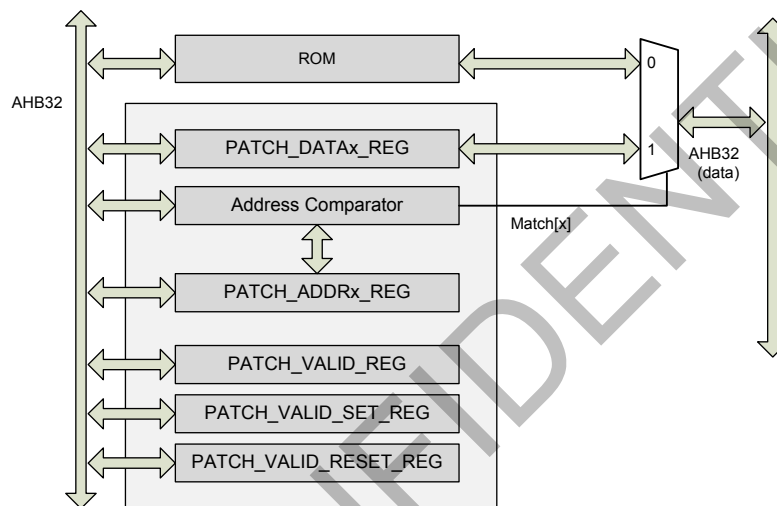


Figure 11: Patch Block

The patch block's registers are accessible via the AHB bus. The patch unit resides in the System Power domain. It is clocked by the system bus clock (HCLK) and must be initialized after each power-up.

There are 8 patching entries of two 32-bit words each. A patch entry consists of a 32-bit address register PATCH_ADDRx_REG and one 32-bit data register PATCH_DATAx_REG, implemented as sequential register pairs. The programmer must program the address register first followed by the data register. As soon as both registers are written, the corresponding bit PATCH_VALID_REG[x] is automatically set.

Patch Address Comparators

The patch unit has an octal address comparator to compare AHB address bus with PATCH_ADDRx_REG

With enabled patch entries, upon an address match, the data bus value is replaced with the corresponding PATCH_DATAx_REG. Matching uses priorities starting with highest priority at the last entry (PATCH_ADDR7_REG) and lowest priority at the first entry (PATCH_ADDR0_REG). This priority scheme helps SW to add more patches without having to check the previous entries with a lower entry number.

Disabling/enabling Patch Entries

Patch entries can be disabled by writing a '1' to the write-only register PATCH_VALID_RESET_REG[x]. Disabled entries can be re-enabled by writing a '1' to write-only register PATCH_VALID_SET_REG[x]

To introduce new patches, the SW must check the available patch entries, evaluate the priorities (e.g. if it will use two patch entries for the same 32-bit address) and program the corresponding register pair.

9 Memory Map

Table 14: Memory Map

| Address | Description |
|--------------------------|--|
| 0x00000000 0x00014FFF | Boot/BLE ROM Contains 6 kB of Boot ROM code and 78 kB of Bluetooth low energy protocol related code |
| 0x00015000 0x0001FFFF | RESERVED |
| 0x00020000 0x00034FFF | Boot/BLE ROM Contains 6 kB of Boot ROM code and 78 kB of Bluetooth low energy protocol related code |
| 0x00035000 0x0003FFFF | RESERVED |
| 0x00040000 0x00047FFF | OTP Contains the OTP cell actual memory space |
| 0x00048000 0x0007FFFF | RESERVED |
| 0x00080000 0x000807FF | Retention RAM (Note 2) 2 kB of retained memory space. |
| 0x00080800 0x000813FF | Retention RAM2 (Note 2) 3 kB of retained memory space. |
| 0x00081400 0x00081BFF | Retention RAM3 (Note 2) 2 kB of retained memory space. |
| 0x00081C00 0x00081FFF | Retention RAM4 (Note 2) 1 kB of retained memory space. |
| 0x00082000 0x1FFFFFFF | RESERVED |
| 0x20000000 0x2000A7FF | System RAM (Note 2) 42 kB. Contains OTP code, data for the application |
| 0x2000A800 0x3FFFFFFF | RESERVED |
| 0x40000000 0x40007FFF | AHB/BLE-Regs Contains the control registers of the Bluetooth low energy Link Layer Processor |
| 0x40008000 0x400083FF | AHB/OTP-Regs Contains the control registers of the OTP Subsystem |
| 0x40008400 0x400087FF | AHB/Patch-Regs Contains the registers for the HW patching |
| 0x40008800 0x4FFFFFFF | RESERVED |
| 0x50000000 0x500000FF | APB/PMU-CRG Contains the control registers of the Power Management Unit and the Clock Generator |
| 0x50000100 0x500001FF | APB/wake-up Contains an event capture timers that can wake up the system |
| 0x50000200 0x500002FF | APB/Quadrature Decoder Contains Logic that implements a step counter for X and Y axis from a rotary encoder |
| 0x50000300 0x50000FFF | RESERVED |
| 0x50001000 0x500010FF | APB/UART Contains the control registers of the UART |
| 0x50001100 0x500011FF | APB/UART2 Contains the control registers of the UART2 |
| 0x50001200 0x500012FF | APB/SPI Contains the control registers of the SPI interface |

Table 14: Memory Map

| Address | Description |
|--------------------------|---|
| 0x50001300 0x500013FF | APB/I2C Contains the control registers of the I2C interface |
| 0x50001400 0x500014FF | APB/Kbrd Contains the registers of the Keyboard controller |
| 0x50001500 0x500015FF | APB/ADC Contains the registers of the 4 channels ADC |
| 0x50001600 0x500016FF | APB/AnaMisc Contains registers for various analog blocks |
| 0x50001700 0x50001FFF | RESERVED |
| 0x50002000 0x50002FFF | APB/Radio Contains the control registers of the Bluetooth low energy Radio |
| 0x50003000 0x500030FF | APB/Ports Contains the mode and direction registers of the GPIOs |
| 0x50003100 0x500031FF | APB/Watchdog Contains the control registers of the Watchdog timer |
| 0x50003200 0x500032FF | APB/Version Contains the version/revision of the chip |
| 0x50003300 0x500033FF | APB/Gen Purpose Contains general purpose control registers |
| 0x50003400 0x500034FF | APB/Timer Contains the control registers of the SW Timer |
| 0x50003500 0xDFFFFFFF | RESERVED |
| 0xE0000000 0xE0FFFFFF | Internal Private Bus Contains various registers of the ARM Cortex-M0 |

Note 1: 1 kB = 1024 bytes. 1 Mbit = 1024*1024 bits. 1 GB = 1024*1024*1024 bytes.

Note 2: For Exchange memory configurations please refer to [Figure 71](#).

10 Memory Controller

The Memory Controller implements smart multiplexing and arbitration of three different masters accessing 2 memory resources, namely the System RAM and the Retention RAM cells. The masters that require access to these memories are:

- The CPU over the system AHB bus. This master has full access to both resources.
- The OTP Controller. This master can access the System RAM only through a dedicated AHB bus.

- The BLE Controller. This master can also access both resources through its proprietary memory bus.

The block diagram is presented in [Figure 12](#).

Features

- Meets all timing constraints for any access to the physical cells.
- Transparently interfaces AHB busses to memory signalling
- Fixed arbitration algorithm with time sharing.

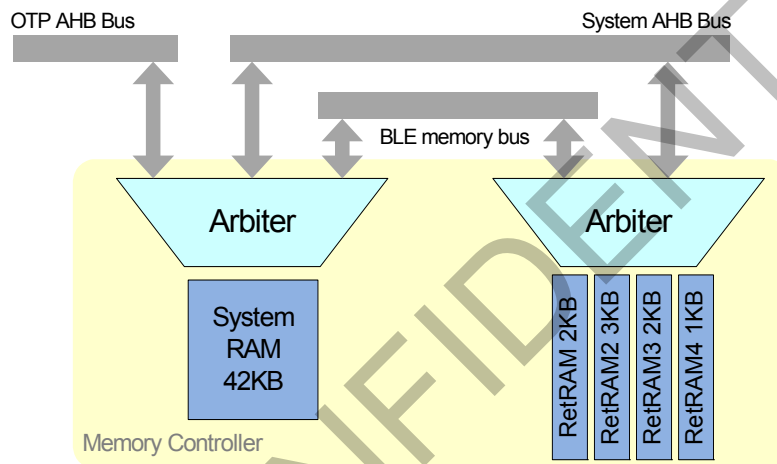


Figure 12: Memory Controller Block Diagram

10.1 ARBITRATION

The arbitration is a mixture of highest priority and a fair use policy. If more masters request access, time division is employed. This to make sure none of the buses is able to stall another for a long period. The OTP and BLE accesses are handled as very critical and therefore they have highest priority. In general BLE and OTP accesses are mutually exclusive i.e. no BLE accesses occur while OTP requests. OTP has highest priority and will only occur during the OTP mirroring i.e. the system has not yet started executing application code. When OTP mirroring is done, BLE gets highest priority and only one every 3 cycles the CPU is allowed to get access. If none of the interfaces requests access, the IDLE or power down state is selected.

11 Clock Generation

11.1 CRYSTAL OSCILLATORS

The Digital Controlled Xtal Oscillator (DXCO) is a Pierce configured type of oscillator designed for low power consumption and high stability. There are two such crystal oscillators in the system, one at 16 MHz(XTAL16M) and a second at 32.768 kHz (XTAL32K). The 32.768 kHz oscillator has no trimming capabilities and is used as the clock of the Extended/ Deep Sleep modes. The 16 MHz oscillator can be trimmed.

The principle schematic of the two oscillators is shown in Figure 13 below. No external components to the DA14582 are required other than the crystal itself. If the crystal has a case connection, it is advised to connect the case to ground.

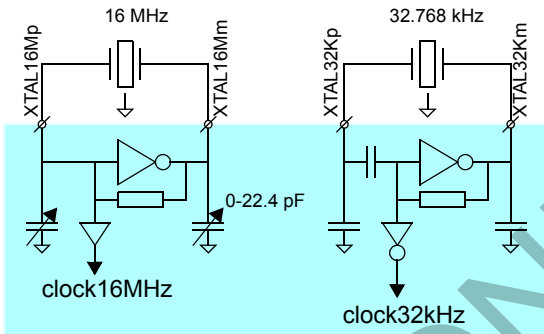


Figure 13: Crystal Oscillator Circuits

11.1.1 Frequency Control (16 MHz Crystal)

Register CLK_FREQ_TRIM_REG controls the trimming of the 16 MHz crystal oscillator. The frequency is trimmed by two on-chip variable capacitor banks. Both capacitor banks are controlled by the same register.

The capacitance of both variable capacitor banks varies from minimum to maximum value in 2048 equal steps. With CLK_FREQ_TRIM_REG = 0x000 the maximum capacitance and thus the minimum frequency is selected. With CLK_FREQ_TRIM_REG = 0x7FF the minimum capacitance and thus the maximum frequency is selected.

The eight least significant bits of CLK_FREQ_TRIM_REG directly control eight binary weighted capacitors, as shown in Figure 14. The three most significant bits are decoded according to Table 15. Each of the seven outputs of the decoder controls a capacitor (value is 256 times the value of the smallest capacitor).

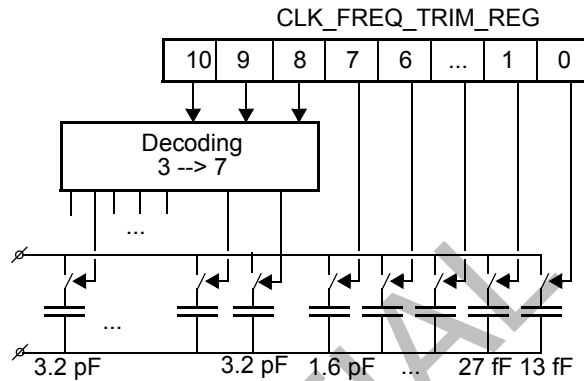


Figure 14: Frequency Trimming

Table 15: CLK_FREQ_TRIM_REG Decoding 3 --> 7

| Input[2:0] | | | Output[6:0] | | | | | | |
|------------|---|---|-------------|---|---|---|---|---|---|
| 2 | 1 | 0 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Trimming might cause phase jumps in the oscillator signal. To reduce these phase jumps the user should only change one switch at a time (this especially applies to the seven larger capacitors). Use bits 10 to 8 for coarse adjustment and always increment or decrement this value by 1. Wait approximately 200 μs to allow the adjustment to settle.

Bits 7...0 are used for fine adjustment.

As an example, the recommended way to change the frequency trim register from 0x7FF to 0x100 is first to decrement the value of the three most significant bits by 1 at a time, and then change the least significant bits until the desired frequency is reached: 0x7FF --> 0x6FF --> 0x5FF --> 0x4FF --> 0x3FF --> 0x2FF --> 0x1FF --> 0x100.

11.1.2 Automated Trimming Mechanism

There is provision in the DA14582 for automating the actual trimming of the 16 MHz crystal oscillator. This is a special hardware block that realizes the XTAL trimming in a single step.

Two programmable parameters are involved in the trimming of the XTAL16M oscillator:

- TRIM_CTRL_REG[TRIM_TIME]
This field controls the time that elapses between the enabling of the XTAL and the application of the trim value specified by CLK_FREQ_TRIM_REG. The TRIM_TIME value counts cycles of 250 μ s. As soon as TRIM_TIME*250 μ s time has passed, the SYS_STAT_REG[XTAL16_TRIM_READY] bit becomes '1'. This is the point that the trim value will be applied at the oscillator. If TRIM_TIME = 0000 the trim value is applied immediately when XTAL16M starts up, to reduce time.
- TRIM_CTRL_REG[SETTLE_TIME]
This field controls a counter that counts 250 μ s intervals starting when XTAL16_TRIM_READY has been asserted. Upon completion of the SETTLE_TIME periods, the XTAL_SETTLED bit in the SYS_STAT_REG becomes '1', indicating that XTAL16M clock is settled and trimmed and may be safely used as the system clock.

An initial manual frequency measuring action is required to store the wanted value in the OTP header (See "OTP Memory Layout" on page 11).

11.2 RC OSCILLATORS

There are 3 RC oscillators in the DA14582: one providing 16 MHz (RC16M), one providing 32 kHz (RC32K) and one providing a frequency in the range of 10.5 kHz (RCX).

The 16 MHz RC oscillator is powered by the Digital LDO i.e. the VDD = 1.2 V which is available for the core logic during Active or Sleep mode. The output clock is slightly slower than 16 MHz and is used to clock the system during the OTP mirroring procedure after wake-up, while the XTAL16M oscillator is settling. It is important to keep the 16 MHz RC oscillator at a frequency equal or below 16 MHz to guarantee correct behavior of the digital circuits. This is why the reset value of CLK_16M_REG[RC16M_TRIM] is equal to the minimum value.

The simple RC oscillator (RC32K) operates on VDD = 1.2 V and provides 32 kHz. The enhanced RC oscillator (RCX) provides a stable 18 kHz and operates only if the external voltage is > 1.8 V (i.e. either on pin VBAT1V in BOOST mode or on pin VBAT3V in BUCK mode). The main usage of the RC32K oscillator is for internal clocking during startup.

The RCX oscillator can be used to replace a 32.768 kHz crystal, since it has a precision of < 500 ppm, while its output frequency is quite stable over temperature. Although a different frequency is used, the accuracy is good enough to ensure that the correct number of slots (625 μ s) is counted for the Sleep time.

11.2.1 Frequency Calibration

The output frequency of the 32 kHz crystal oscillator and the three RC-oscillators can be measured relative to the 16 MHz crystal oscillator using the on-chip reference counter.

The measurement procedure is as follows:

1. REF_CNT_VAL = N (the higher N, the more accurate and longer the calibration will be)
2. CLK_REF_SEL_REG = 0x0006 (XTAL32K) or CLK_REF_SEL_REG = 0x0005 (RC16M) or CLK_REF_SEL_REG = 0x0004 (RC32K) or CLK_REF_SEL_REG = 0x0007 (RCX)
3. Wait until CLK_REF_SEL_REG[REF_CAL_START] = 0
4. Read CLK_REF_VAL_H_REG and CLK_REF_VAL_L_REG = M (32-bits values)
5. Frequency = (N/M) * 16 MHz

In the case of using the RCX as a sleep clock, the frequency calibration should be implemented on each active time of a connection interval to guarantee correct operation.

11.3 SYSTEM CLOCK GENERATION

The DA14582 clock generation overview is presented in the following figure. External clock sources of the device are pins XTAL16Mp/XTAL16Mm and pins

XTAL32Kp/XTAL32Km for the 16 MHz and 32.768 kHz oscillators respectively.

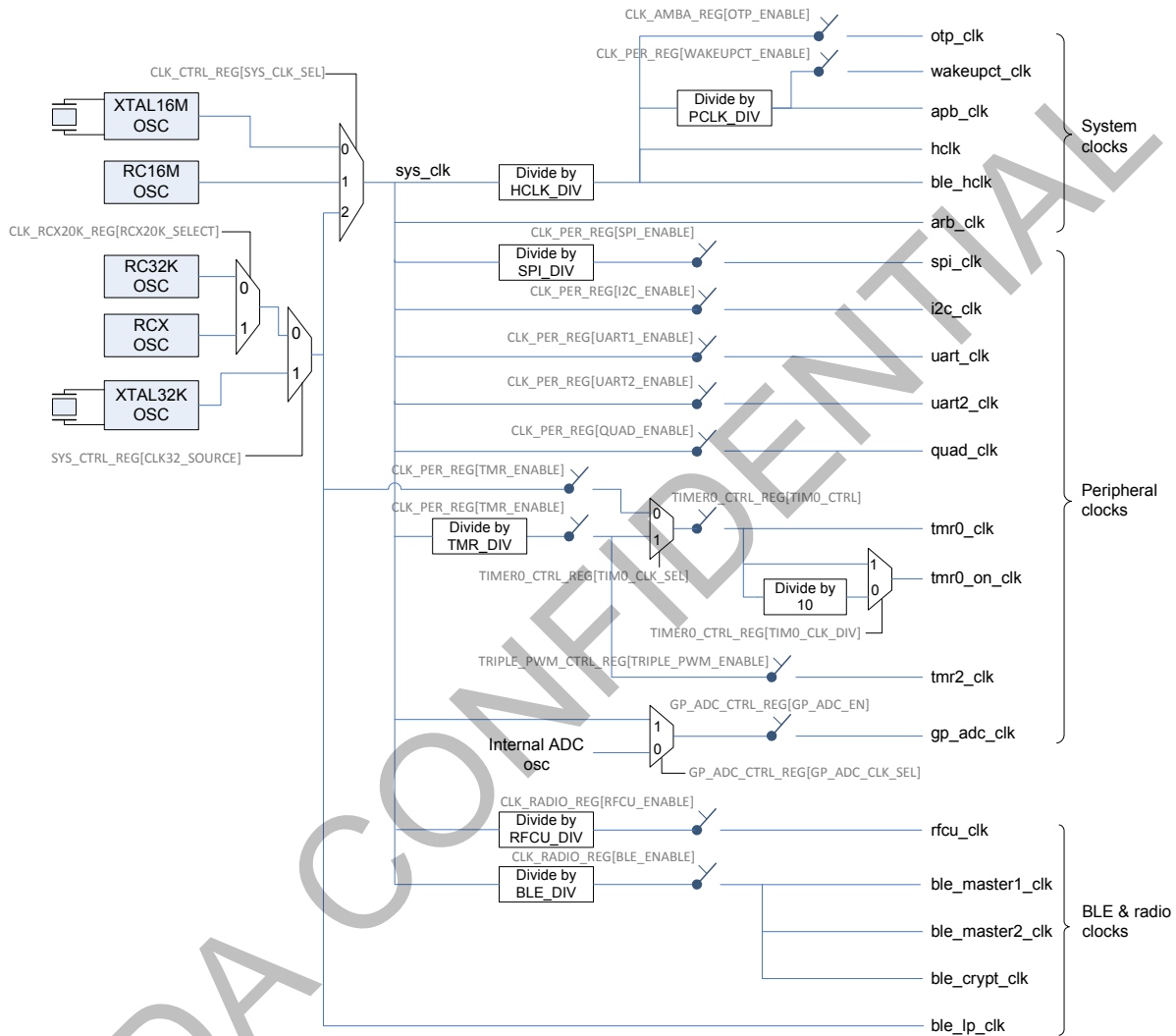


Figure 15: Clock Generation Block Diagram

Each of the clocks is generated by either a divider or a clock gate. Both dividers and clock gates are able to start/stop the clock of the module they drive. The control of the peripheral clocks is provided by CLK_PER_REG.

The clock names depicted in Figure 15 are explained in the following table:

Table 16: Generated Clocks Description

| Clock Name | Description |
|--------------|--|
| wakeupct_clk | This is the clock for the wake-up capture timer. |

Table 16: Generated Clocks Description

| Clock Name | Description |
|------------|--|
| apb_clk | AMBA APB interface clock |
| otp_clk | OTP controller clock. This is also used for the OTP macro cell and should not be higher than 16 MHz |
| hclk | AMBA AHB interface clock |
| ble_hclk | AMBA AHB clock for the BLE core |
| spi_clk | Clock for the SPI controller. This clock is further divided by 2, 4, 8 or 14 as defined by SPI_CTRL_REG[SPI_CLK] |

Table 16: Generated Clocks Description

| Clock Name | Description |
|-----------------------|--|
| i2c_clk | Clock for the I2C controller. This clock is further divided to provide 100 kHz or 400 kHz as defined by I2C_CON_REG[I2C_SPEED] |
| uart_clk | Clock for the UART |
| uart2_clk | Clock for the UART2 |
| quad_clk | Clock for the quadrature decoders |
| rfcu_clk | Clock for the RF control unit of the Radio |
| tmr0_clk, tmr2_clk | Timer0/2 clocks |
| tmr0_on_clk | Timer0 ON counter clock |
| gp_adc_clk | General Purpose ADC conversion clock |
| ble_crypt_clk | Clock for the Crypto block of the BLE core |
| ble_master1_clk | Internal clock for the BLE core |
| ble_master2_clk | Internal clock for the BLE core |
| arb_clk | Clock for the memory controller arbiter |
| ble_lp_clk | BLE core low power clock |

11.4 GENERAL CLOCK CONSTRAINTS

There are certain constraints on various clocks regarding their frequency relations or the effectiveness. This section summarizes these rules:

- Minimum hclk clock should be 8 MHz. This is also the clock of the ARM CPU and ensures the required MIPS for the Bluetooth low energy protocol handling.
- hclk should always be greater or equal to the ble_*_clks. This is required for the proper operation of the protocol. For example, hclk at 16 MHz and BLE clocks at 8 MHz is an acceptable combination but not the other way around.
- When the SPI block is set to Slave mode, then the spi_clk (as depicted in [Figure 15](#)) should be at least 4x the clock frequency provided by the external SPI Master. This is required for proper sampling of the data line.
- The UART clock is derived from the system clock as follows: baud rate = (system clock DIV 16) / divisor. The division by 16 is hard coded and hence, the baud rate is generated by the divisor on the 1 MHz input. This results in a certain error percentage with respect to the well defined baud rates as presented in the following table:

Table 17: Actual UART Baud Rates

| Target Baud Rate (kBd) | Divisor Value | Actual Baud Rate (kBd) | Error (%) |
|------------------------|---------------|------------------------|-----------|
| 115.2 | 9 | 111.1 | 3.54 |
| 57.6 | 17 | 58.82 | 2.12 |
| 38.4 | 26 | 38.46 | 0.16 |
| 28.8 | 35 | 28.57 | 0.79 |
| 19.2 | 52 | 19.23 | 0.16 |
| 9.6 | 104 | 9.61 | 0.16 |

If there is a hard requirement for having an average baud rate deviation of no more than 5 % on both ends of the UART connection, a baud rate with an error of maximum 2.5 % should be selected from the table above.

12 OTP Controller

The OTP Controller realizes all functions of the OTP macro cell in an automated and transparent way. The controller facilitates all data transfers (reading and programming), while implementing the required OTP test modes in hardware. The block diagram is presented in Figure 16.

Features

- Implements all timing constraints for any access to the physical memory cells.
- Transparent random address access to the OTP memory cells via the AHB slave memory interface.
- Embedded DMA engine for fast mirroring of the OTP contents into the System RAM.
- Embedded DMA supports reading in bursts of 4 32-bit words
- Standby state consumes minimum power
- Hardwired handshaking with the PMU to realize the mirroring procedure
- Emulation of the mirroring procedure for the Development mode.

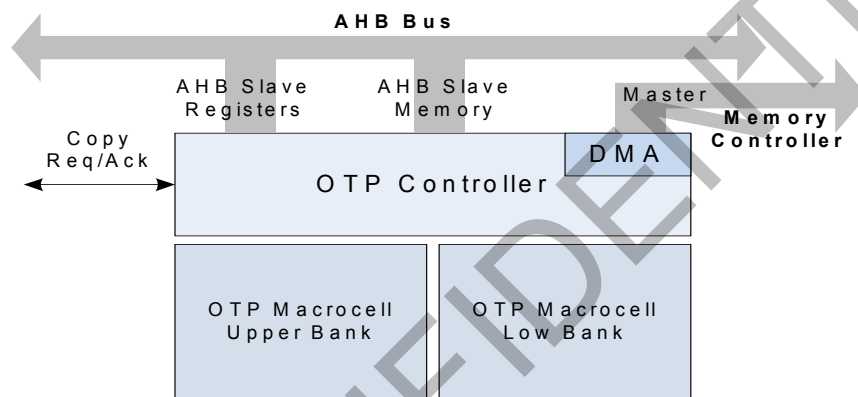


Figure 16: OTP Controller Block Diagram

12.1 OPERATING MODES

There are two different functional modes of operation for reading and programming respectively: manual (MREAD, MPROG) and automatic (AREAD, APROG). The OTP operating mode is programmable at `OTPC_MODE_REG[OTPC_MODE_MODE]`.

The **MREAD** mode enables the use of the memory slave interface. By activating this mode the contents of the macro cell are transparently mapped onto the specific AHB slave address space. This mode can be used for execution of software in place (XIP).

The **AREAD** mode provides the ability for reading data from the macro cell in bursts, without the use of the slave interface. This mode is used for copying large data blocks from the macro cell, as in the case of the OTP mirroring into the System RAM.

The **MPROG** mode implements the single step of the program/verify sequence. In this mode, when the programming of a bit fails, the controller does not automatically try to reprogram the bit. Hence, reprogramming should be triggered by software. The value and the address of the bit are defined through a configuration register (`OTPC_PCTRL_REG`), for both macro cell banks. The programming sequence can be enabled or disabled for each of the memory banks. The result of the program/verify is stored in register `OTPC_STAT_REG`. There is a status bit for each one of the two memory banks.

The **APROG** mode gives the ability for programming large data blocks into the macro cell. The programming is an automated procedure, during which it is only necessary to feed the controller with the required data. Data blocks can be fetched in two ways:

- Via the AHB master interface, i.e. the DMA
- Via the AHB memory interface

In the latter case, data are pumped into the OTP controller through a register, which acts as a port providing access to a FIFO.

12.2 AHB MASTER INTERFACE

The AHB master interface is controlled by a DMA engine with an internal FIFO of 8 32-bit words. The DMA engine supports AHB reads and writes. The AHB address where memory access should begin, is programmed into the DMA engine at `OTPC_AHBADR_REG[OTPC_AHBADR]`. The number of 32-bit words (minus 1) of a transfer must be specified in `OTPC_NWORDS_REG[OTPC_NWORDS]`.

The DMA engine internally supports the following burst types:

- Four words incremental burst
- Unspecified incremental burst of 2 or 3 words
- Single word access

13 I2C Interface

The I2C interface is a programmable control bus that provides support for the communications link between Integrated Circuits in a system. It is a simple two-wire bus with a software-defined protocol for system control, which is used in temperature sensors and voltage level translators to EEPROMs, general-purpose I/O, A/D and D/A converters.

Features

- Two-wire I2C serial interface consists of a serial data line (SDA) and a serial clock (SCL)
- Two speeds are supported:
- Standard mode (0 to 100 kbit/s)
- Fast mode (<= 400 kbit/s)
- Clock synchronization
- 32 deep transmit/receive FIFOs
- Master transmit, Master receive operation
- 7 or 10-bit addressing
- 7 or 10-bit combined format transfers
- Bulk transmit mode
- Default slave address of 0x055
- Interrupt or polled-mode operation
- Handles Bit and Byte waiting at both bus speeds
- Programmable SDA hold time

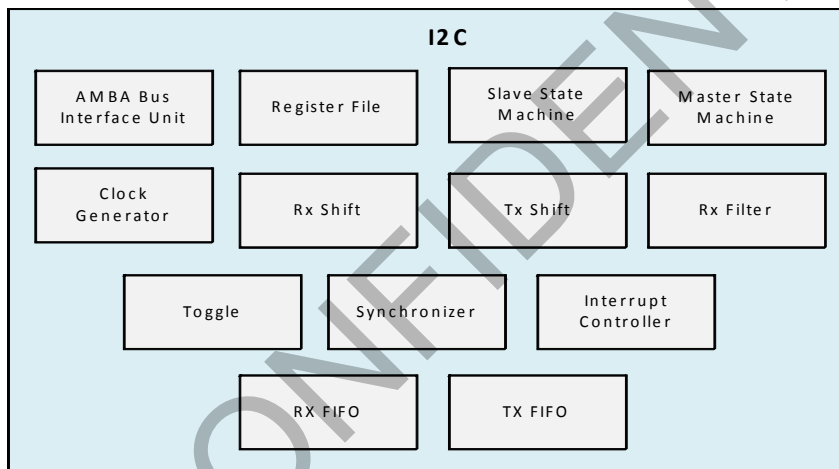


Figure 17: I2C Controller Block Diagram

The I2C Controller block diagram is shown in [Figure 17](#). It contains the following sub-blocks:

- AMBA Bus Interface Unit. Interfacing via the APB interface to access the register file.
- Register File. Contains configuration registers and is the interface with software.
- Master State Machine. Generates the I2C protocol for the master transfers.
- Clock Generator. Calculates the required timing to do the following:
 - Generate the SCL clock when configured as a master
 - Check for bus idle
 - Generate a START and a STOP
 - Setup the data and hold the data
- Rx Shift. Takes data into the design and extracts it in byte format.
- Tx Shift. Presents data supplied by CPU for transfer on the I2C bus.
- Rx Filter. Detects the events in the bus; for example, start, stop and arbitration lost.
- Toggle. Generates pulses on both sides and toggles to transfer signals across clock domains.
- Synchronizer. Transfers signals from one clock domain to another.
- Interrupt Controller. Generates the raw interrupt and interrupt flags, allowing them to be set and cleared.
- RX FIFO/TX. Holds the RX FIFO and TX FIFO register banks and controllers, along with their status levels.

13.1 I2C BUS TERMS

The following terms relate to how the role of the I2C device and how it interacts with other I2C devices on the bus.

- Transmitter. the device that sends data to the bus. A transmitter can either be a device that initiates the data transmission to the bus (a master-transmitter) or responds to a request from the master to send data to the bus (a slave-transmitter).

- Receiver. The device that receives data from the bus. A receiver can either be a device that receives data on its own request (a master-receiver) or in response to a request from the master (a slave-receiver).
- Master. The component that initializes a transfer (START command), generates the clock (SCL) signal and terminates the transfer (STOP command). A master can be either a transmitter or a receiver.
- Slave. The device addressed by the master. A slave can be either receiver or transmitter.

These concepts are illustrated in [Figure 18](#)

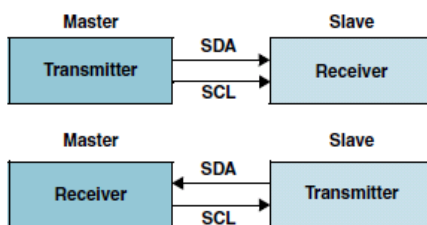


Figure 18: Master/Slave and Transmitter/Receiver Relationships

- Multi-master. The ability for more than one master to co-exist on the bus at the same time without collision or data loss.
- Arbitration. The predefined procedure that authorizes only one master at a time to take control of the bus. For more information about this behavior, refer to Multiple Master Arbitration chapter
- Synchronization. The predefined procedure that synchronizes the clock signals provided by two or more masters. For more information about this feature, refer to Clock Synchronization chapter
- SDA. Data signal line (Serial Data)
- SCL. Clock signal line (Serial CLock)

13.1.1 Bus Transfer Terms

The following terms are specific to data transfers that occur to/from the I2C bus.

- START (RESTART). data transfer begins with a

START or RESTART condition. The level of the SDA data line changes from high to low, while the SCL clock line remains high. When this occurs, the bus becomes busy.

- STOP. data transfer is terminated by a STOP condition. This occurs when the level on the SDA data line passes from the low state to the high state, while the SCL clock line remains high. When the data transfer has been terminated, the bus is free or idle once again. The bus stays busy if a RESTART is generated instead of a STOP condition.

Note 3: START and RESTART conditions are functionally identical.

13.2 I2C BEHAVIOR

The I2C can only be controlled via software to be an I2C master only, communicating with other I2C slaves;

The master is responsible for generating the clock and controlling the transfer of data. The slave is responsible for either transmitting or receiving data to/from the master. The acknowledgement of data is sent by the device that is receiving data, which can be either a master or a slave. As mentioned previously, the I2C protocol also allows multiple masters to reside on the I2C bus and uses an arbitration procedure to determine bus ownership.

Each slave has a unique address that is determined by the system designer. When a master wants to communicate with a slave, the master transmits a START/RESTART condition that is then followed by the slave's address and a control bit (R/W) to determine if the master wants to transmit data or receive data from the slave. The slave then sends an acknowledge (ACK) pulse after the address.

If the master (master-transmitter) is writing to the slave (slave-receiver), the receiver gets one byte of data. This transaction continues until the master terminates the transmission with a STOP condition. If the master is reading from a slave (master-receiver), the slave transmits (slave-transmitter) a byte of data to the master, and the master then acknowledges the transaction with the ACK pulse. This transaction continues until the master terminates the transmission by not acknowledging (NACK) the transaction after the last byte is received, and then the master issues a STOP condition or addresses another slave after issuing a RESTART condition. This behavior is illustrated in [Figure 19](#).

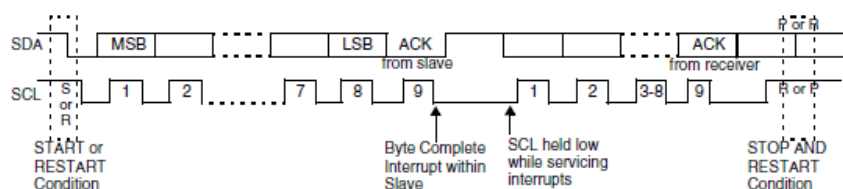


Figure 19: Data Transfer on the I2C Bus

The I2C is a synchronous serial interface. The SDA line is a bidirectional signal and changes only while the SCL line is low, except for STOP, START, and RESTART conditions. The output drivers are open-drain or open-collector to perform wire-AND functions on the bus. The maximum number of devices on the bus is limited by only the maximum capacitance specification of 400 pF. Data is transmitted in byte packages.

13.2.1 START and STOP Generation

When operating as an I2C master, putting data into the transmit FIFO causes the I2C Controller to generate a START condition on the I2C bus. Allowing the transmit FIFO to empty causes the I2C Controller to generate a STOP condition on the I2C bus.

When operating as a slave, the I2C Controller does not generate START and STOP conditions, as per the protocol. However, if a read request is made to the I2C Controller, it holds the SCL line low until read data has been supplied to it. This stalls the I2C bus until read data is provided to the slave I2C Controller, or the I2C Controller slave is disabled by writing a 0 to I2C_ENABLE.

13.2.2 Combined Formats

The I2C Controller supports mixed read and write combined format transactions in both 7-bit and 10-bit addressing modes.

The I2C Controller does not support mixed address and mixed address format, that is, a 7-bit address transaction followed by a 10-bit address transaction or vice versa, combined format transactions.

To initiate combined format transfers, I2C_CON.I2C_RESTART_EN should be set to 1. With this value set and operating as a master, when the I2C Controller completes an I2C transfer, it checks the transmit FIFO and executes the next transfer. If the direction of this transfer differs from the previous transfer, the combined format is used to issue the transfer. If the transmit FIFO is empty when the current I2C transfer completes, a STOP is issued and the next transfer is issued following a START condition.

13.3 I2C PROTOCOLS

The I2C Controller has the following protocols:

- START and STOP Conditions
- Addressing Slave Protocol
- Transmitting and Receiving Protocol
- START BYTE Transfer Protocol

13.3.1 START and STOP Conditions

When the bus is idle, both the SCL and SDA signals are pulled high through external pull-up resistors on the bus. When the master wants to start a transmission on the bus, the master issues a START condition. This is defined to be a high-to-low transition of the SDA signal while SCL is 1. When the master wants to terminate the transmission, the master issues a STOP condition. This is defined to be a low-to-high transition of the SDA line while SCL is 1. Figure 20 shows the timing of the START and STOP conditions. When data is being transmitted on the bus, the SDA line must be stable when SCL is 1.

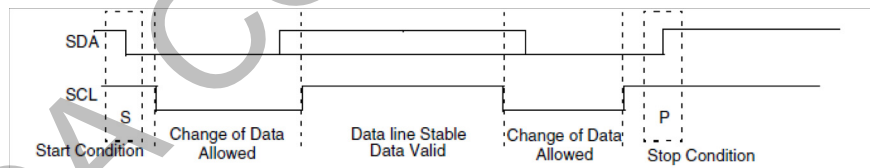


Figure 20: START and STOP Conditions

Note 4: The signal transitions for the START/STOP conditions, as depicted in Figure 19, reflect those observed at the output signals of the Master driving the I2C bus. Care should be taken when observing the SDA/SCL signals at the input signals of the Slave(s), because unequal line delays may result in an incorrect SDA/SCL timing relationship.

13.3.2 Addressing Slave Protocol

There are two address formats: the 7-bit address format and the 10-bit address format.

7-bit Address Format

During the 7-bit address format, the first seven bits (bits 7:1) of the first byte set the slave address and the LSB bit (bit 0) is the R/W bit as shown in Figure 21. When bit 0 (R/W) is set to 0, the master writes to the slave. When bit 0 (R/W) is set to 1, the master reads from the slave.

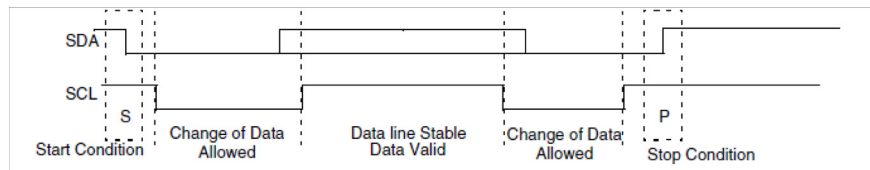


Figure 21: 7-Bit Address Format

10-Bit Address Format

During 10-bit addressing, two bytes are transferred to set the 10-bit address. The transfer of the first byte contains the following bit definition. The first five bits (bits 7:3) notify the slaves that this is a 10-bit transfer followed by the next two bits (bits 2:1), which set the

slaves address bits 9:8, and the LSB bit (bit 0) is the R/W bit. The second byte transferred sets bits 7:0 of the slave address. Figure 22 shows the 10-bit address format, and Table 18 defines the special purpose and reserved first byte addresses.

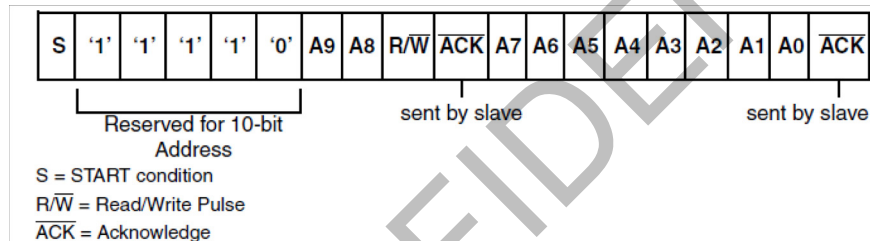


Figure 22: 10-Bit Address Format

Table 18: I2C Definition of Bits in First Byte

| Slave Address | R/W Bits | Description |
|---------------|----------|---|
| 0000 000 | 0 | General Call Address. I2C Controller places the data in the receive buffer and issues a General Call interrupt. |
| 0000 000 | 1 | START byte. For more details, refer to “START BYTE Transfer Protocol” 0000 |
| 0000 001 | X | CBUS address. I2C Controller ignores these accesses |
| 0000 010 | X | Reserved |
| 0000 011 | X | Reserved |
| 0000 1XX | X | High-speed master code (for more information, refer to “Multiple Master Arbitration”) |
| 1111 1XX | X | Reserved |
| 1111 0XX | X | 10-bit slave addressing |

I2C Controller does not restrict you from using these reserved addresses. However, if you use these reserved addresses, you may run into incompatibilities with other I2C components.

13.3.3 Transmitting and Receiving Protocols

The master can initiate data transmission and reception to/from the bus, acting as either a master-transmitter or master-receiver. A slave responds to requests

from the master to either transmit data or receive data to/from the bus, acting as either a slave-transmitter or slave-receiver, respectively.

Master-Transmitter and Slave-Receiver

All data is transmitted in byte format, with no limit on the number of bytes transferred per data transfer. After the master sends the address and R/W bit or the master transmits a byte of data to the slave, the slave-receiver must respond with the acknowledge signal

(ACK). When a slave-receiver does not respond with an ACK pulse, the master aborts the transfer by issuing a STOP condition. The slave must leave the SDA line high so that the master can abort the transfer.

If the master-transmitter is transmitting data as shown in Figure 23, then the slave-receiver responds to the master-transmitter with an acknowledge pulse after every byte of data is received.

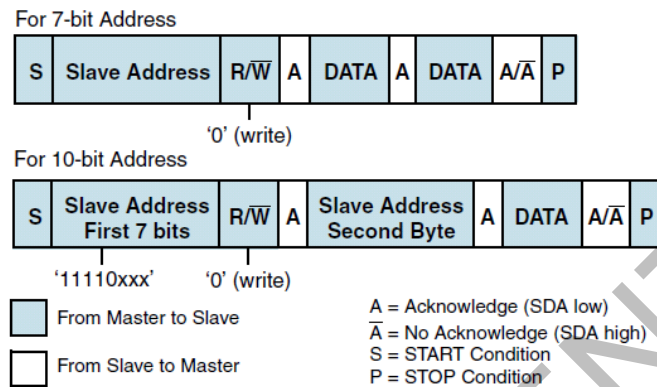


Figure 23: Master-Transmitter Protocol

Master-Receiver and Slave-Transmitter

If the master is receiving data as shown in Figure 24 then the master responds to the slave-transmitter with an acknowledge pulse after a byte of data has been received, except for the last byte. This is the way the master-receiver notifies the slave-transmitter that this is the last byte. The slave-transmitter relinquishes the SDA line after detecting the No Acknowledge (NACK) so that the master can issue a STOP condition.

When a master does not want to relinquish the bus with a STOP condition, the master can issue a RESTART condition. This is identical to a START condition except it occurs after the ACK pulse. The master can then communicate with the same slave or a different slave.

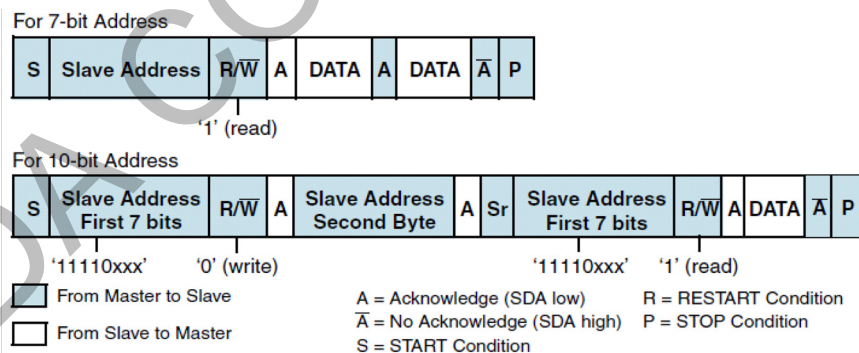


Figure 24: Master-Receiver Protocol

START BYTE Transfer Protocol

The START BYTE transfer protocol is set up for systems that do not have an on-board dedicated I2C hardware module. When the I2C Controller is addressed as a slave, it always samples the I2C bus at the highest speed supported so that it never requires a START BYTE transfer. However, when I2C Controller is a master, it supports the generation of START BYTE transfers at the beginning of every transfer in case a slave

device requires it. This protocol consists of seven zeros being transmitted followed by a 1, as illustrated in Figure 25. This allows the processor that is polling the bus to under-sample the address phase until 0 is detected. Once the microcontroller detects a 0, it switches from the under sampling rate to the correct rate of the master.

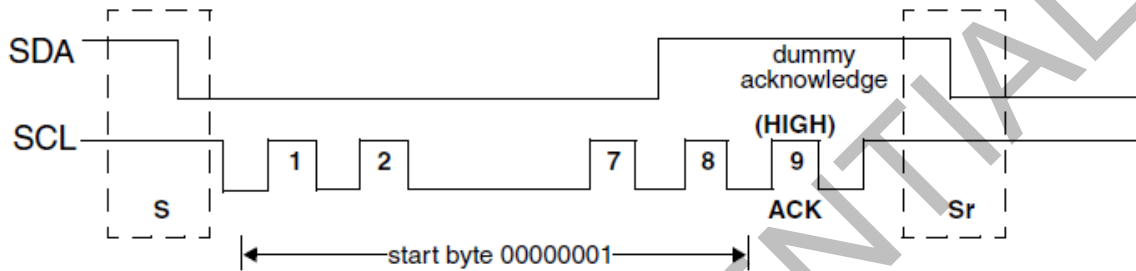


Figure 25: START BYTE Transfer

The START BYTE procedure is as follows:

1. Master generates a START condition.
2. Master transmits the START byte (0000 0001).
3. Master transmits the ACK clock pulse. (Present only to conform with the byte handling format used on the bus)
4. No slave sets the ACK signal to 0.
5. Master generates a RESTART (R) condition.

A hardware receiver does not respond to the START BYTE because it is a reserved address and resets after the RESTART condition is generated.

13.4 MULTIPLE MASTER ARBITRATION

The I2C Controller bus protocol allows multiple masters to reside on the same bus. If there are two masters on the same I2C-bus, there is an arbitration procedure if both try to take control of the bus at the same time by generating a START condition at the same time. Once a master (for example, a microcontroller) has control of the bus, no other master can take control until the first master sends a STOP condition and places the bus in an idle state.

Arbitration takes place on the SDA line, while the SCL line is 1. The master, which transmits a 1 while the other master transmits 0, loses arbitration and turns off its data output stage. The master that lost arbitration can continue to generate clocks until the end of the byte transfer. If both masters are addressing the same slave device, the arbitration could go into the data phase. Figure 26 illustrates the timing of when two masters are arbitrating on the bus.

For high-speed mode, the arbitration cannot go into the data phase because each master is programmed with a unique high-speed master code. This 8-bit code is

defined by the system designer and is set by writing to the High Speed Master Mode Code Address Register, I2C_HS_MADDR. Because the codes are unique, only one master can win arbitration, which occurs by the end of the transmission of the high-speed master code.

Control of the bus is determined by address or master code and data sent by competing masters, so there is no central master nor any order of priority on the bus.

Arbitration is not allowed between the following conditions:

- A RESTART condition and a data bit
- A STOP condition and a data bit
- A RESTART condition and a STOP condition

Slaves are not involved in the arbitration process.

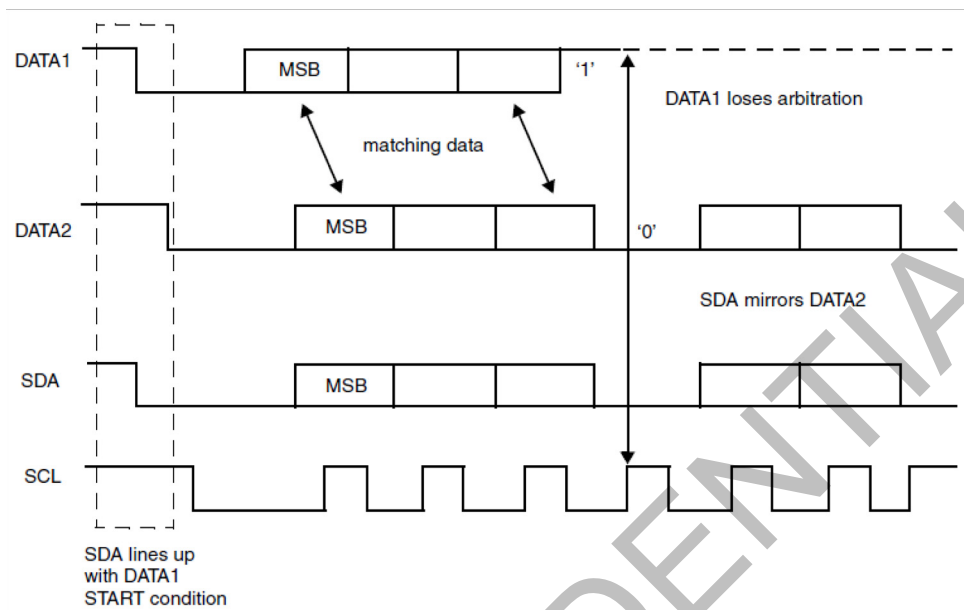


Figure 26: Multiple Master Arbitration

13.5 CLOCK SYNCHRONIZATION

When two or more masters try to transfer information on the bus at the same time, they must arbitrate and synchronize the SCL clock. All masters generate their own clock to transfer messages. Data is valid only during the high period of SCL clock. Clock synchronization is performed using the wired-AND connection to the SCL signal. When the master transitions the SCL clock to 0, the master starts counting the low time of the SCL clock and transitions the SCL clock signal to 1 at the beginning of the next clock period. However, if another master is holding the SCL line to 0, then the master

goes into a HIGH wait state until the SCL clock line transitions to 1.

All masters then count off their high time, and the master with the shortest high time transitions the SCL line to 0. The masters then counts out their low time and the one with the longest low time forces the other master into a HIGH wait state. Therefore, a synchronized SCL clock is generated, which is illustrated in Figure 27. Optionally, slaves may hold the SCL line low to slow down the timing on the I2C bus.

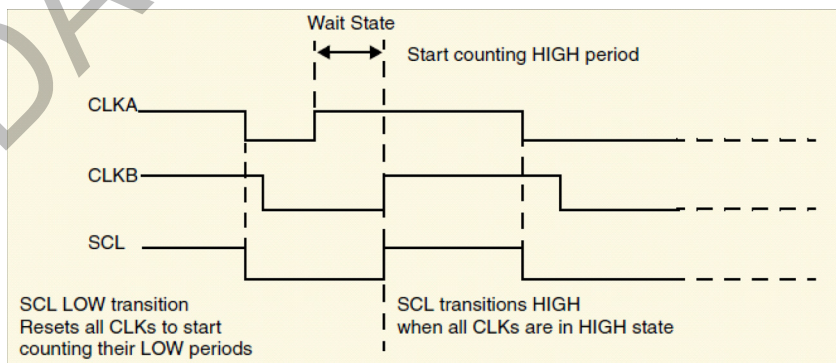


Figure 27: Multiple Master Clock Synchronization

13.6 OPERATION MODES

This section provides information on the following topics:

- Slave mode operation
- Master mode operation
- Disabling I2C Controller

Note 5: It is important to note that the I2C Controller should only be set to operate as an I2C Master, or I2C Slave, but not both simultaneously. This is achieved by ensuring that bit 6 (I2C_SLAVE_DISABLE) and 0 (I2C_MASTER_MODE) of the I2C_CON register are never set to 0 and 1, respectively.

13.6.1 Slave Mode Operation

This section includes the following procedures:

- Initial Configuration
- Slave-Transmitter Operation for a Single Byte
- Slave-Receiver Operation for a Single Byte
- Slave-Transfer Operation For Bulk Transfers

Initial Configuration

To use the I2C Controller as a slave, perform the following steps:

1. Disable the I2C Controller by writing a '0' to bit 0 of the I2C_ENABLE register.
2. Write to the I2C_SAR register (bits 9:0) to set the slave address. This is the address to which the I2C Controller responds.
3. Write to the I2C_CON register to specify which type of addressing is supported (7- or 10-bit by setting bit 3). Enable the I2C Controller in slave-only mode by writing a '0' into bit 6 (I2C_SLAVE_DISABLE) and a '0' to bit 0 (MASTER_MODE).

Note 6: Slaves and masters do not have to be programmed with the same type of addressing 7- or 10-bit address. For instance, a slave can be programmed with 7-bit addressing and a master with 10-bit addressing, and vice versa.

4. Enable the I2C Controller by writing a '1' in bit 0 of the I2C_ENABLE register.

Note 7: Depending on the reset values chosen, steps 2 and 3 may not be necessary because the reset values can be configured. For instance, if the device is only going to be a master, there would be no need to set the slave address because you can configure I2C Controller to have the slave disabled after reset and to enable the master after reset. The values stored are static and do not need to be reprogrammed if the I2C Controller is disabled.

Slave-Transmitter Operation for a Single Byte

When another I2C master device on the bus addresses the I2C Controller and requests data, the I2C Controller acts as a slave-transmitter and the following steps occur:

1. The other I2C master device initiates an I2C transfer with an address that matches the slave address in the I2C_SAR register of the I2C Controller.

2. The I2C Controller acknowledges the sent address and recognizes the direction of the transfer to indicate that it is acting as a slave-transmitter.

3. The I2C Controller asserts the RD_REQ interrupt (bit 5 of the I2C_RAW_INTR_STAT register) and holds the SCL line low. It is in a wait state until software responds. If the RD_REQ interrupt has been masked, due to I2C_INTR_MASK[5] register (M_RD_REQ bit field) being set to 0, then it is recommended that a hardware and/or software timing routine be used to instruct the CPU to perform periodic reads of the I2C_RAW_INTR_STAT register.

- a. Reads that indicate I2C_RAW_INTR_STAT[5] (R_RD_REQ bit field) being set to 1 must be treated as the equivalent of the RD_REQ interrupt being asserted.

- b. Software must then act to satisfy the I2C transfer.

- c. The timing interval used should be in the order of 10 times the fastest SCL clock period the I2C Controller can handle. For example, for 400 kb/s, the timing interval is 25us.

Note 8: The value of 10 is recommended here because this is approximately the amount of time required for a single byte of data transferred on the I2C bus.

4. If there is any data remaining in the TX FIFO before receiving the read request, then the I2C Controller asserts a TX_ABRT interrupt (bit 6 of the I2C_RAW_INTR_STAT register) to flush the old data from the TX FIFO.

Note 9: Because the I2C Controller's TX FIFO is forced into a flushed/reset state whenever a TX_ABRT event occurs, it is necessary for software to release the I2C Controller from this state by reading the I2C_CLR_TX_ABRT register before attempting to write into the TX FIFO. See register I2C_RAW_INTR_STAT for more details.

If the TX_ABRT interrupt has been masked, due to I2C_INTR_MASK[6] register (M_TX_ABRT bit field) being set to 0, then it is recommended that re-using the timing routine (described in the previous step), or a similar one, be used to read the I2C_RAW_INTR_STAT register.

- a. Reads that indicate bit 6 (R_TX_ABRT) being set to 1 must be treated as the equivalent of the TX_ABRT interrupt being asserted.

- b. There is no further action required from software.

- c. The timing interval used should be similar to that described in the previous step for the I2C_RAW_INTR_STAT[5] register.

5. Software writes to the I2C_DATA_CMD register with the data to be written (by writing a '0' in bit 8).

6. Software must clear the RD_REQ and TX_ABRT interrupts (bits 5 and 6, respectively) of the I2C_RAW_INTR_STAT register before proceeding.

If the RD_REQ and/or TX_ABORT interrupts have been masked, then clearing of the I2C_RAW_INTR_STAT register will have already been performed when either the R_RD_REQ or R_TX_ABORT bit has been read as 1.

7. The I2C Controller releases the SCL and transmits the byte.

8. The master may hold the I2C bus by issuing a RESTART condition or release the bus by issuing a STOP condition.

Slave-Receiver Operation for a Single Byte

When another I2C master device on the bus addresses the I2C Controller and is sending data, the I2C Controller acts as a slave-receiver and the following steps occur:

1. The other I2C master device initiates an I2C transfer with an address that matches the I2C Controller's slave address in the I2C_SAR register.

2. The I2C Controller acknowledges the sent address and recognizes the direction of the transfer to indicate that the I2C Controller is acting as a slave-receiver.

3. I2C Controller receives the transmitted byte and places it in the receive buffer.

Note 10: If the RX FIFO is completely filled with data when a byte is pushed, then an overflow occurs and the I2C Controller continues with subsequent I2C transfers. Because a NACK is not generated, software must recognize the overflow when indicated by the I2C Controller (by the R_RX_OVER bit in the I2C_INTR_STAT register) and take appropriate actions to recover from lost data. Hence, there is a real time constraint on software to service the RX FIFO before the latter overflow as there is no way to re-apply pressure to the remote transmitting master. You must select a deep enough RX FIFO depth to satisfy the interrupt service interval of their system.

4. I2C Controller asserts the RX_FULL interrupt (I2C_RAW_INTR_STAT[2] register).

If the RX_FULL interrupt has been masked, due to setting I2C_INTR_MASK[2] register to 0 or setting I2C_TX_TL to a value larger than 0, then it is recommended that a timing routine (described in "Slave-Transmitter Operation for a Single Byte") be implemented for periodic reads of the "I2C_STATUS" on page 138 register. Reads of the I2C_STATUS register, with bit 3 (RFNE) set at 1, must then be treated by software as the equivalent of the RX_FULL interrupt being asserted.

5. Software may read the byte from the I2C_DATA_CMD register (bits 7:0).

6. The other master device may hold the I2C bus by issuing a RESTART condition or release the bus by issuing a STOP condition.

Slave-Transfer operation for bulk transfers

In the standard I2C protocol, all transactions are single

byte transactions and the programmer responds to a remote master read request by writing one byte into the slave's TX FIFO. When a slave (slave-transmitter) is issued with a read request (RD_REQ) from the remote master (master-receiver), at a minimum there should be at least one entry placed into the slave-transmitter's TX FIFO.

I2C Controller is designed to handle more data in the TX FIFO so that subsequent read requests can take that data without raising an interrupt to get more data. Ultimately, this eliminates the possibility of significant latencies being incurred between raising the interrupt for data each time had there been a restriction of having only one entry placed in the TX FIFO.

This mode only occurs when I2C Controller is acting as a slave-transmitter. If the remote master acknowledges the data sent by the slave-transmitter and there is no data in the slave's TX FIFO, the I2C Controller holds the I2C SCL line low while it raises the read request interrupt (RD_REQ) and waits for data to be written into the TX FIFO before it can be sent to the remote master.

If the RD_REQ interrupt is masked, due to bit 5 (M_RD_REQ) of the I2C_INTR_STAT register being set to 0, then it is recommended that a timing routine be used to activate periodic reads of the I2C_RAW_INTR_STAT register. Reads of I2C_RAW_INTR_STAT that return bit 5 (R_RD_REQ) set to 1 must be treated as the equivalent of the RD_REQ interrupt referred to in this section. This timing routine is similar to that described in "Slave-Transmitter Operation for a Single Byte"

The RD_REQ interrupt is raised upon a read request, and like interrupts, must be cleared when exiting the interrupt service handling routine (ISR). The ISR allows you to either write 1 byte or more than 1 byte into the TX FIFO. During the transmission of these bytes to the master, if the master acknowledges the last byte, then the slave must raise the RD_REQ again because the master is requesting for more data.

If the programmer knows in advance that the remote master is requesting a packet of n bytes, then when another master addresses I2C Controller and requests data, the TX FIFO could be written with n number bytes and the remote master receives it as a continuous stream of data. For example, the I2C Controller slave continues to send data to the remote master as long as the remote master is acknowledging the data sent and there is data available in the TX FIFO. There is no need to hold the SCL line low or to issue RD_REQ again.

If the remote master is to receive n bytes from the I2C Controller but the programmer wrote a number of bytes larger than n to the TX FIFO, then when the slave finishes sending the requested n bytes, it clears the TX FIFO and ignores any excess bytes.

The the I2C Controller generates a transmit abort (TX_ABORT) event to indicate the clearing of the TX FIFO in this example. At the time an ACK/NACK is

expected, if a NACK is received, then the remote master has all the data it wants. At this time, a flag is raised within the slave's state machine to clear the leftover data in the TX FIFO. This flag is transferred to the processor bus clock domain where the FIFO exists and the contents of the TX FIFO is cleared at that time.

13.6.2 Master Mode Operation

This section includes the following topics:

- Initial Configuration
- Master Transmit and Master Receive

Initial Configuration

The procedures are very similar and are only different with regard to where the I2C_10BITADDR_MASTER bit is set (either bit 4 of I2C_CON register or bit 12 of I2C_TAR register).

To use the I2C Controller as a master perform the following steps:

1. Disable the I2C Controller by writing 0 to the I2C_ENABLE register.
2. Write to the I2C_CON register to set the maximum speed mode supported (bits 2:1) and the desired speed of the I2C Controller master-initiated transfers, either 7-bit or 10-bit addressing (bit 4).

Ensure that bit 6 I2C_SLAVE_DISABLE = 1 and bit 0 MASTER_MODE = 1

Note 11: Slaves and masters do not have to be programmed with the same type of addressing 7- or 10-bit address. For instance, a slave can be programmed with 7-bit addressing and a master with 10-bit addressing, and vice versa.

3. Write to the I2C_TAR register the address of the I2C device to be addressed (bits 9:0). This register also indicates whether a General Call or a START BYTE command is going to be performed by I2C.
4. Only applicable for high-speed mode transfers. Write to the I2C_HS_MADDR register the desired master code for the I2C Controller. The master code is programmer-defined.
5. Enable the I2C Controller by writing a 1 in bit 0 of the I2C_ENABLE register.
6. Now write transfer direction and data to be sent to the I2C_DATA_CMD register. If the I2C_DATA_CMD register is written before the I2C Controller is enabled, the data and commands are lost as the buffers are kept cleared when I2C Controller is disabled.

This step generates the START condition and the address byte on the I2C Controller. Once I2C Controller is enabled and there is data in the TX FIFO, I2C Controller starts reading the data.

Note 12: Depending on the reset values chosen, steps 2, 3, 4, and 5 may not be necessary because the reset values can be configured. The values stored are static and do not need to be reprogrammed if the I2C Controller is disabled, with the exception of the transfer direction and data.

Master Transmit and Master Receive

The I2C Controller supports switching back and forth between reading and writing dynamically. To transmit data, write the data to be written to the lower byte of the IC_DATA_CMD register. The CMD bit [8] should be written to 0 for I2C write operations.

Subsequently, a read command may be issued by writing "don't cares" to the lower byte of the IC_DATA_CMD register, and a 1 should be written to the CMD bit.

The I2C Controller master continues to initiate transfers as long as there are commands present in the transmit FIFO. If the transmit FIFO becomes empty, the master inserts a STOP condition after completing the current transfer.

13.6.3 Disabling the I2C Controller

The register I2C_ENABLE_STATUS is added to allow software to unambiguously determine when the hardware has completely shutdown in response to the I2C_ENABLE register being set from 1 to 0. Only one register is required to be monitored

Procedure

1. Define a timer interval (ti2c_poll) equal to the 10 times the signalling period for the highest I2C transfer speed used in the system and supported by I2C Controller. For example, if the highest I2C transfer mode is 400 kb/s, then this ti2c_poll is 25us.
 2. Define a maximum time-out parameter, MAX_T_POLL_COUNT, such that if any repeated polling operation exceeds this maximum value, an error is reported.
 3. Execute a blocking thread/process/function that prevents any further I2C master transactions to be started by software, but allows any pending transfers to be completed.
- Note 13:** This step can be ignored if I2C Controller is programmed to operate as an I2C slave only.
4. The variable POLL_COUNT is initialized to zero.
 5. Set I2C_ENABLE to 0.
 6. Read the I2C_ENABLE_STATUS register and test the I2C_EN bit (bit 0). Increment POLL_COUNT by one. If POLL_COUNT >= MAX_T_POLL_COUNT, exit with the relevant error code.
 7. If I2C_ENABLE_STATUS[0] is 1, then sleep for ti2c_poll and proceed to the previous step.
- Otherwise, exit with a relevant success code.

14 UART

The DA14582 contains two identical instances of this block, i.e. UART and UART2.

The UART is compliant to the industry-standard 16550 and is used for serial communication with a peripheral, modem (data carrier equipment, DCE) or data set. Data is written from a master (CPU) over the APB bus to the UART and it is converted to serial form and transmitted to the destination device. Serial data is also received by the UART and stored for the master (CPU) to read back.

There is no DMA support on the UART block since it contains internal FIFOs. Both UARTs support hardware flow control signals (RTS, CTS, DTR, DSR).

Features

- 16 bytes Transmit and receive FIFOs
- Hardware flow control support (CTS/RTS)

- Shadow registers to reduce software overhead and also include a software programmable reset
- Transmitter Holding Register Empty (THRE) interrupt mode
- IrDA 1.0 SIR mode supporting low power mode.
- Functionality based on the 16550 industry standard:
- Programmable character properties, such as number of data bits per character (5-8), optional
- parity bit (with odd or even select) and number of stop bits (1, 1.5 or 2)
- Line break generation and detection
- Prioritized interrupt identification
- Programmable serial data baud rate as calculated by the following: $\text{baud rate} = (\text{serial clock frequency}) / (\text{divisor})$.

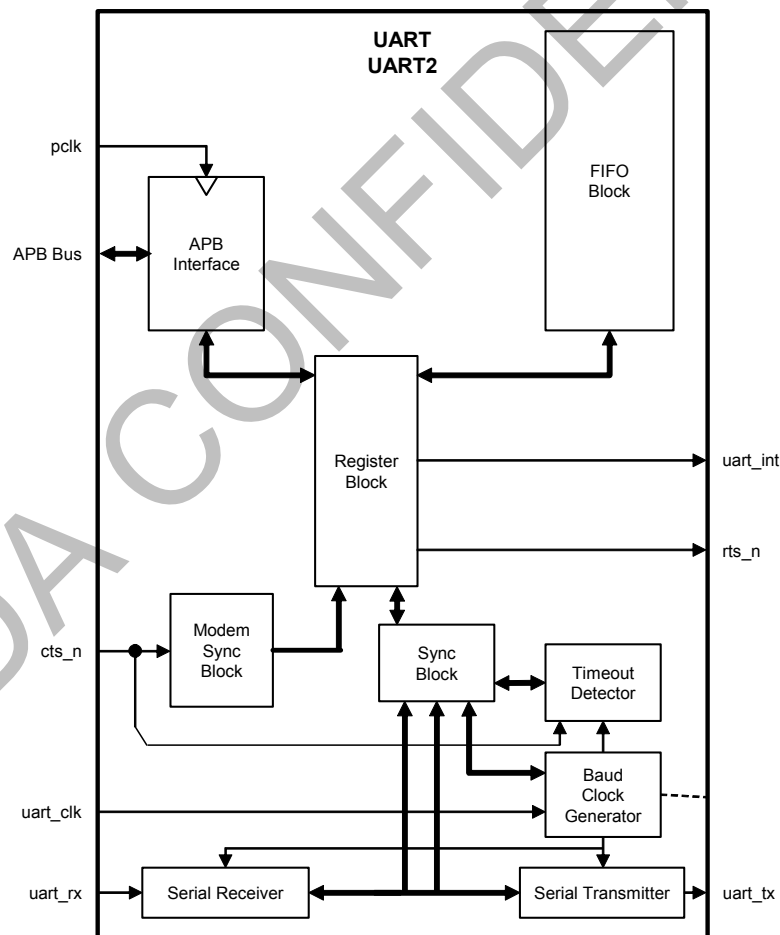


Figure 28: UART Block Diagram

14.1 UART (RS232) SERIAL PROTOCOL

Because the serial communication between the UART and the selected device is asynchronous, additional bits (start and stop) are added to the serial data to indicate the beginning and end. Utilizing these bits allows two devices to be synchronized. This structure of serial data accompanied by start and stop bits is referred to as a character, as shown in Figure 29.

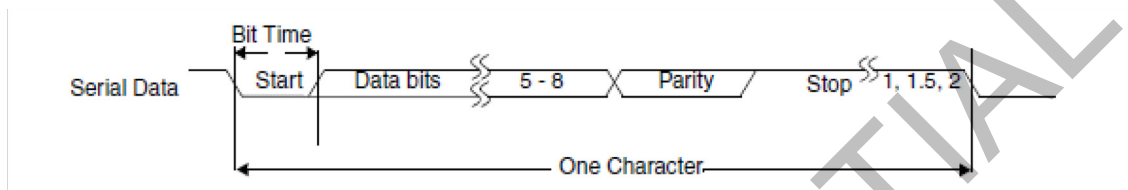


Figure 29: Serial Data Format

An additional parity bit may be added to the serial character. This bit appears after the last data bit and before the stop bit(s) in the character structure to provide the UART with the ability to perform simple error checking on the received data.

The UART Line Control Register (UART_LCR_REG) is used to control the serial character characteristics. The individual bits of the data word are sent after the start bit, starting with the least-significant bit (LSB). These are followed by the optional parity bit, followed by the stop bit(s), which can be 1, 1.5 or 2.

All the bits in the transmission (with exception to the half stop bit when 1.5 stop bits are used) are transmit-

ted for exactly the same time duration. This is referred to as a Bit Period or Bit Time. One BitTime equals 16 baud clocks. To ensure stability on the line the receiver samples the serial input data at approximately the mid point of the Bit Time once the start bit has been detected. As the exact number of baud clocks that each bit was transmitted for is known, calculating the mid point for sampling is not difficult, that is every 16 baud clocks after the mid point sample of the start bit. Figure 30 shows the sampling points of the first couple of bits in a serial character.

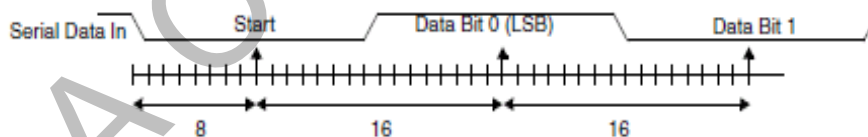


Figure 30: Receiver Serial Data Sampling Points

As part of the 16550 standard an optional baud clock reference output signal (baudout_n) is supplied to provide timing information to receiving devices that require it. The baud rate of the UART is controlled by the serial clock (sclk or pclk in a single clock implementation) and the Divisor Latch Register (DLH and DLL).

14.2 IRDA 1.0 SIR PROTOCOL

The Infrared Data Association (IrDA) 1.0 Serial Infrared (SIR) mode supports bi-directional data communications with remote devices using infrared radiation as the transmission medium. IrDA 1.0 SIR mode specifies a maximum baud rate of 115.2 kBd.

Note 14: Attention. Information provided on IrDA SIR mode in this section assumes that the reader is fully familiar with the IrDA Serial Infrared Physical Layer Specifications. This specification can be obtained from the following website: <http://www.irda.org>

The data format is similar to the standard serial (sout and sin) data format. Each data character is sent serially, beginning with a start bit, followed by 8 data bits, and ending with at least one stop bit. Thus, the number of data bits that can be sent is fixed. No parity information can be supplied and only one stop bit is used while in this mode.

Trying to adjust the number of data bits sent or enable parity with the Line Control Register (LCR) has no

effect. When the UART is configured to support IrDA 1.0 SIR it can be enabled with Mode Control Register (MCR) bit 6. When the UART is not configured to support IrDA SIR mode, none of the logic is implemented and the mode cannot be activated, reducing total gate counts. When SIR mode is enabled and active, serial data is transmitted and received on the sir_out_n and sir_in ports, respectively.

Transmitting a single infrared pulse signals a logic zero, while a logic one is represented by not sending a pulse. The width of each pulse is 3/16th of a normal

serial bit time. Thus, each new character begins with an infrared pulse for the start bit. However, received data is inverted from transmitted data due to the infrared pulses energizing the photo transistor base of the IrDA receiver, pulling its output low. This inverted transistor output is then fed to the UART sir_in port, which then has correct UART polarity. Figure 31 shows the timing diagram for the IrDA SIR data format in comparison to the standard serial format.

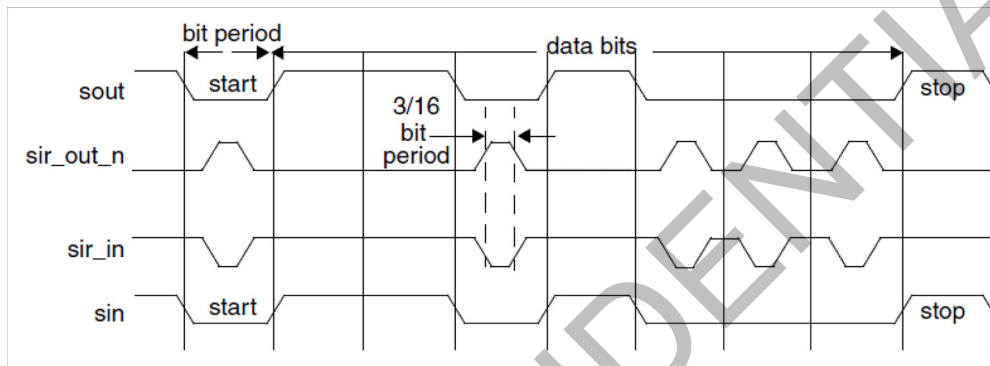


Figure 31: IrDA SIR Data Format

As detailed in the IrDA 1.0 SIR, the UART can be configured to support a low-power reception mode. When the UART is configured in this mode, the reception of SIR pulses of 1.41 μs (minimum pulse duration) is possible, as well as nominal 3/16th of a normal serial bit time. Using this low-power reception mode requires programming the Low Power Divisor Latch (LPDLL/LPDLH) registers. It should be noted that for all sclk frequencies greater than or equal to 7.37 MHz (and obey the requirements of the Low Power Divisor Latch registers), pulses of 1.41 μs are detectable. However there are several values of sclk that do not allow the detection of such a narrow pulse and these are as follows:

Table 19: Low power Divisor Latch Register Values

| SCLK | Low Power Divisor Latch Register Value | Min. Pulse Width for Detection |
|----------|--|--------------------------------|
| 1.84 MHz | 1 | 3.77 μs |
| 3.69 MHz | 2 | 2.086 μs |
| 5.33 MHz | 3 | 1.584 μs |

When IrDA SIR mode is enabled, the UART operation is similar to when the mode is disabled, with one exception; data transfers can only occur in half-duplex fashion when IrDA SIR mode is enabled. This is because the IrDA SIR physical layer specifies a minimum of 10 ms delay between transmission and reception. This 10 ms delay must be generated by software.

14.3 CLOCK SUPPORT

The UART has two system clocks (pclk and sclk). Having the second asynchronous serial clock (sclk) implemented accommodates accurate serial baud rate settings, as well as APB bus interface requirements.

With the two clock design a synchronization module is implemented for synchronization of all control and data across the two system clock boundaries.

A serial clock faster than four-times the PCLK does not leave enough time for a complete incoming character to be received and pushed into the receiver FIFO. However, in most cases, the PCLK signal is faster than the serial clock and this should never be an issue.

The serial clock modules must have time to see new register values and reset their respective state machines. This total time is guaranteed to be no more than eight clock cycles of the slower of the two system clocks. Therefore, no data should be transmitted or received before this maximum time expires, after initial configuration.

For a clear view of the baud rate generation and the constraints, please refer to [section 11.4](#).

14.4 INTERRUPTS

The assertion of the UART interrupt (UART_INT) occurs whenever one of the several prioritized interrupt types are enabled and active. The following interrupt types can be enabled with the IER register:

- Receiver Error
- Receiver Data Available

- Character Timeout (in FIFO mode only)
- Transmitter Holding Register Empty at/below threshold (in Programmable THRE interrupt mode)

When an interrupt occurs the master accesses the UART_IIR_REG to determine the source of the interrupt before dealing with it accordingly. These interrupt types are described in more detail in [Table 20](#)

Table 20: UART Interrupt Priorities

| Interrupt Id | Interrupt Set and Reset Functions | | | |
|--------------|-----------------------------------|------------------------------------|--|--|
| Bits [3-0] | Priority Level | Interrupt Type | Interrupt Source | Interrupt Reset Control |
| 0001 | - | None | | |
| 0110 | Highest | Receiver Line status | Overrun/parity/ framing errors or break interrupt | Reading the line status register |
| 0100 | 1 | Receiver Data Available | Receiver data available (non-FIFO mode or FIFOs disabled) or RCVR FIFO trigger level reached (FIFO mode and FIFOs enabled) | Reading the receiver buffer register (non-FIFO mode or FIFOs disabled) or the FIFO drops below the trigger level (FIFO mode and FIFOs enabled) |
| 1100 | 2 | Character timeout indication | No characters in or out of the RCVR FIFO during the last 4 character times and there is at least 1 character in it during this time. | Reading the receiver buffer register |
| 0010 | 3 | Transmitter holding register empty | Transmitter holding register empty (Prog. THRE Mode disabled) or XMIT FIFO at or below threshold (Prog. THRE Mode enabled). | Reading the IIR register (if source of interrupt); or, writing into THR (FIFOs or THRE Mode not selected or disabled) or XMIT FIFO above threshold (FIFOs and THRE Mode selected and enabled). |
| 0000 | 4 | Reserved | | |
| 0111 | Lowest | Reserved | - | - |

14.5 PROGRAMMABLE THRE INTERRUPT

The UART can be configured to have a Programmable THRE Interrupt mode available to increase system performance.

When Programmable THRE Interrupt mode is selected it can be enabled via the Interrupt Enable Register (IER[7]). When FIFOs and the THRE Mode are implemented and enabled, THRE Interrupts are active at, and below, a programmed transmitter FIFO empty threshold level, as opposed to empty, as shown in the flowchart in [Figure 32](#).

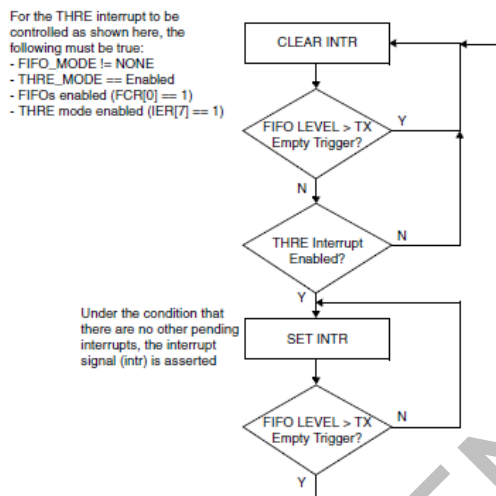


Figure 32: Flowchart of Interrupt Generation for Programmable THRE Interrupt Mode

This threshold level is programmed into FCR[5:4]. The available empty thresholds are: empty, 2, ¼ and ½. See UART_FCR_REG for threshold setting details. Selection of the best threshold value depends on the system's ability to begin a new transmission sequence in a timely manner. However, one of these thresholds should prove optimum in increasing system performance by preventing the transmitter FIFO from running empty.

In addition to the interrupt change, Line Status Register (LSR[5]) also switches function from indicating transmitter FIFO empty, to FIFO full. This allows software to fill the FIFO each transmit sequence by polling LSR[5] before writing another character. The flow then becomes, "fill transmitter FIFO whenever an interrupt

occurs and there is data to transmit", instead of waiting until the FIFO is completely empty. Waiting until the FIFO is empty causes a performance hit whenever the system is too busy to respond immediately.

Even if everything else is selected and enabled, if the FIFOs are disabled via FCR[0], the Programmable THRE Interrupt mode is also disabled. When not selected or disabled, THRE interrupts and LSR[5] function normally (both reflecting an empty THR or FIFO). The flowchart of THRE interrupt generation when not in programmable THRE interrupt mode is shown in Figure 33.

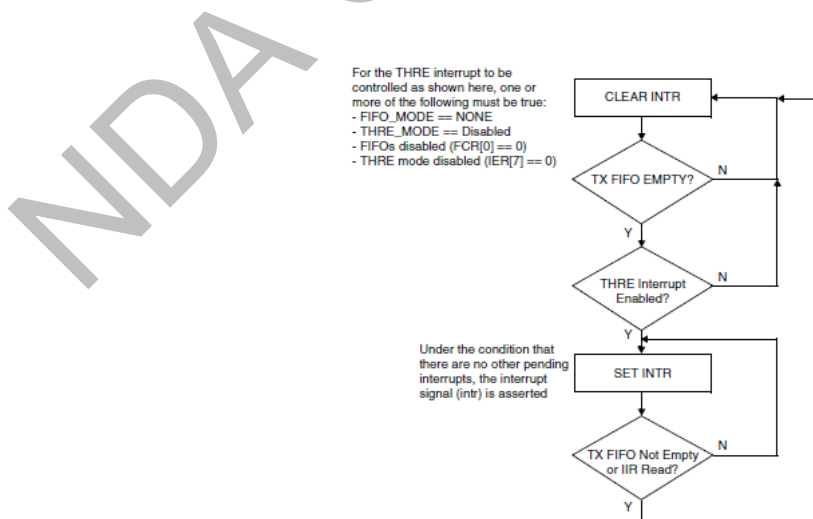


Figure 33: Flowchart of Interrupt Generation When Not in Programmable THRE Interrupt Mode

14.6 SHADOW REGISTERS

The shadow registers shadow some of the existing register bits that are regularly modified by software. These can be used to reduce the software overhead that is introduced by having to perform read-modify writes.

- UART_SRBR_REG support a host burst mode where the host increments its address but still accesses the same Receive buffer register
- UART_STHR support a host burst mode where the host increments its address but still accesses the same transmit holding register.
- UART_SFE_REG accesses the FCR[0] register without accessing the other UART_FCR_REG bits.
- UART_SRT_REG accesses the FCR[7-6] register without accessing the other UART_FCR_REG bits.
- UART_STER_REG accesses the FCR[5-4] register without accessing the other UART_FCR_REG bits.

14.7 DIRECT TEST MODE

The on-chip UARTS can be used for the Direct Test Mode required for the final product PHY layer testing. It can be done either over the HCI layer, which engages a full CTS/RTS UART or using a 2-wire UART directly as described in the Bluetooth Low Energy Specification (Volume 6, Part F).

15 SPI+ Interface

This interface supports a subset of the Serial Peripheral Interface SPI™. The serial interface can transmit and receive 8, 16 or 32 bits in master/slave mode and transmit 9 bits in master mode. The SPI+ interface has enhanced functionality with bidirectional 2x16-bit word FIFOs.

SPI™ is a trademark of Motorola, Inc.

Features

- Slave and Master mode
- 8 bit, 9 bit, 16 bit or 32 bit operation

- Clock speeds up to 16 MHz for the SPI controller. Programmable output frequencies of SPI source clock divided by 1, 2, 4, 8
- SPI clock line speed up to 8 MHz
- SPI mode 0, 1, 2, 3 support (clock edge and phase)
- Programmable SPI_DO idle level
- Maskable Interrupt generation
- Bus load reduction by unidirectional writes-only and reads-only modes.
- Built-in RX/TX FIFOs for continuous SPI bursts.

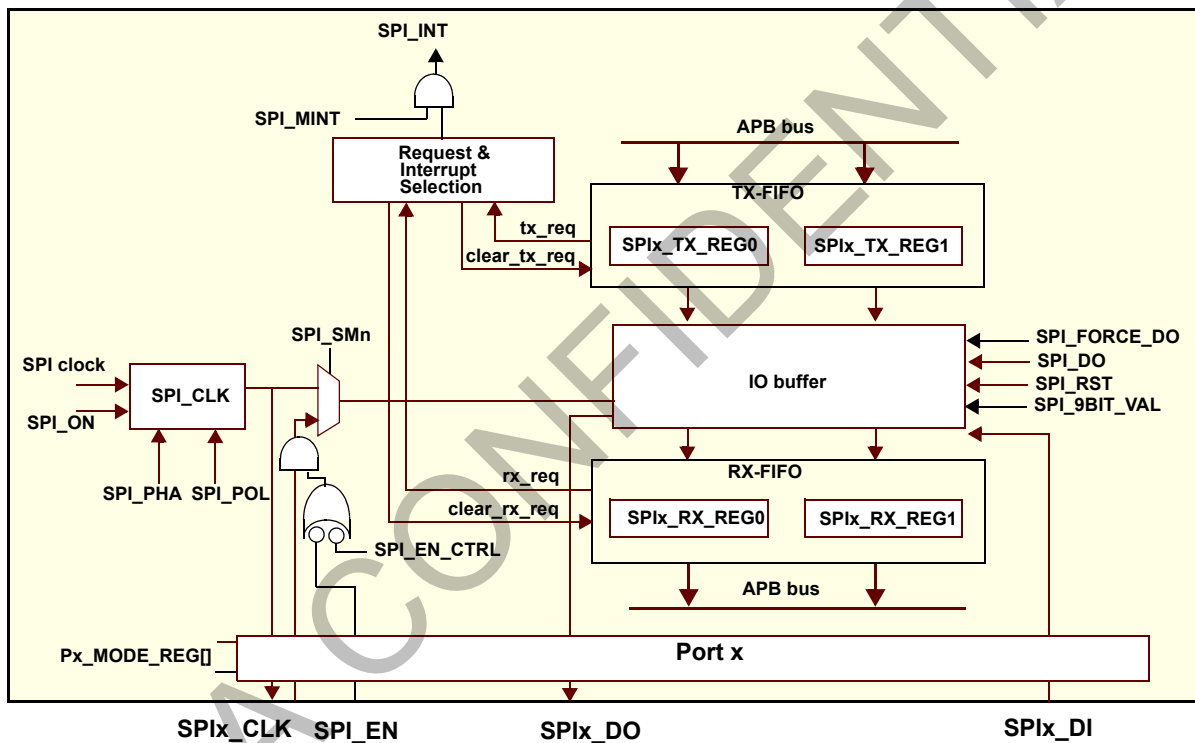


Figure 34: SPI Block Diagram

15.1 OPERATION WITHOUT FIFOS

This mode is the default mode.

Master Mode

To enable SPI™ operation, first the individual port signal must be enabled. Next the SPI must be configured in SPI_CTRL_REG, for the desired mode. Finally bit **SPI_ON** must be set to 1.

A SPI transfer cycle starts after writing to the SPIx_RX_TX_REG0. In case of 32 bits mode, the SPIx_RX_TX_REG1 must be written first. Writing to SPIx_RX_TX_REG0 also sets the SPI_TXH. As soon as the holding register is copied to the IO buffer, the SPI_TXH is reset and a serial transfer cycle of 8/9/16/32 clock-cycles is started which causes 8/9/16/32 bits

to be transmitted on SPIx_DO. Simultaneously, data is received on SPIx_DI and shifted into the IO buffer. The transfer cycle finishes after the 8th/9th/16th/32nd clock cycle and SPI_INT_BIT bit is set in the SPIx_CTRL_REG and SPI_INT_PENDING bit in (RE)SET_INT_PENDING_REG is set. The received bits in the IO buffer are copied to the SPIx_RX_TX_REG0 (and SPIx_RX_TX_REG1 in case of 32 bits mode) where they can be read by the CPU.

Interrupts to the ICU can be disabled using the SPI_MINT bit. To clear the SPI interrupt source, any value to SPIx_CLEAR_INT_REG must be written. Note however that SPI_INT will be set as long as the RX-FIFO contains unread data.

Slave Mode

The slave mode is selected with **SPI_SMn** set to 1 and the **Px_MODE_REG** must also select **SPIx_CLK** as input. The functionality of the IO buffer in slave and master mode is identical. The SPI module clocks data in on **SPIx_DI** and out on **SPIx_DO** on every active edge of **SPIx_CLK**. As shown in [Figure 35](#) to [Figure 38](#). SPI1 has an active low clock enable **SPI1_EN** which can be enabled with bit **SPI_EN_CTRL=1**.

In slave mode the internal SPI clock must be more than four times the **SPIx_CLK**

In slave mode the **SPI_EN** serves as a clock enable and bit synchronization. If enabled with bit **SPI_EN_CTRL**. As soon as **SPI_EN** is deactivated between the MSB and LSB bits, the I/O buffer is reset.

SPI_POL and SPI_PHA

The phase and polarity of the serial clock can be changed with bits **SPI_POL** and **SPI_PHA** in the **SPIx_CTRL_REG**.

SPI_DO Idle Levels

The idle level of signal **SPI_DO** depends on the master or slave mode and polarity and phase mode of the clock.

In master mode pin **SPIx_DO** gets the value of bit **SPI_DO** if the SPI is idle in all modes. Also if slave in SPI modes 0 and 2, **SPI_DO** is the initial and final idle level.

In SPI modes 1 and 3 however there is no clock edge after the sampled lsb and pin **SPIx_DO** gets the lsb value of the IO buffer. If required, the **SPIx_DO** can be forced to the **SPI_DO** bit level by resetting the SPI to the idle state by shortly setting bit **SPI_RST** to 1. (Optionally **SPI_FORCE_DO** can be set, but this does not reset the IO buffer). The following diagrams show the timing of the SPITM interface.

Writes Only Mode

In “writes only” mode (**SPI_FIFO_MODE** = “10”) only the TX-FIFO is used. Received data will be copied to the **SPIx_RX_TX_REGx**, but if a new SPI transfer is finished before the old data is read from the memory, this register will be overwritten.

SPI_INT acts as a tx_request signal, indicating that there is still place in the FIFO. It will be ‘0’ when the FIFO is full or else ‘1’ when it’s not full. This is also indicated in the **SPIx_CTRL_REG[SPI_TXH]**, which is ‘1’ if the TX-FIFO is full. Writing to the FIFO if this bit is still 1, will result in transmission of undefined data. If all data has been transferred, **SPIx_CTRL_REG1[SPI_BUSY]** will become ‘0’.

Reads Only Mode

In “reads only” mode (**SPI_FIFO_MODE** = “01”) only the RX-FIFO is used. Transfers will start immediately when the SPI is turned on in this mode. In transmit direction the **SPI_DO** pin will transmit the IO buffer contents being the actual value of the **SPIx_TX_REGx**

(all 0’s after reset). This means that no dummy writes are needed for reads only transfers.

In **Slave Mode** transfers only take place if the external master initiates them, but in master mode this means that transfers will continue until the RX-FIFO is full. If this happens **SPIx_CTRL_REG1[SPI_BUSY]** will become ‘0’. If exactly N words need to be read from SPI device, first read (N - *fifosize*+1) words. Then wait until the **SPI_BUSY** becomes ‘0’, set **SPI_FIFO_MODE** to “00” and finally read the remaining (*fifosize* +1) words. Here *fifosize* is 4/2/1 words for 8/16/32 bits mode respectively.

If this is not done, more data will be read from the SPI device until the FIFO is completely filled, or the SPI is turned off.

Bidirectional Transfers with FIFO

If **SPI_FIFO_MODE** is “00”, both registers are used as a FIFO. **SPI_TXH** indicates that TX-FIFO is full, **SPI_INT** indicates that there is data in the RX-FIFO.

15.2 9 BITS MODE

The 9 bits mode can be used to support 9 bits displays and is selected with **SPIx_CTRL_REG[SPI_WORD]** set to ‘11’. The value of the 9th bit, set in the **SPIx_CTRL_REG1[SPI_9BIT_VAL]** and is used to determine if the next 8 bits form a command word or data word. Because the 9th bit is not part of the data, the FIFOs are still used in the 8 bits mode. The 9th bit is received but not saved because it is shifted out of the 8 bits shift register upon reception.

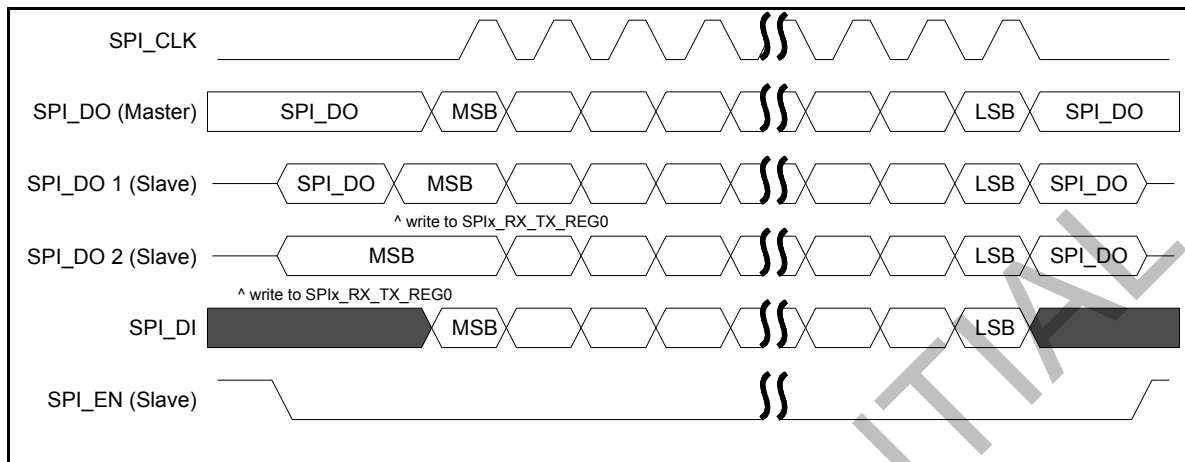


Figure 35: SPI Master/Slave, Mode 0: SPI_POL=0 and SPI_PHA=0

Note 15: If 9 bits SPI mode, the MSB bit in transmit direction is determined by bit SPIx_CTRL_REG[SPI_9BIT_VAL]. In receive direction, the MSB is received but not stored.

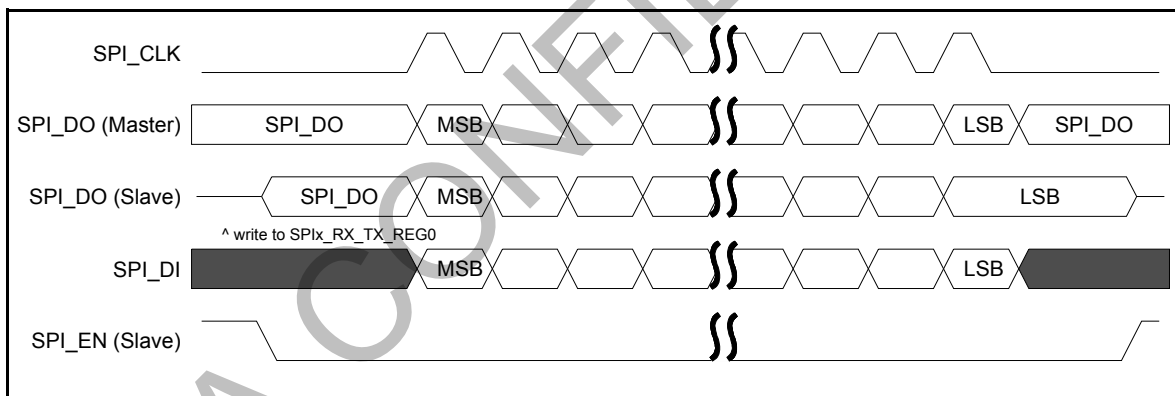


Figure 36: SPI Master/Slave, Mode 1: SPI_POL=0 and SPI_PHA=1

For the MSB bit refer to note 15.

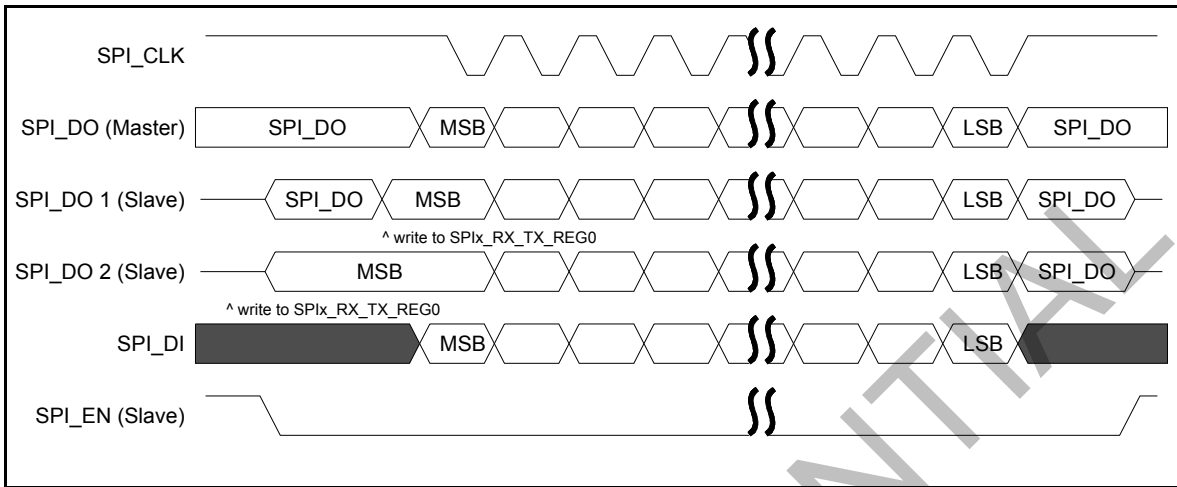


Figure 37: SPI Master/Slave, Mode 2: SPI_POL=1 and SPI_PHA=0

For the MSB bit refer to note 15.

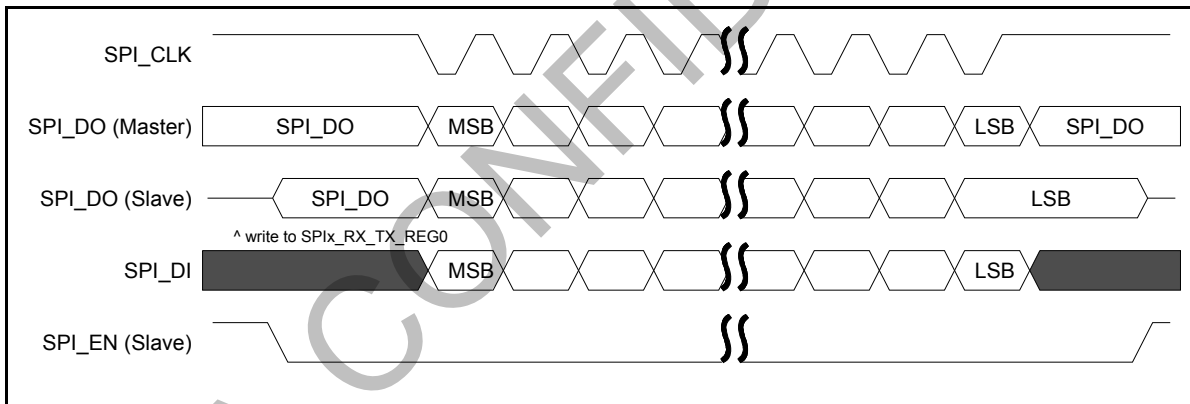


Figure 38: SPI Master/Slave, Mode 3: SPI_POL=1 and SPI_PHA=1

For the MSB bit refer to note 15.

16 Quadrature Decoder

The DA14582 has a integrated quadrature decoder that can automatically decode the signals for the X, Y and Z axes of a HID input device, reporting step count and direction. This block can be used for waking up the chip as soon as there is any kind of movement from the external device connected to it. The block diagram of the quadrature decoder is presented in Figure 39.

Features

- Three 16-bit signed counters that provide the step count and direction on each of the axes (X, Y and Z)
- Programmable system clock sampling at maximum 16 MHz.
- APB interface for control and programming
- Programmable source from P0, P1 and P2 ports
- Digital filter on the channel inputs to avoid spikes

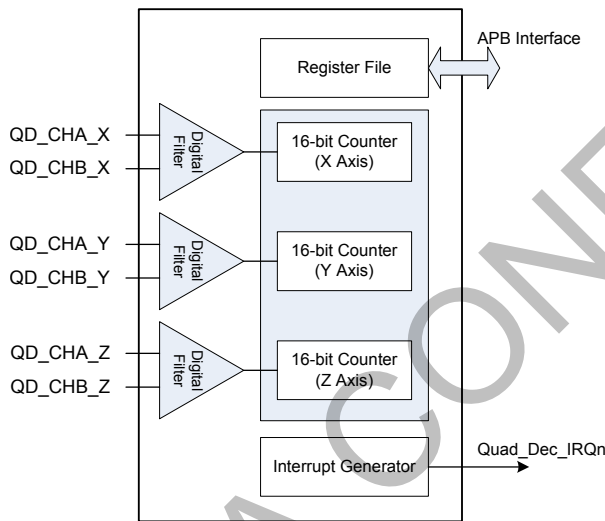


Figure 39: Quadrature Decoder Block Diagram

Channels are expected to provide a pulse train with 90 degrees rotation as displayed in the following figures:

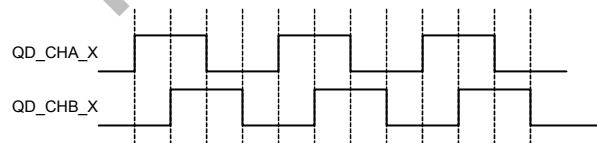


Figure 40: Moving Forward on Axis X

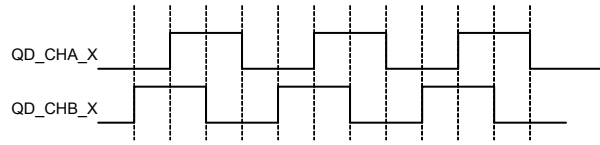


Figure 41: Moving Backwards on Axis X

Depending on whether channel A or channel B is leading in phase, the quadrature decoding block calculates the direction on the related axis. Furthermore, the signed counter value represents the number of steps moved.

Since six channels are required (two for each axis), all P0 and P1 signals and some of the P2 port can be mapped onto this block. The user can choose which GPIOs to use for the channels by programming the QDEC_CTRL2_REG register.

The digital filter eliminates any spike shorter than two clock periods. The counter holds the movement events of the channel. When a channel is disabled the counter is reset. The counters are accessible via the APB bus.

The quadrature decoder operates on the system clock. The QDEC_CLOCKDIV register defines the number of clock cycles of the period at which the decoding logic samples the data on the channel inputs.

17 Wake-Up Timer

The Wake-up timer can be programmed to wake up the DA14582 from power down mode after a pre-programmed number of GPIO events.

Each of the GPIO inputs can be selected to generate an event by programming the corresponding WKUP_SELECT_Px_REG register. When all WKUP_SELECT_Px_REG registers are configured to generate a wake-up interrupt, a toggle on any GPIO will wake up the system.

The input signal edge can be selected by programming the WKUP_POL_Px_REG register.

The block diagram illustrating the Wake-up function is shown in Figure 42.

Features

- Monitors any GPIO state change
- Implements debouncing time from 0 up to 63 ms
- Accumulates external events and compares the number to a programmed value
- Generates an interrupt to the CPU

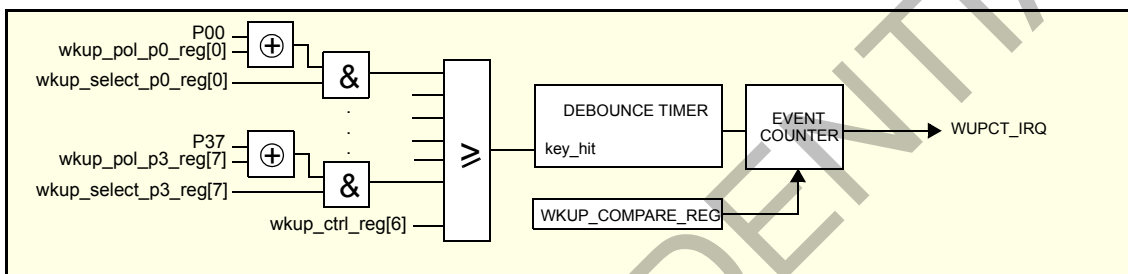


Figure 42: Wake-Up Timer Block Diagram

A LOW to HIGH level transition on the selected input port, while WKUP_POL_Px_REG[y] = 0, sets internal signal “key_hit” to ‘1’. This signal triggers the event counter state machine as shown in Figure 43.

The debounce timer is loaded with value WKUP_CTRL_REG[WKUP_DEB_VALUE]. The timer counts down every 1 ms. If the timer reaches 0 and the “key_hit” signal is still ‘1’, the event counter will be incremented.

The event counter is edge sensitive. After detecting an active edge a reverse edge must be detected first before it goes back to the IDLE state and from there starts waiting for a new active edge.

A minimum pulse duration of 2 sleep clock cycles must be applied to the GPIO to ensure a successful system wake-up.

If the event counter is equal to the value set in the WKUP_COMPARE_REG register, the counter will be reset and an interrupt will be generated, if it was enabled by WKUP_CTRL_REG[ENABLE_IRQ].

The interrupt can be cleared by writing any value to register WKUP_RESET_IRQ_REG.

The event counter can be reset by writing any value to register WKUP_RESET_CNTR_REG.

The value of the event counter can be read at any time by reading register WKUP_COUNTER_REG.

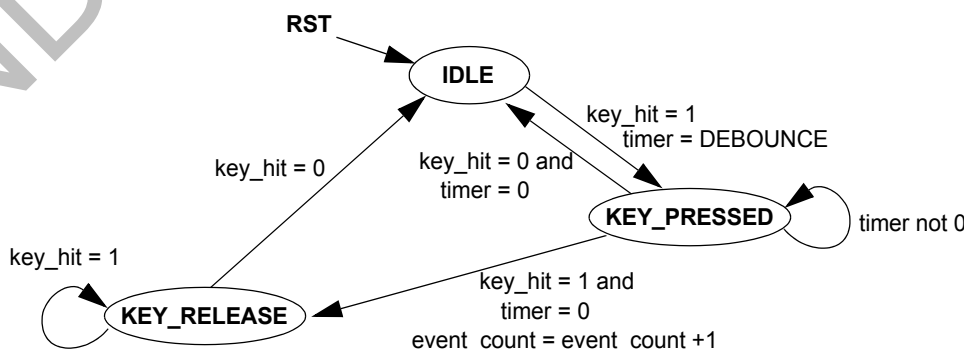


Figure 43: Event Counter State Machine for the Wake-up Interrupt Generator

18 General Purpose Timers

The Timer block contains 2 timer modules that are software controlled, programmable and can be used for various tasks. Timer 0 is a 16-bit general purpose timer with a PWM output capability. Timer 2 is a 14-bit counter that generates three identical PWM signals in a quite flexible manner.

18.1 TIMER 0

Timer 0 is a 16-bit general purpose software programmable timer, which has the ability of generating Pulse Width Modulated signals, namely PWM0 and PWM1. It also generates the SWTIM_IRQ interrupt to the ARM Cortex-M0. It can be configured in various modes regarding output frequency, duty cycle and the modulation of the PWM signals.

Features

- 16-bit general purpose timer
- Ability to generate 2 Pulse Width Modulated signals (PWM0 and PWM1)
- Programmable output frequency:

$$f = \frac{(16, 8, 4, 2 \text{ MHz or } 32 \text{ kHz})}{(M + 1) + (N + 1)}$$
 with $N = 0$ to $(2^{16}-1)$, $M = 0$ to $(2^{16}-1)$
- Programmable duty cycle:

$$\delta = \frac{M + 1}{(M + 1) + (N + 1)} \times 100 \%$$
- Separately programmable interrupt timer:

$$T = \frac{(16, 8, 4, 2 \text{ MHz or } 32 \text{ kHz})}{(ON + 1)}$$

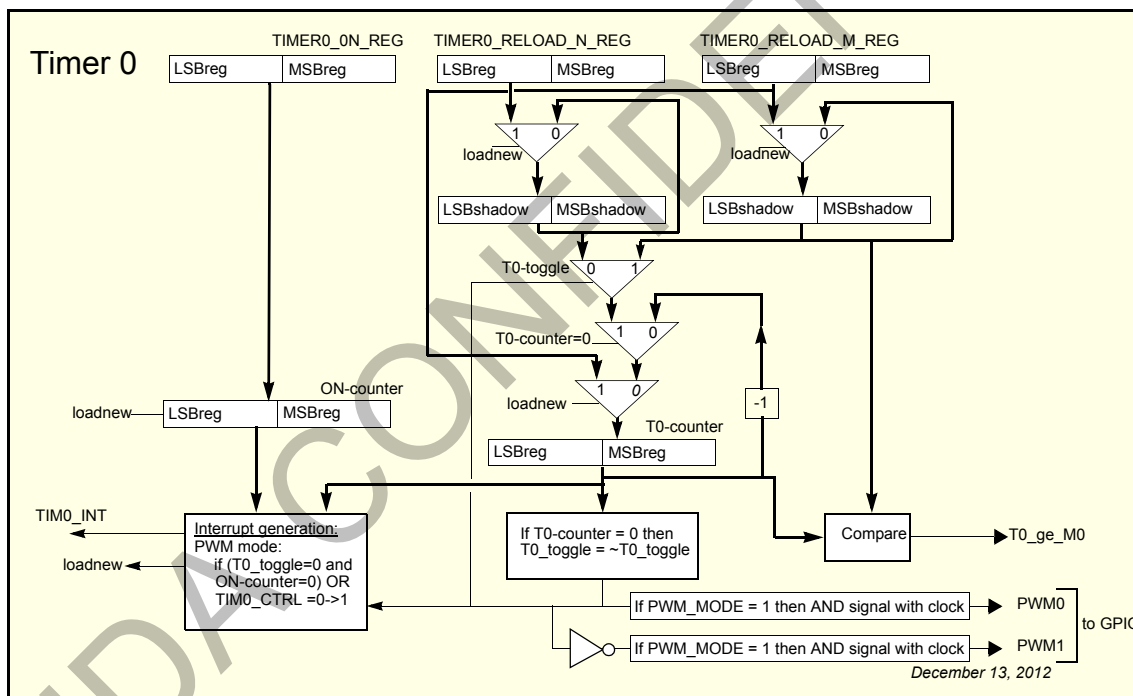


Figure 44: Timer 0 Block Diagram

Figure 44 shows the block diagram of Timer 0. The 16 bits timer consists of two counters: T0-counter and ON-counter, and three registers: TIMER0_RELOAD_M_REG, TIMER0_RELOAD_N_REG and TIMER0_ON_REG. Upon reset, the counter and register values are 0x0000. Timer 0 will generate a Pulse Width Modulated signal PWM0. The frequency and duty cycle of PWM0 are determined by the contents of the TIMER0_RELOAD_N_REG and the TIMER0_RELOAD_M_REG registers.

The timer can run at five different clocks: 16 MHz, 8 MHz, 4 MHz, 2 MHz or 32 kHz. The 32 kHz clock is

selected by default with bit TIM0_CLK_SEL in the TIMER0_CTRL_REG register. This 'slow' clock has no enabling bit. The other four options can be selected by setting the TIM0_CLK_SEL bit and the TMR_ENABLE bit in the CLK_PER_REG (default disabled). This register also controls the frequency via the TMR_DIV bits. An extra clock divider is available that can be activated via bit TIM0_CLK_DIV of the timer control register TIMER0_CTRL_REG. This clock divider is only used for the ON-counter and always divides by 10.

Timer 0 operates in PWM mode. The signals PWM0 and PWM1 can be mapped to any GPIOs.

Timer 0 PWM Mode

If bit TIM0_CTRL in the TIMER0_CTRL_REG is set, Timer 0 will start running. SWTIM_IRQ will be generated and the T0-counter will load its start value from the TIMER0_RELOAD_M_REG register, and will decrement on each clock. The ON-counter also loads its start value from the TIMER0_ON_REG register and decrements with the selected clock.

When the T0-counter reaches zero, the internal signal T0-toggle will be toggled to select the TIMER0_RELOAD_N_REG whose value will be loaded in the T0-counter. Each time the T0-counter reaches zero it will alternately be reloaded with the values of the M0- and N0-shadow registers respectively. PWM0 will be high when the M0-value decrements and low when the N0-value decrements. For PWM1 the opposite is applicable since it is inverted. If bit PWM_MODE in the TIMER0_CTRL_REG register is set, the PWM signals are not HIGH during the 'high time' but output a clock in that stage. The frequency is based on the clock settings defined in the CLK_PER_REG register (also in 32 kHz mode), but the selected clock frequency is divided by two to get a 50 % duty cycle.

If the ON-counter reaches zero it will remain zero until the T0-counter also reaches zero, while decrementing the value loaded from the

TIMER0_RELOAD_N_REG register (PWM0 is low). The counter will then generate an interrupt (SWTIM_IRQ). The ON-counter will be reloaded with the value of the TIMER0_ON_REG register. The T0-counter as well as the M0-shadow register will be loaded with the value of the TIMER0_RELOAD_M_REG register. At the same time, the N0-shadow register will be loaded by the TIMER0_RELOAD_N_REG register. Both counters will be decremented on the next clock again and the sequence will be repeated.

Note that it is possible to generate interrupts at a high rate, when selecting a high clock frequency in combination with low counter values. This could result in missed interrupt events.

During the time that the ON-counter is non-zero, new values for the ON-register, M0-register and N0-register can be written, but they are not used by the T0-counter until a full cycle is finished. More specifically, the newly written values in the TIMER0_RELOAD_M_REG and TIMER0_RELOAD_N_REG registers are only stored into the shadow registers when the ON-counter and the T0-counter have both reached zero and the T0-counter was decrementing the value loaded from the TIMER0_RELOAD_N_REG register (see Figure 45).

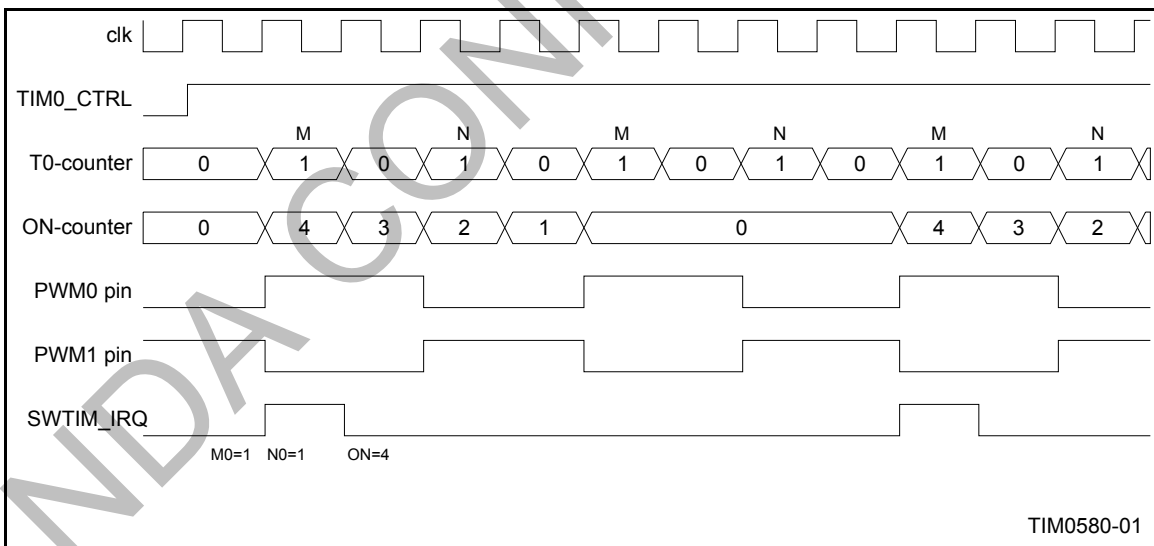


Figure 45: Timer 0 PWM Mode

At start-up both counters and the PWM0 signal are LOW so also at start-up an interrupt is generated. If Timer 0 is disabled all flip-flops, counters and outputs are in reset state except for the ON-register, the TIMER0_RELOAD_N_REG register and the TIMER0_RELOAD_M_REG register.

The timer input registers ON-register, TIMER0_RELOAD_N_REG and TIMER0_RELOAD_M_REG can be written and the

counter registers ON-counter and T0-counter can be read. When reading from the address of the ON-register, the value of the ON-counter is returned. Reading from the address of either the TIMER0_RELOAD_N_REG or the TIMER0_RELOAD_M_REG register, returns the value of the T0-counter.

It is possible to freeze Timer 0 with bit FRZ_SWTIM of the register SET_FREEZE_REG. When the timer is fro-

zen the timer counters are not decremented. This will freeze all the timer registers at their last value. The timer will continue its operation again when bit FRZ_SWTIM is cleared via register RESET_FREEZE_REG.

18.2 TIMER 2

Timer 2 has three Pulse Width Modulated (PWM) outputs. The block diagram is shown in Figure 46.

Features

- 14-bit general purpose timer
- Ability to generate 3 Pulse Width Modulated signals (PWM2, PWM3 and PWM4)

- Input clock frequency:

$$f_{IN} = \frac{\text{sys_clk}}{N} \text{ with } N = 1, 2, 4 \text{ or } 8$$

and sys_clk = 16 MHz or 32 kHz

- Programmable output frequency:

$$f_{OUT} = \left(\frac{f_{IN}}{2}\right) \text{ to } \left(\frac{f_{IN}}{2^{14}-1}\right)$$

- Three outputs with programmable duty cycle from 0 % to 100 %
- Used for white LED intensity (on/off) control

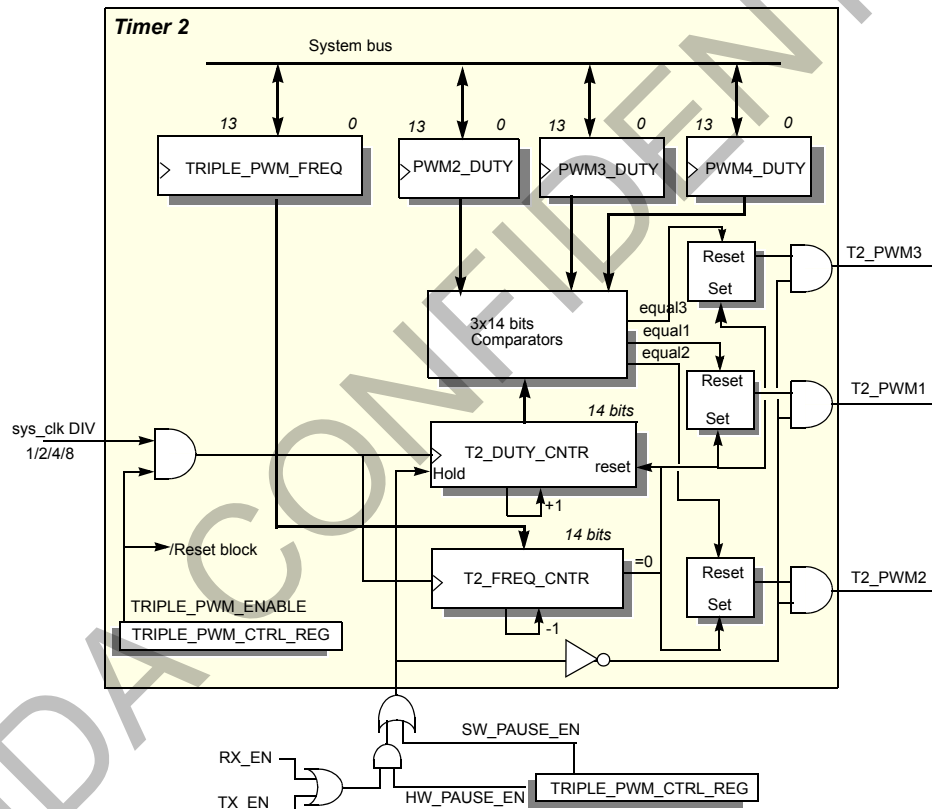


Figure 46: Timer 2 PWM Block Diagram

The Timer 2 is clocked with the system clock divided by TMR_DIV (1, 2, 4 or 8) and can be enabled with TRIPLE_PWM_CTRL_REG[TRIPLE_PWM_ENABLE].

T2_FREQ_CNTR determines the output frequency of the T2_PWMn output. This counter counts down from the value stored in register TRIPLE_PWM_FREQUENCY. At counter value 0, T2_FREQ_CNTR sets the T2_PWMn output to '1' and the counter is reloaded again.

T2_DUTY_CNTR is an up-counter that determines the

duty cycle of the T2_PWMn output signal. After the block is enabled, the counter starts from 0. If T2_DUTY_CNTR is equal to the value stored in the respective PWMn_DUTY_CYCLE register, this resets the T2_PWMn output to 0. T2_DUTY_CNTR is reset when TRIPLE_PWM_FREQUENCY is 0.

Note that the value of PWMn_DUTY_CYCLE must be less or equal than TRIPLE_PWM_FREQUENCY.

The Timer 2 is enabled/disabled by programming the TRIPLE_PWM_CTRL_REG[TRIPLE_PWM_EN] bit.

The timing diagram of Timer 2 is shown in Figure 47.

Freeze Function

During RF activity it may be desirable to temporarily suppress the PWM switching noise. This can be done by setting TRIPLE_PWM_CTRL_REG[HW_PAUSE_EN] = 1. The effect is that whenever there is a transmission or a reception process from the Radio, T2_DUTY_CNTR is frozen and T2_PWMx output is switched to '0' to disable the selected T2_PWM1, T2_PWM2, T2_PWM3. As soon as the Radio is idle (i.e. RX_EN or TX_EN sig-

nals are zero), T2_DUTY_CNTR resumes counting and finalizes the remaining part of the PWM duty cycle.

TRIPLE_PWM_CTRL_REG[SW_PAUSE_EN] can be set to '0' to disable the automatic, hardware driven freeze function of the duty counter and keep the duty cycle constant.

Note that the RX_EN and TX_EN signals are not software driven but controlled by the BLE core hardware.

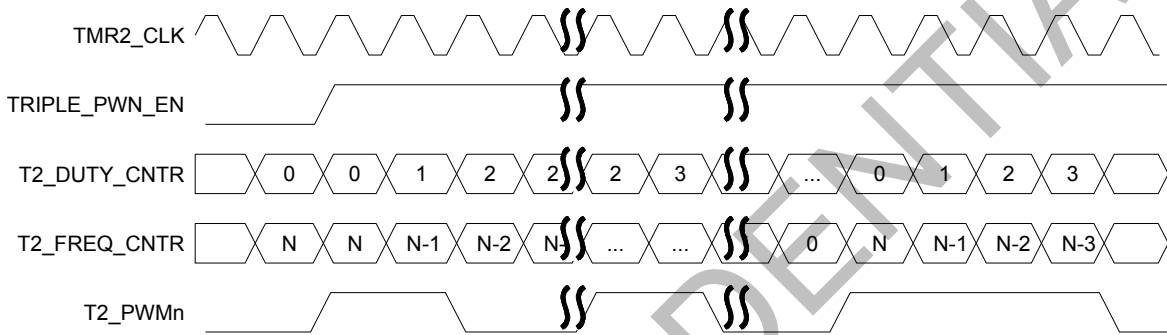


Figure 47: Timer 2 PWM Timing Diagram

19 Watchdog Timer

The Watchdog timer is an 8-bit timer with sign bit that can be used to detect an unexpected execution sequence caused by a software run-away and can generate a full system reset or a Non-Maskable Interrupt (NMI).

Features

- 8 bits down counter with sign bit, clocked with a 10.24 ms clock for a maximum 2.6 s time-out.

- Non-Maskable Interrupt (NMI) or WDOG reset.
- Optional automatic WDOG reset if NMI handler fails to update the Watchdog register.
- Non-maskable Watchdog freeze of the Cortex-M0 Debug module when the Cortex-M0 is halted in Debug state.
- Maskable Watchdog freeze by user program.

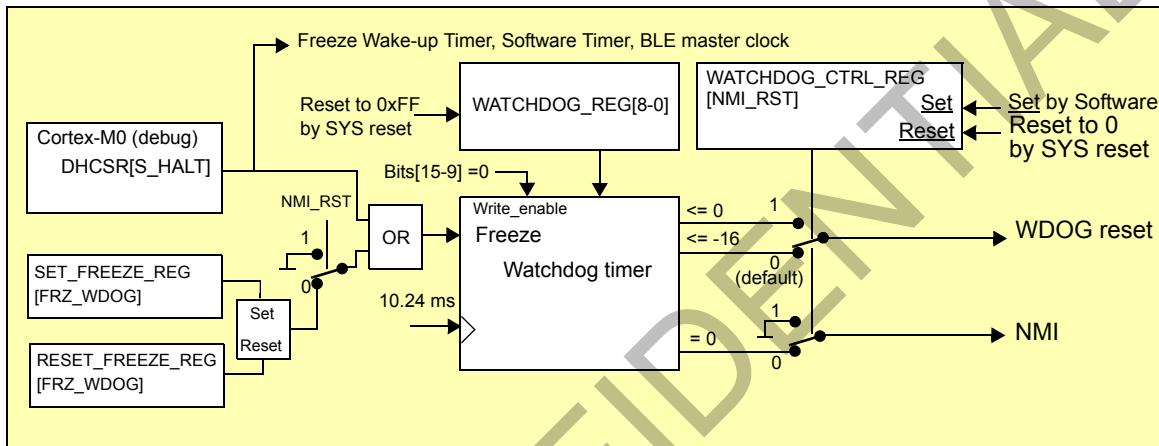


Figure 48: Watchdog Timer Block Diagram

The 8 bits watchdog timer is decremented by 1 every 10.24 ms. The timer value can be accessed through the WATCHDOG_REG register which is set to 255 (FF_{16}) at reset. This results in a maximum watchdog time-out of ~2.6 s. During write access the WATCHDOG_REG[WDOG_WEN] bits must be 0. This provides extra filtering for a software run-away writing all ones to the WATCHDOG_REG. If the watchdog counter reaches 0, the counter value will get a negative value by setting bit 8. The counter sequence becomes 1, 0, $1FF_{16}$ (-1), $1FE_{16}$ (-2), ... $1F0_{16}$ (-16).

If WATCHDOG_CTRL_REG[NMI_RST] = 0, the watchdog timer will generate an NMI if the watchdog timer reaches 0 and a WDOG reset if the counter becomes less or equal to -16 ($1F0_{16}$). The NMI handler must write any value > -16 to the WATCHDOG_REG to prevent the generation of a WDOG reset at counter value -16 after $16 \times 10.24 = 163.8$ ms.

If WATCHDOG_CTRL_REG[NMI_RST] = 1, the watchdog timer generates a WDOG reset if the timer becomes less or equal than 0.

The WDOG reset is one of the SYS (system) reset sources and resets the whole device, including setting the WATCHDOG_REG register to 255, except for the RST pin, the Power On reset, the HW reset and the DBG (debug module) reset. Since the HW reset is not triggered, the SYS_CTRL_REG[REMAP_ADR0] bits will retain their value and the Cortex-M0 will start executing again from the current selected memory at

address zero. Refer to the “POR/Reset” chapter for an overview of the complete reset circuit and conditions.

For debugging purposes, the Cortex-M0 Debug module can always freeze the watchdog by setting the DHCSR[DBGKEY | C_HALT | C_DEBUGEN] control bits (reflected by the status bit S_HALT). This is automatically done by the debug tool, e.g. during step-by-step debugging. Note that this bit also freezes the Wake-up Timer, the Software Timer and the BLE master clock. For additional information also see the DEBUG_REG[DEBUGS_FREEZE_EN] mask register. The C_DEBUGEN bit is not accessible by the user software to prevent freezing the watchdog.

In addition to the S_HALT bit, the watchdog timer can also be frozen if NMI_RST=0 and SET_FREEZE_REG[FRZ_WDOG] is set to ‘1’. The watchdog timer resumes counting when RESET_FREEZE_REG[FRZ_WDOG] is set to ‘1’. The WATCHDOG_CTRL_REG[NMI_RST] bit can only be set by software and will only be reset on a SYS reset. Note that if the system is not remapped, i.e. SysRAM is at address 0x20000000, then a watchdog fire will trigger the BootROM code to be executed again.

20 Keyboard Controller

The Keyboard controller can be used for debouncing the incoming GPIO signals when implementing a keyboard scanning engine. It generates an interrupt to the CPU (KEYBR_IRQ).

In parallel, five extra interrupt lines can be triggered by a state change on 32 selectable GPIOs (GPIOx_IRQ).

The block diagram of the Keyboard controller is presented in Figure 49.

Features

- Monitors any of the 32 available GPIOs (12 in the WLCSP package, 22 in the QFN40 and 32 in the QFN48)
- Generates a keyboard interrupt on key press or key release
- Implements debouncing time from 0 up to 63 ms
- Supports five separate interrupt generation lines from GPIO toggling

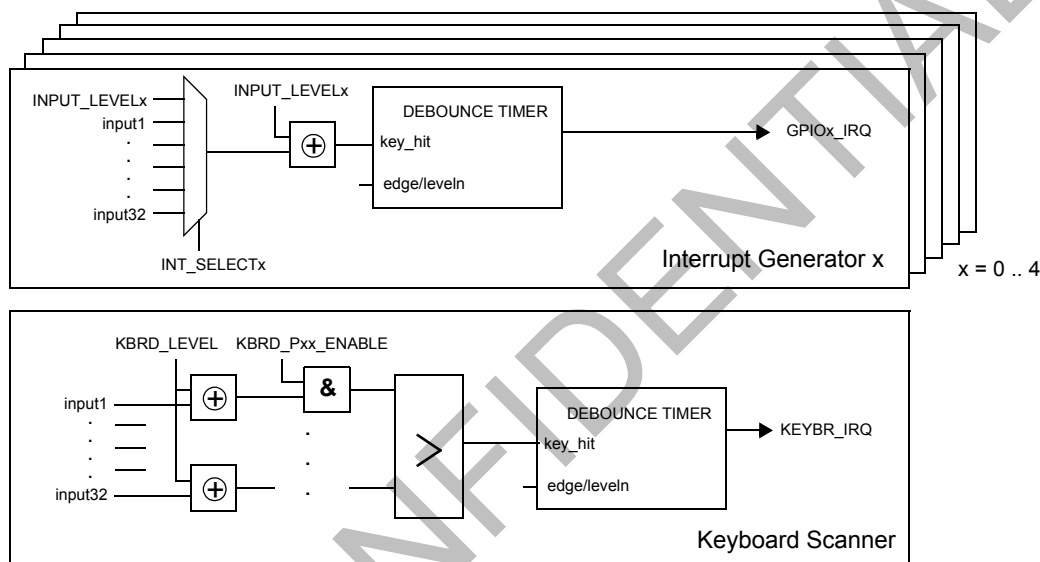


Figure 49: Keyboard Controller Block Diagram

20.1 KEYBOARD SCANNER

A HIGH-to-LOW transition on one of the port inputs, while $KBRD_IRQ_IN_SEL0_REG[KBRD_LEVEL] = 0$ and $KBRD_IRQ_IN_SELx_REG[KBRD_Pyy_EN] = 1$ sets internal signal “key_hit” to 1. This signal triggers the keyboard interface state machine as shown in Figure 50. The debounce timer is loaded with value $GPIO_DEBOUNCE_REG[DEB_VALUE]$. The timer counts down every 1 ms. When the timer reaches 0 and the “key_hit” signal is still ‘1’, the timer is loaded with $KBRD_IRQ_IN_SEL0_REG[KEY_REPEAT]$, generating a repeating sequence of interrupts every time the timer reaches 0.

When the key is released ($key_hit = 0$) and bit $KBRD_REL$ (key release) is set to ‘1’, a new debounce sequence is started and a $KEYBR_IRQ$ interrupt is generated after the debounce time.

The debounce timer can be disabled with $GPIO_DEBOUNCE_REG[DEB_ENABLE_KBRD] = 0$. The key repeat function can be disabled by setting KEY_REPEAT to ‘0’.

The level for generating an interrupt is programmable via bit $KBRD_IRQ_IN_SEL0_REG[KBRD_LEVEL]$. The key release function can be disabled by setting bit $KBRD_IRQ_IN_SEL0_REG[KBRD_REL]$ to ‘0’. The

inputs for the keyboard interface can be selected by setting the corresponding bits $KBRD_IRQ_IN_SEL0_REG[KBRD_Pxx_EN]$ to ‘1’.

The keyboard interrupt service routine can distinguish which input has caused the interrupt by reading the Px_DATA_REG registers.

20.2 GPIO INTERRUPT GENERATOR

Five identical GPIO interrupt generators support the generation of up to five interrupts ($GPIO0_IRQ$ to $GPIO_4_IRQ$). One of the GPIO inputs can be selected to generate an interrupt by programming the corresponding $GPIO_IRQx_IN_SEL_REG$ register. The input level can be selected by $GPIO_INT_LEVEL_CTRL_REG[INPUT_LEVELx]$.

A LOW-to-HIGH level transition on one of the port inputs, while bit $INPUT_LEVELx = 0$, sets internal signal “key_hit” to ‘1’. This signal triggers the GPIO Interrupt Generator state machine as shown in Figure 50. The debounce timer is loaded with value $GPIO_DEBOUNCE_REG[DEB_VALUE]$. The timer counts down every 1 ms. If the timer reaches 0 and the “key_hit” signal is still ‘1’, an interrupt will be generated. The debounce timer for each interrupt can be disabled with $GPIO_DEBOUNCE_REG[DEB_ENABLEx]$.

The interrupt flag will remain set until it is reset by writing to the corresponding bit in the GPIO_RESET_IRQ_REG register. When the GPIO interrupt is edge sensitive, selected with bit GPIO_INT_LEVEL_CTRL_REG[EDGE_LEVELNx], the state machine will progress to state WAIT_FOR_RELEASE when the interrupt is reset. It will progress to the IDLE state only after detecting the non-active edge.

For detecting both signal edges the edge polarity INPUT_LEVELx must be inverted in the WAIT_FOR_RELEASE state. This will result in "key_hit" = 0 and will advance the state machine to the Idle state, allowing detection of the next inverted edge.

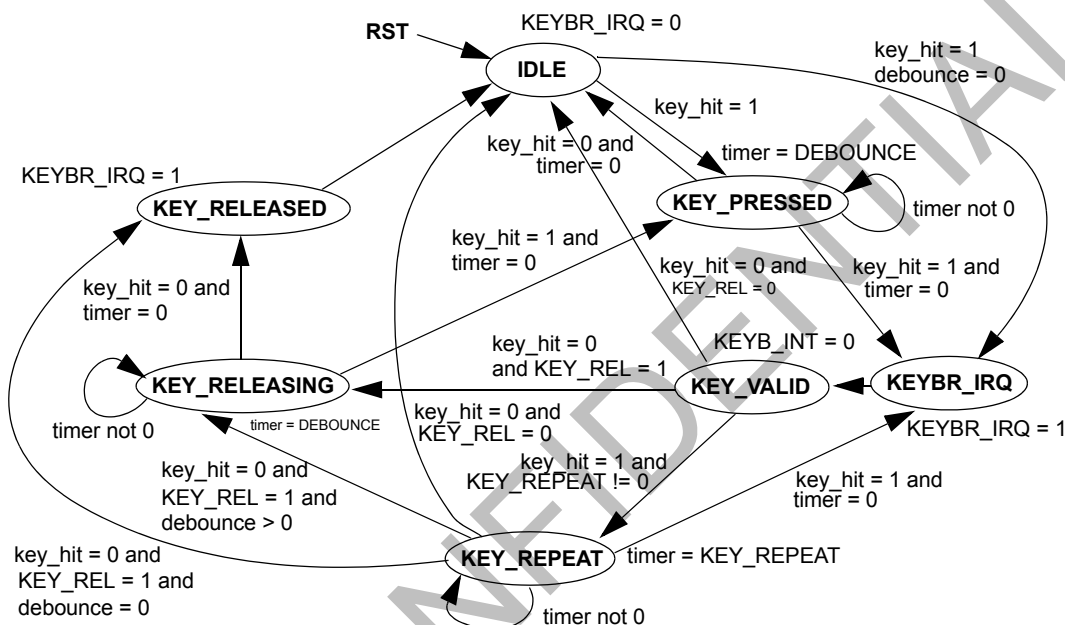


Figure 50: Keyboard Scanner State Machine

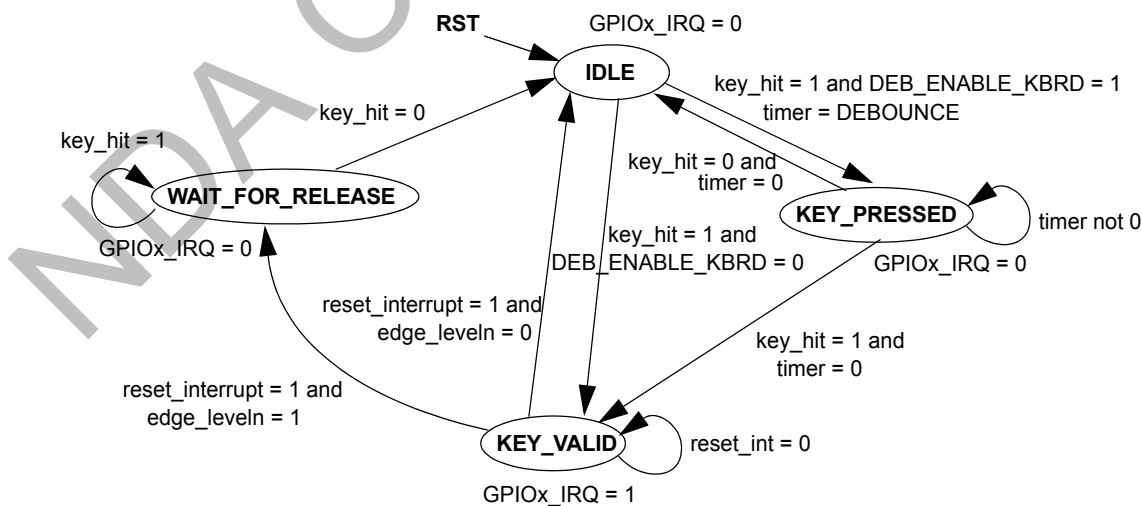


Figure 51: GPIO Interrupt Generator State Machine

21 Input/Output Ports

The DA14582 has software-configurable I/O pin assignment, organized into ports Port 0, Port1, Port2 and Port 3. Port 2 is only available at the QFN40 package while ports 2 and 3 are available at the QFN48 package.

Features

- Port 0: 8 pins, Port 1: 6 pins (including SW_CLK and SWDIO), Port 2: 10 pins, Port 3: 8 pins

- Fully programmable pin assignment
- Selectable 25 kΩ pull-up, pull-down resistors per pin
- Pull-up voltage either VBAT3V (BUCK mode) or VBAT1V (BOOST mode) configurable per pin
- Fixed assignment for analog pin ADC[3:0]
- Pins retain their last state when system enters the Extended or Deep Sleep mode.

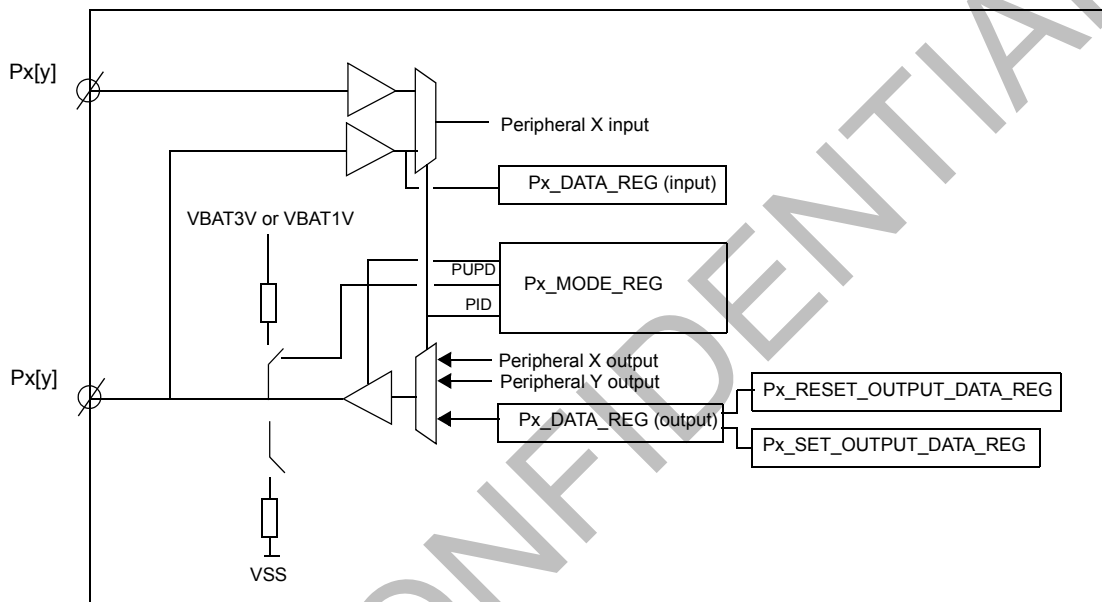


Figure 52: Port P0, P1, P2 and P3 with Programmable Pin Assignment

21.1 PROGRAMMABLE PIN ASSIGNMENT

The Programmable Pin Assignment (PPA) provides a multiplexing function to the I/Os of on-chip peripherals. Any peripheral input or output signal can be freely mapped to any I/O port bit by setting:

Pxy_MODE_REG[4-0]:

0x00 to 0x16: Peripheral IO ID (PID)

Refer to the Px_MODE_REGS for an overview of the available PIDs. Analog ADC has fixed pin assignment in order to limit interference with the digital domain. The SWD interface (JTAG) is mapped on P1_4 and P1_5.

Priority

The firmware has the possibility to assign the same peripheral output to more than one pin. It is the responsibility of the user to make a unique assignment.

In case more than one input signal is assigned to a peripheral input, the left most pin in the lowest port pin number has priority. (e.g P00_MODE_REG has priority over P01_MODE_REG)

Direction Control

The port direction is controlled by setting:

Pxy_MODE_REG[9-8]

- 00 = Input, no resistors selected
- 01 = Input, pull-up selected
- 10 = Input, Pull-down selected
- 11 = Output, no resistors selected

In output mode and analog mode the pull-up/down resistors are automatically disabled.

21.2 GENERAL PURPOSE PORT REGISTERS

The general purpose ports are selected with PID=0. The port function is accessible through registers:

- Px_DATA_REG: Port data input/output register
- Px_SET_OUTPUT_DATA_REG: Port set output register
- Px_RESET_OUTPUT_DATA_REG: Port reset output register

21.2.1 Port Data Register

The registers input Px_DATA_REG and output Px_DATA_REG are mapped on the same address.

The data input register (Px_DATA_REG) is a read-only register that returns the current state on each port pin even if the output direction is selected, regardless of the programmed PID, unless the analog function is selected (in this case it reads 0). The ARM CPU can read this register at any time even when the pin is configured as an output.

The data output register (Px_DATA_REG) holds the data to be driven on the output port pins. In this configuration, writing to the register changes the output value.

21.2.2 Port Set Data Output Register

Writing a 1 in the set data output register (Px_SET_OUTPUT_DATA_REG) sets the corresponding output pin. Writing a 0 is ignored.

21.2.3 Port Reset Data Output Register

Writing a 1 in the reset data output register (Px_RESET_OUTPUT_DATA_REG) resets the corresponding output pin. Writing a 0 is ignored.

21.3 FIXED ASSIGNMENT FUNCTIONALITY

There are certain signals that have a fixed mapping on specific general purpose IOs. This assignment is illustrated in the following table:

Table 21: Fixed Assignment of Specific Signals

| GPIO | SWD | QUAD DEC | ADC (Note 16) |
|------|--------|-------------------|---------------|
| P0_0 | | CHY_A/CHX_A/CHZ_A | ADC_0 |
| P0_1 | | CHY_B/CHX_B/CHZ_B | ADC_1 |
| P0_2 | | CHY_A/CHX_A/CHZ_A | ADC_2 |
| P0_3 | | CHY_B/CHX_B/CHZ_B | ADC_3 |
| P0_4 | | CHY_A/CHX_A/CHZ_A | |
| P0_5 | | CHY_B/CHX_B/CHZ_B | |
| P0_6 | | CHY_A/CHX_A/CHZ_A | |
| P0_7 | | CHY_B/CHX_B/CHZ_B | |
| P1_0 | | CHY_A/CHX_A/CHZ_A | |
| P1_1 | | CHY_B/CHX_B/CHZ_B | |
| P1_2 | | CHY_A/CHX_A/CHZ_A | |
| P1_3 | | CHY_B/CHX_B/CHZ_B | |
| P1_4 | SW_CLK | CHY_A/CHX_A/CHZ_A | |
| P1_5 | SWDIO | CHY_B/CHX_B/CHZ_B | |
| P2_0 | | CHY_A/CHX_A/CHZ_A | |
| P2_9 | | CHY_B/CHX_B/CHZ_B | |

Note 16: The ADC case can be selected by the PID bit field on the respective P port. However, the mapping of the Quad Decoder signals on the respective pins, is overruled by the QDEC_CTRL_REG[CHx_PORT_SEL] register. Furthermore, the SWD signals mapping is defined by SYS_CTRL_REG[DEBUGGER_ENABLE]. However, these signals are mapped on the ports by default.

22 General Purpose ADC

The DA14582 is equipped with a high-speed ultra low power 10-bit general purpose Analog-to-Digital Converter (GPADC). It can operate in unipolar (single ended) mode as well as in bipolar (differential) mode. The ADC has its own voltage regulator (LDO) of 1.2 V, which represents the full scale reference voltage.

Features

- 10-bit dynamic ADC with 65 ns conversion time
- Maximum sampling rate 3.3 Msample/s

- Ultra low power (5 μ A typical supply current at 100 ksample/s)
- Single-ended as well as differential input with two input scales
- Four single-ended or two differential external input channels
- Battery monitoring function
- Chopper function
- Offset and zero scale adjust
- Common-mode input level adjust

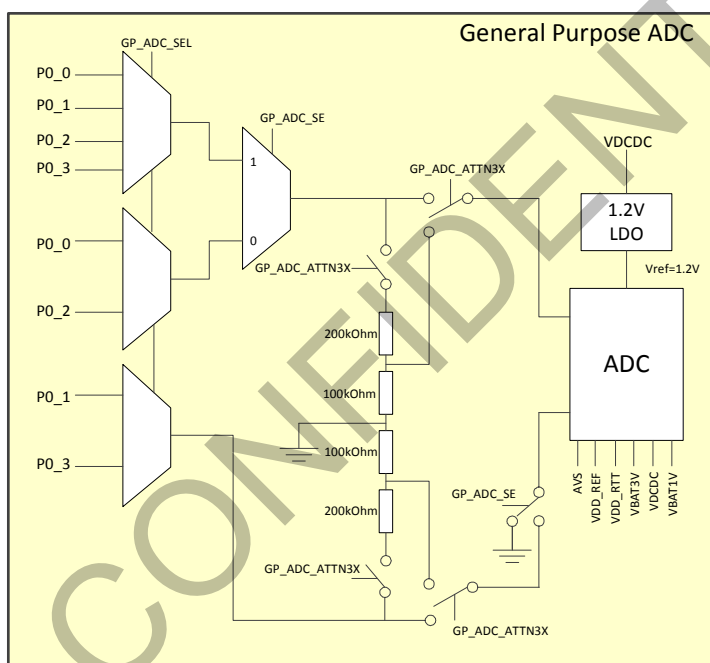


Figure 53: Block Diagram of the General Purpose ADC

22.1 INPUT CHANNELS AND INPUT SCALE

The DA14582 has a multiplexer between the ADC and four specific GPIO ports (P0_0 to P0_3). Furthermore, the ADC can also be used to monitor the battery voltage and several internal voltages of the system (see GP_ADC_CTRL_REG).

Single-ended or differential operation is selected via bit GP_ADC_CTRL_REG[GP_ADC_SE]. In differential mode the voltage difference between two GPIO input ports will be converted. Via bit GP_ADC_CTRL2_REG[GP_ADC_ATT3X] the input scale can be enlarged by a factor of three, as summarized in Table 22.

Table 22: GPADC Input Channels and Voltage Scale

| GP_ADC_ATT3X | GP_ADC_SE | Input Channels | Input Scale | Input Limits |
|--------------|-----------|----------------------------|------------------|--------------------|
| 0 | 1 | P0_0, P0_1, P0_2, P0_3 | 0 V to +1.2 V | -0.1 V to +1.3 V |
| 0 | 0 | [P0_0, P0_1], [P0_2, P0_3] | -1.2 V to +1.2 V | -1.3 V to +1.3 V |
| 1 | 1 | P0_0, P0_1, P0_2, P0_3 | 0 V to +3.6 V | -0.1 V to +3.45 V |
| 1 | 0 | [P0_0, P0_1], [P0_2, P0_3] | -3.6 V to +3.6 V | -3.45 V to +3.45 V |

22.2 STARTING THE ADC AND SAMPLING RATE

The GPADC is a dynamic ADC and consumes no static

power, except for the LDO which consumes less than 5 μ A.

Enabling/disabling of the ADC is triggered by configuring bit `GP_ADC_CTRL_REG[GP_ADC_LDO_EN]`. After enabling the LDO, a settling time of 20 μ s is required before an AD-conversion can be started.

Each conversion has two phases: the sampling phase and the conversion phase. When bit `GP_ADC_CTRL_REG[GP_ADC_EN]` is set to '1', the ADC continuously tracks (samples) the selected input voltage. Writing a '1' at bit `GP_ADC_CTRL_REG[GP_ADC_START]` ends the sampling phase and triggers the conversion phase. When the conversion is ready, the ADC resets bit `GP_ADC_START` to '0' and returns to the sampling phase.

The conversion itself is fast and takes approximately one clock cycle of 16 MHz, though the data handling will require several additional clock cycles, depending on the software code style. The fastest code can handle the data in four clock cycles of 16 MHz, resulting to a highest sampling rate of $16 \text{ MHz}/5 = 3.3 \text{ Msample/s}$.

At full speed the ADC consumes approximately 50 μ A. If the data rate is less than 100 ksample/s, the current consumption will be in the range of 5 μ A.

22.3 NON-IDEAL EFFECTS

Besides Differential Non-Linearity (DNL) and Integral Non-Linearity (INL), each ADC has a gain error (linear) and an offset error (linear). The gain error of the GPADC slightly reduces the effective input scale (up to 50 mV). The offset error causes the effective input scale to become non-centred. The offset error of the GPADC is less than 20 mV and can be reduced by chopping or by offset calibration.

The ADC result will also include some noise. If the input signal itself is noise free (inductive effects included), the average noise level will be ± 1 LSB. Taking more samples and calculating the average value will reduce the noise and increase the resolution.

With a 'perfect' input signal (e.g. if a filter capacitor is placed close to the input pin) most of the noise comes from the low-power voltage regulator (LDO) of the ADC. Since the DA14582 is targeted for ultra-compact applications, there is no pin available to add a capacitor at this voltage regulator output.

The dynamic current of the ADC causes extra noise at the regulator output. This noise can be reduced by setting bits `GP_ADC_CTRL2_REG[GP_ADC_I20U]` and `GP_ADC_CTRL2_REG[GP_ADC_IDYN]` to '1'. Bit `GP_ADC_I20U` enables a constant 20 μ A load current at the regulator output so that the current will not drop

to zero. Bit `GP_ADC_IDYN` enables a 10 μ A load current during sampling phase so that the load current during sampling and conversion phase becomes approximately the same.

22.4 CHOPPING

Chopping is a technique to cancel offset by taking two samples with opposite signal polarity. This method also smooths out other non-ideal effects and is recommended for DC and slowly changing signals.

Chopping is enabled by setting bit `GP_ADC_CTRL_REG[GP_ADC_CHOP]` to '1'.

The mid-scale value of the ADC is the 'natural' zero point of the ADC (ADC result = 511.5 = 1FF or 200 Hex = 01.1111.1111 or 10.0000.0000 Bin). Ideally this corresponds to $V_i = 1.2 \text{ V}/2 = 0.6 \text{ V}$ in single-ended mode and $V_i = 0.0 \text{ V}$ in differential mode.

If bit `GP_ADC_CTRL2_REG[GP_ADC_ATTN3X]` is set to '1', the zero point is 3 times higher (1.8 V single-ended and 0.0 V differential).

With bit `GP_ADC_CTRL_REG[GP_ADC_MUTE]`, the ADC input is switched to the centre scale input level, so the ADC result ideally is 511.5. If instead a value of 515 is observed, the output offset is +3.5 (`adc_off_p = 3.5`).

With bit `GP_ADC_CTRL_REG[GP_ADC_SIGN]` the sign of the ADC input and output is changed. Two sign changes have no effect on the signal path, though the sign of the ADC offset will change.

If `adc_off_p = 3.5` the ADC result with opposite `GP_ADC_SIGN` will be 508. The sum of these equals $515 + 508 = 1023$. This is the mid-scale value of an 11-bit ADC, so one extra bit due to the over-sampling by a factor of two.

The LSB of this 11-bit word should be ignored if a 10-bit word is preferred. In that case the result is 511.5, so the actual output value will be 511 or 512.

22.5 OFFSET CALIBRATION

A relative high offset caused by a very small dynamic comparator (up to 20 mV, so approximately 20 LSB).

This offset can be cancelled with the chopping function, but it still causes unwanted saturation effects at zero scale or full scale. With the `GP_ADC_OFFP` and `GP_ADC_OFFN` registers the offset can be compensated in the ADC network itself.

To calibrate the ADC follow the steps in [Table 23](#).

Table 23: GPADC Calibration Procedure for Single-Ended and Differential Modes

| Step | Single-Ended Mode (<code>GP_ADC_SE = 1</code>) | Differential Mode (<code>GP_ADC_SE = 0</code>) |
|------|---|---|
| 1 | Set <code>GP_ADC_OFFP = GP_ADC_OFFN = 0x200</code> ; <code>GP_ADC_MUTE = 0x1</code> ; <code>GP_ADC_SIGN = 0x0</code> | Set <code>GP_ADC_OFFP = GP_ADC_OFFN = 0x200</code> ; <code>GP_ADC_MUTE = 0x1</code> ; <code>GP_ADC_SIGN = 0x0</code> |
| 2 | Start conversion | Start conversion |
| 3 | <code>adc_off_p = GP_ADC_RESULT - 0x200</code> | <code>adc_off_p = GP_ADC_RESULT - 0x200</code> |

Table 23: GPADC Calibration Procedure for Single-Ended and Differential Modes

| Step | Single-Ended Mode (GP_ADC_SE = 1) | Differential Mode (GP_ADC_SE = 0) |
|------|--|--|
| 4 | Set GP_ADC_SIGN = 0x1 | Set GP_ADC_SIGN = 0x1 |
| 5 | Start conversion | Start conversion |
| 6 | adc_off_n = GP_ADC_RESULT - 0x200 | adc_off_n = GP_ADC_RESULT - 0x200 |
| 7 | GP_ADC_OFFP = 0x200 - 2*adc_off_p GP_ADC_OFFN = 0x200 - 2*adc_off_n | GP_ADC_OFFP = 0x200 - adc_off_p GP_ADC_OFFN = 0x200 - adc_off_n |

Note: The average of GP_ADC_OFFP and GP_ADC_OFFN should be 0x200 (with a margin of 20 LSB).

It is recommended to implement the above calibration routine during the initialization phase of the DA14582. To verify the calibration results, check whether the GP_ADC_RESULT value is close to 0x200 while bit GP_ADC_MUTE = 1.

22.6 ZERO-SCALE ADJUSTMENT

The GP_ADC_OFFP and GP_ADC_OFFN registers can also be used to set the zero-scale or full-scale input level at a certain target value. For instance, they can be used to calibrate GP_ADC_RESULT to 0x000 at an input voltage of exactly 0.0 V, or to calibrate the zero scale of a sensor.

22.7 COMMON MODE ADJUSTMENT

The common mode level of the differential signal must be 0.6 V (or 1.8 V with GP_ADC_ATT3X = 1). If the common mode input level of 0.6 V cannot be achieved, the common mode level of the GP_ADC can be adjusted (the GP_ADC can tolerate a common mode margin up to 50 mV) according to Table 24.

Table 24: Common Mode Adjustment

| CM Voltage (V _{cmm}) | GP_ADC_OFFP = GP_ADC_OFFN |
|--------------------------------|---------------------------|
| 0.3 V | 0x300 |
| 0.6 V | 0x200 |
| 0.9 V | 0x100 |

Any other common mode level between 0.0 V and 1.2 V can be calculated from the table above. Offset calibration can be combined with common mode adjustment by replacing the 0x200 value in the offset calibration routine by the value required to get the appropriate common mode level.

Note: The input voltage limits for the ADC in differential mode are: -1.3 V to +1.3 V (for GP_ADC_ATT3X = 0, see Table 22). The differential input range of the ADC is: -1.2 V < V[P0_0,P0_1] < +1.2 V. Therefore, if V_{cmm} < 0.5 V or V_{cmm} > 0.7 V, the input can no longer cover the whole ADC range.

22.8 INPUT IMPEDANCE, INDUCTANCE AND INPUT SETTLING

The GPADC has no input buffer stage. During sampling phase a capacitor of 0.2 pF is switched to the input line. The precharge of this capacitor is at mid-

scale level so the input impedance is infinite.

At 100 ksample/s, zero or full-scale single-ended input signal, this sampling capacitor will load the input with:
 $I_{LOAD} = V * C * f_S = \pm 0.6 V * 0.2 pF * 100 kHz = \pm 12 nA$
 (differential: $\pm 1.2 V * 0.2 pF * 100 kHz = \pm 24 nA$ at both pins).

During sampling phase a certain settling time is required. A 10-bit accuracy requires at least 7 time constants of the output impedance of the input signal source and the 0.2 pF sampling capacitor. The conversion time is approximately one clock cycle of 16 MHz (62.5 ns).

$$7 * R_{OUT} * 0.2 pF - 62.5 ns < 1/f_S$$

$$\Rightarrow R_{OUT} < (1 + 62.5 ns * f_S) / (7 * 0.2 pF * f_S)$$

Examples:

$$R_{OUT} < 7.2 M\Omega \text{ at } f_S = 100 kHz$$

$$R_{OUT} < 760 k\Omega \text{ at } f_S = 1 MHz$$

The inductance from the signal source to the ADC input pin must be very small. Otherwise, filter capacitors are required from the input pins to ground (differential mode: from pin to pin).

To observe the noise level of the ADC and the voltage regulator, bit GP_ADC_CTRL_REG[GP_ADC_MUTE] must be set to '1'. The noise should be less than ± 1 LSB on average, with occasionally a ± 2 LSB peak value. If a higher noise level is observed on the input channel(s), applying filter capacitor(s) will reduce the noise.

The 3x input attenuator is realized with a resistor divider network. When bit GP_ADC_CTRL_REG2[GP_ADC_ATT3X] is set to '1', the input impedance of the selected ADC input channel becomes 300 k Ω (typical) instead of infinite. In addition, the resistor divider network will require more settling time in the sampling phase. The general guideline with bit GP_ADC_ATT3X = 1 is: select the input channel, then wait 1 μs (16 clock cycles) before starting the conversion. Only the required sampling time is affected by the attenuator, the conversion time remains approximately one clock cycle of 16 MHz (62.5 ns).

Note: Selecting the battery measurement channel automatically activates the 3x input attenuator (bit GP_ADC_ATT3X = 1). Therefore the 1 μs waiting time also applies when measuring the battery voltage, otherwise the resulting V_{bat} level will be too low.

22.9 DELAY COUNTER

The GPADC has a delay counter that can be used to add delays to several ADC control signals. A delay of up to 32 μ s can be added for the bits GP_ADC_LDO_EN, GP_ADC_START and GP_ADC_EN via registers GP_ADC_DELAY_REG and GP_ADC_DELAY2_REG.

The reset values of these two registers are the recommended values for a correct start-up, since it is not allowed to activate all signals at once.

To make use of the delay counter for a certain signal, the corresponding bit has to be set in register GP_ADC_CTRL_REG and the delay counter must be enabled via bit GP_ADC_DELAY_EN in register GP_ADC_CTRL2_REG. The delay counter starts counting when the GP_ADC_START bit is programmed while the GP_ADC_DELAY_EN bit is set. The counter is stopped after the conversion is finished.

The delay counter must be reset before reuse, which is typically only required after the LDO was disabled. Bit GP_ADC_DELAY_EN must be made zero to reset the counter. It is recommended to check that this bit is zero before (re)activating it.

NDA CONFIDENTIAL

23 Power Management

The DA14582 has a complete power management function integrated with a Buck mode DC-DC converter and separate LDOs for the different power domains of the system. The system diagram of the analog power block is presented in Figure 54.

Features

- On-chip LDOs, without external capacitors

- Synchronous DC-DC converter configured as Buck (step-down) converter for increased efficiency when running from a Lithium coin-cell or 2 Alkaline batteries down to 2.35 V.
- Battery voltage measurement ADC (multiplexed input from general purpose ADC)
- Use of small external components (2.2 μ H inductor and 1 μ F capacitor)

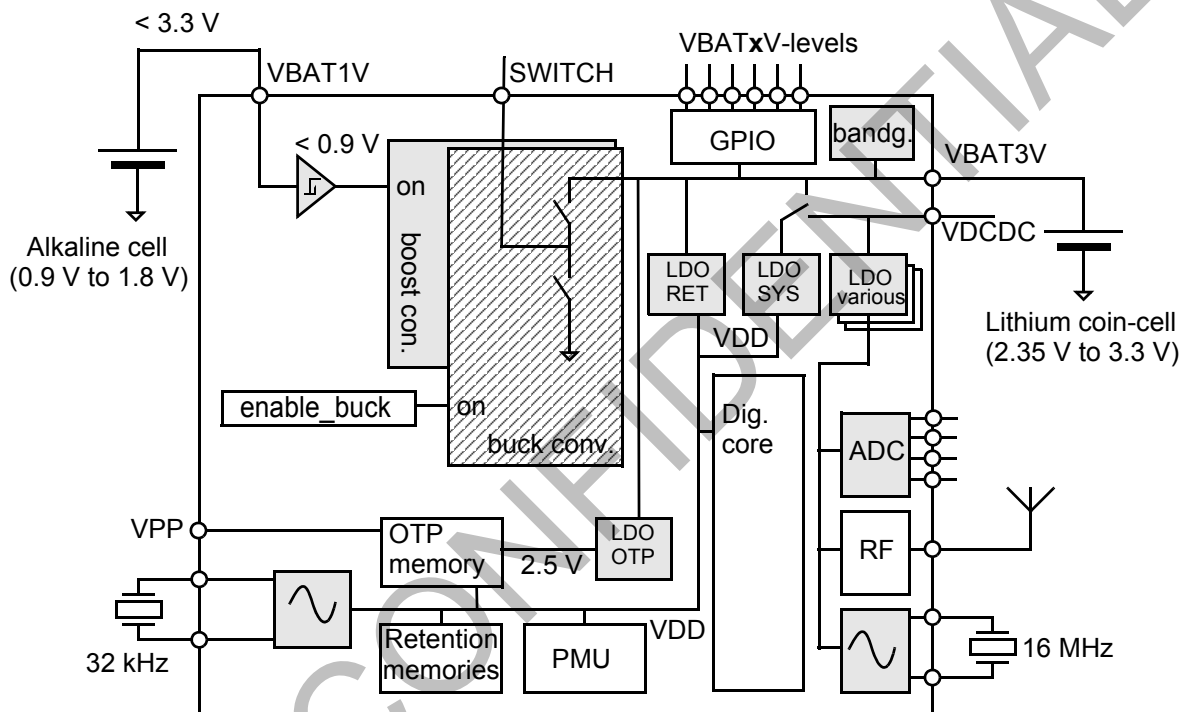


Figure 54: Block Diagram of the Analog Power Block and Internal Interconnections

The Power Block contains a DC-DC converter operating as a Step-Down converter. The converter provides power to four LDO groups in the system:

1. LDO RET: This is the LDO providing power to the Retention domain (PD_AON). It powers the Retention RAMs and the digital part which is always on.
2. LDO OTP: This is the LDO powering the OTP macro cell. This is the reason for using the step-up DC-DC converter when running from an Alkaline battery.
3. LDO SYS: This is the LDO providing the system with the actual VDD power required for the digital part to operate.
4. LDO (various): This a group of LDOs used for the elaborate control of the powering up/down of the Radio, the GP ADC and the XTAL16M oscillator.

The Power Block of the DA14582 only supports the use of Lithium coin cells (2.35 V to 3.3 V) as external battery. The connection diagram is shown in Figure 55.

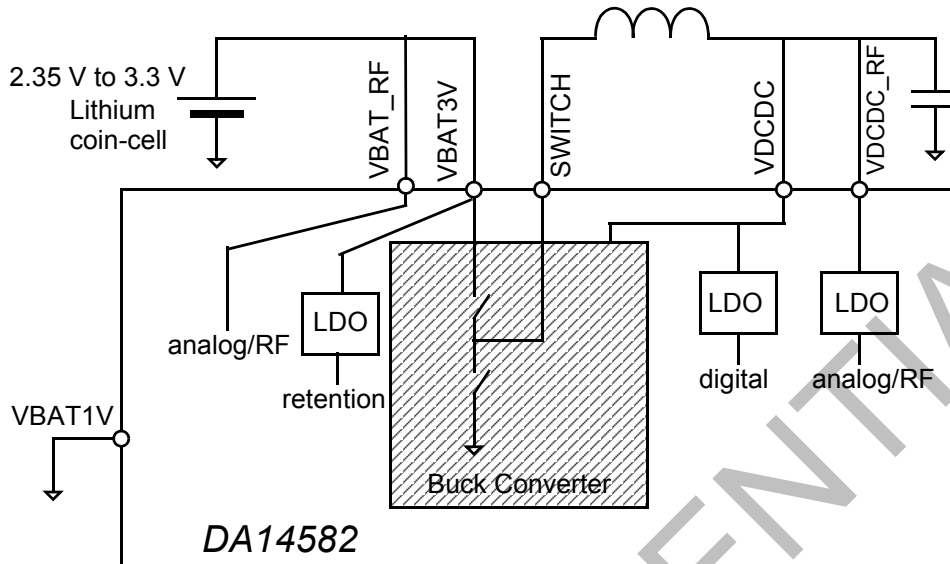


Figure 55: Supply Overview, Coin-Cell Application

The voltage range and efficiency of the Buck mode DC-DC converter is shown in the following figure, assuming a 10 mA constant load.

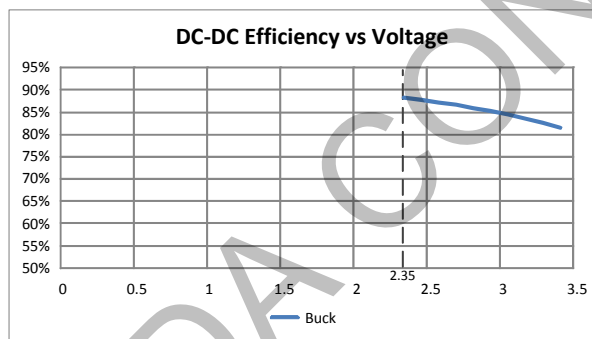


Figure 56: DC-DC Efficiency in Buck Mode at Various Voltage Levels

The X axis represents the supply voltage. The Buck mode DC-DC converter operates correctly with voltages in the range of 2.35 V to 3.3 V.

Note: The system should never be cold booted when the supply voltage is less than 2.5 V. A manual power up with a power supply less than 2.5 V in buck mode might create instability.

24 CODEC with Analog Front-End

A bidirectional 14 bits Sigma-Delta CODEC with fully differential analog front-end for microphones and loudspeakers down to $28\ \Omega$. is connected internally via SPI to the BLE core.

The CODEC sample frequency up to 16 kHz is derived from an internal 46.08 MHz PLL that uses the 16 MHz BLE clock.

The internal SPI connection between the CODEC and the BLE core requires a dedicated pin assignment at the BLE core while the CODEC is operational.

Two general purpose I/O pins provide control of external circuits.

Features

- 14 bits bidirectional Sigma-Delta CODEC
- Sample rate up to 16 kHz
- Fully differential analog front-end

- Programmable microphone gain from 0 dB up to +30 dB in steps of 2 dB
- Digital offset compensation
- Programmable loudspeaker attenuation from +5 dB to -9 dB in 8 steps of ~ 2 dB/step
- Loudspeaker impedance down to $28\ \Omega$
- Programmable sidetone between microphone and loudspeaker for speech applications
- Low noise microphone supply voltage
- SPI interface for control and data
- DMA controller for low overhead sample copying from SPI to Sigma-Delta CODEC
- Two general purpose GPIO pins
- Dedicated 1.8 V supply pin to support deep power-down with external LDO

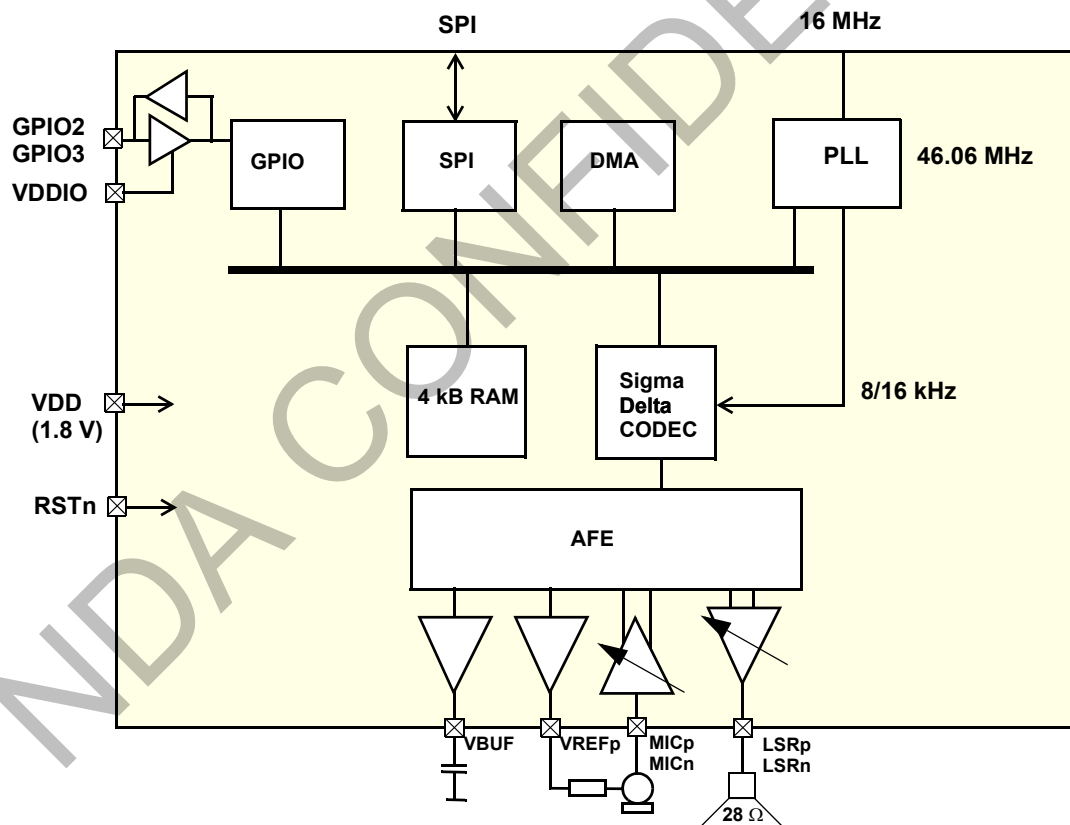


Figure 57: CODEC with AFE Block Diagram

24.1 ARCHITECTURE

The CODEC block contains of the following building blocks:

- SPI Interface with Command Interpreter**
 This interface is connected to dedicated I/O pins of the BLE core. The interface is used to program the CODEC and to transfer the voice samples to/from the BLE core. The command interpreter controls the system bus master which generates addresses to access shared RAM, DMA, CODEC, PLL and GPIO registers.
- Shared RAM**
 4 kB of RAM used for sample buffering.
- Clock Tree and Phase Locked Loop**
 The PLL generates the 46.08 MHz VCO frequency for the 2.304 MHz CODEC and locks to the 16 MHz BLE clock.
- DMA Controller**
 The DMA controller transfers CODEC samples to/from the shared RAM.
- GPIO**
 The general purpose bidirectional GPIO is accessed through the system bus. The I/O pins are supplied by VDDIO.
- Sigma-Delta CODEC**
 Converts digital samples to the analog frontend. It runs at an oversampling rate of 2.304 MHz and down-samples by a factor 144 to the sample rate of 16 kHz.
- Analog Front-End**
 Contains microphone and loudspeaker gain stages and provides a low-noise reference voltage to supply the microphone.

24.1.1 Pin Assignment CODEC to BLE Core

The CODEC is internally connected to the BLE core via a dedicated pin assignment as shown in [Table 25](#).

Table 25: Dedicated Pin Assignment for the CODEC

| DA14582 pin | CODEC port | BLE Core port |
|-----------------|---------------|----------------------|
| P2_1/ SPI_EN | SPI_EN (I) | P2_1/ SPI_EN (O) |
| Not available | SPI_CLK (I) | P3_1/ SPI_CLK (O) |
| P0_4/ SPI_DI | SPI_DI (I) | P0_4/ SPI_DO (O) |
| Not available | SPI_DO (O) | P3_2/ SPI_DI (I) |
| P0_5/ 16MCLK | CLK/16MHz (I) | P0_5/ 16MCLK (O) |

24.1.2 Signal Flow

[Figure 58](#) shows the typical data flow for a speech controlled remote control unit. For more details, refer to User Manual *UM-B-042*.

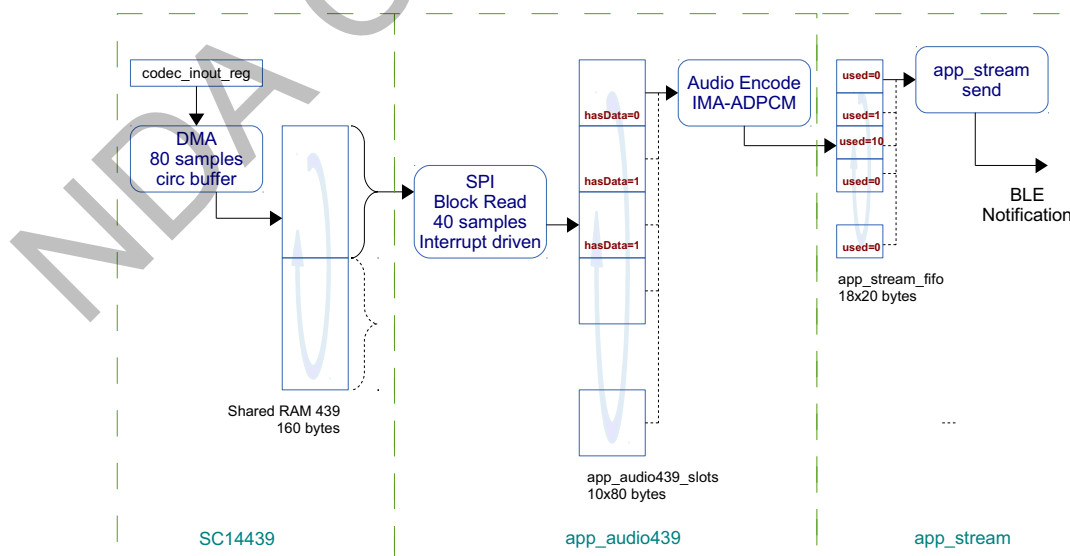


Figure 58: Data Flow for a Speech Controlled Remote Control Unit

24.2 SPI INTERFACE

The Serial Peripheral Interface SPI™ is a synchronous interface used to give commands to and to read status information from the CODEC. In addition, compressed data to/from shared RAM can be transferred over this bus. The serial interface can receive 16-bit words.

The SPI acts as bus master on the CODEC system bus to read or write to any address on the bus in single word or burst mode.

SPI™ is a trademarks of Motorola, Inc

Features

- 16-bit commands and data
- SPI Slave mode 0, 3 support
- SPI Clock speed up to CLK input frequency/2 and $F_{pll}/2 * PER_DIV$ if PLL on.
- Single word and burst mode

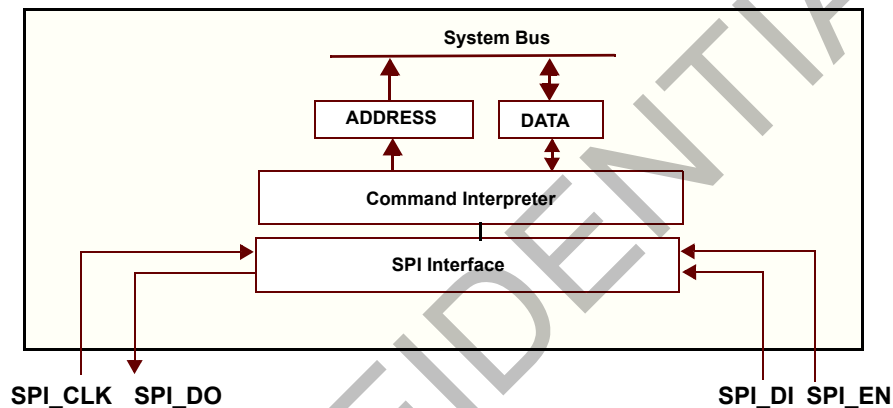


Figure 59: SPI Block Diagram

24.2.1 Architecture

The SPI block contains the following sub-blocks:

- SPI Interface
- SPI Command Interpreter
- Bus Master

24.2.2 SPI Interface

The CODEC SPI interface has the following signals and they are assigned to the DA14582 SPI port as shown in Table 25:

- SPI_CLK, clock input
- SPI_DI, data input
- SPI_DO, Data output, push-pull, Hi-Z if SPI_EN = 1
- SPI_EN, clock enable. The first 16-bit word after a high-to-low transition is interpreted as command and address. A low-to-high transition terminates the current read or write mode.

Refer to Figure 60 and Figure 61 for the SPI timing diagrams.

Timing Diagrams

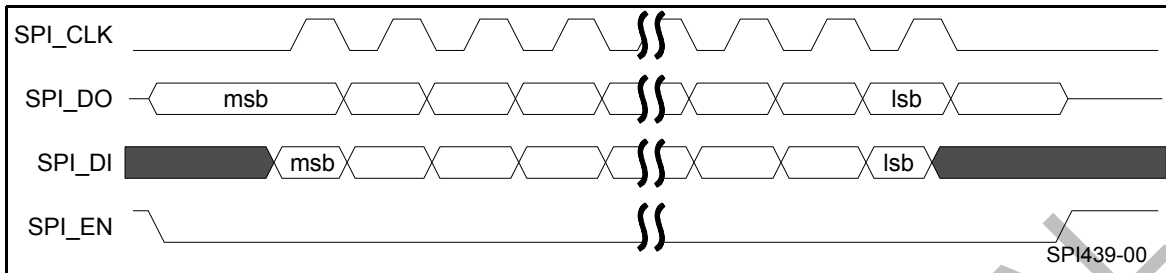


Figure 60: SPI Slave, Mode 0

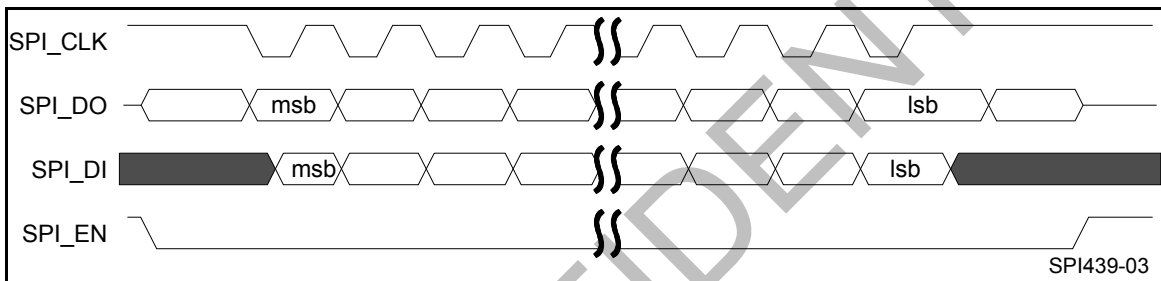


Figure 61: SPI Slave, Mode 3

24.2.3 SPI Command Interpreter

The following tables show the commands that can be given on the SPI interface to access on-chip registers and shared RAM.

Address = Any Address location. See Memory map Table 32 on page 84.

Length = number of words read or written. Max. value: $2^{13} - 1 = 8191$ words.

Table 26: SPI MEM_WR Command Format

| Signal | Word 1 | | Word 2 |
|--------|--------|---------------|------------|
| SPI_DI | MEM_WR | address[12-0] | data[15-0] |
| SPI_DO | 0 | | 0 |

Table 27: SPI MEM_RD Command Format

| Signal | Word 1 | | Word 2 | Word 3 |
|--------|--------|---------------|--------|------------------|
| SPI_DI | MEM_RD | address[12-0] | 0 | 0 or New Command |
| SPI_DO | 0 | | 0 | data[15-0] |

Table 28: SPI MEM_BWR Command Format

| Signal | Word 1 | | Word 2 | Word 3 | Word N |
|--------|---------|---------------|--------|------------|------------|
| SPI_DI | MEM_BWR | address[12-0] | Length | data[15-0] | data[15-0] |
| SPI_DO | 0 | | 0 | 0 | 0 |

Table 29: SPI MEM_BRD command format

| Signal | Word 1 | | Word 2 | Word 3 | Word N |
|--------|---------|---------------|--------|------------|------------------|
| SPI_DI | MEM_BRD | address[12-0] | Length | 0 | 0 or New Command |
| SPI_DO | 0 | | 0 | data[15-0] | data[15-0] |

Table 30: Command Description

| Opcode | Mnemonic | Description |
|--------|----------|---|
| 0x1 | MEM_WR | Memory write word to single address. After Writing command/address and data, any of the commands can be given, while SPI_EN is still LOW. |
| 0x2 | MEM_RD | Memory read word from single address. After Writing command/address and reading data, any of the commands can be given, while SPI_EN is still LOW. |
| 0x3 | MEM_BWR | Memory write words to successive addresses (block write). After Writing command/address and multiple data the command is stopped after "length" number of words or by setting SPI_EN to HIGH. Any of the commands can be given, while SPI_EN is still LOW. |
| 0x4 | MEM_BRD | Memory read words from successive addresses (block read). After Writing command/address and reading multiple data the command is stopped after "length" number of words or by setting SPI_EN to HIGH. Any of the commands can be given, while SPI_EN is still LOW. |

24.2.4 Bus Master

The system bus master generates addresses to access shared RAM, DMA, CODEC, PLL and GPIO registers. The CODEC memory map can be found in the following chapters.

The SPI bus master shares the system bus with the DMA controller channels in a non-blocking way. If the DMA is master on the bus, the SPI read or write commands are not disturbed.

24.3 CLOCK TREE

Figure 62 shows the CODEC clock tree. Table 31 shows the register setting settings for 16 MHz CLK input and 46.08 MHz PLL. The CODEC_DIV_REG[5-0] is programmable to get the constant 1.152 MHz for 8 kHz sample rate or 2.304 MHz for 16 kHz sample rate.

Table 31: Recommended Clock Settings (CLK_SOURCE = 0)

| CLK (MHz) | PLL_DIV_REG (VD/XD) | Fpll (MHz) | PER_DIV_REG[5-0] | CODEC_DIV_REG[5-0] for CODEC_CLK = 1.152 MHz (8 kHz) | CODEC_DIV_REG[5-0] for CODEC_CLK = 2.304 MHz (16 kHz) |
|-----------|---------------------|------------|------------------|--|---|
| | | | | PLL_BYPASSn | PLL_BYPASSn |
| | | | | 1 | 1 |
| 16 | 72/25 | 46.08 | 4 | 40 | 20 |

Note 17: The PER_DIV_REG must meet the following requirements: $SPI_CLK * 2 < F_{sys} / PER_DIV_REG \leq F_{sys} / 2$.

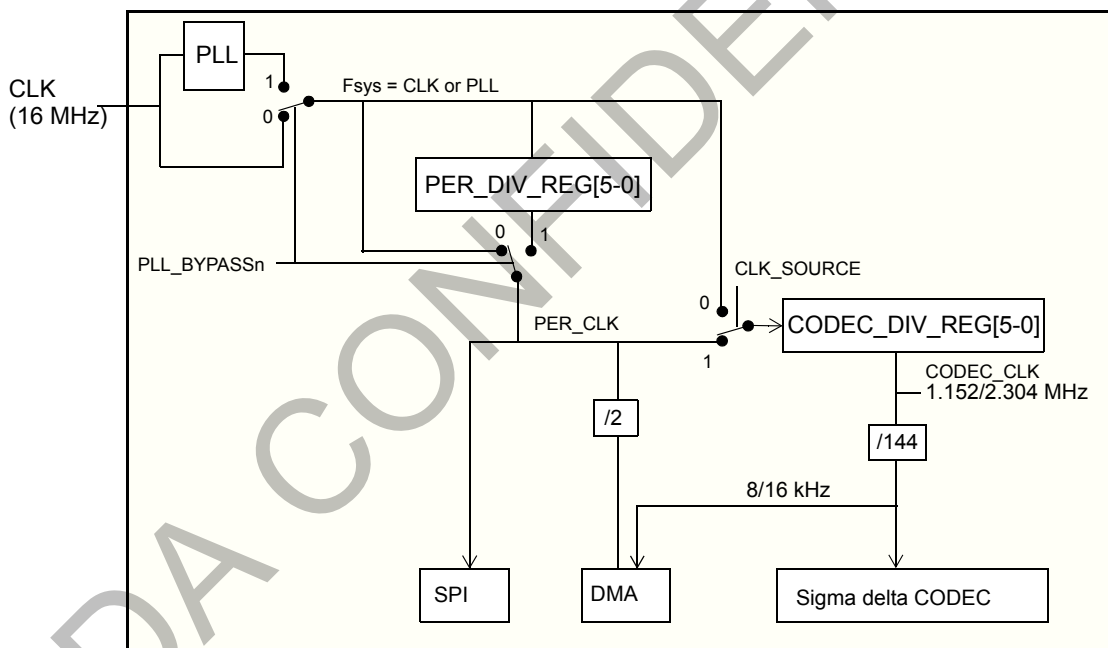


Figure 62: Clock Divider Overview

24.3.1 PLL Clock Switching Procedure

Figure 63 shows the PLL block diagram.

To enable the PLL:

1. The PLL starts up in the bypass mode with $\text{CLK_CTRL_REG}[3] = \text{PLL_BYPASSn}$ is '0'.
2. Next the PLL Xtal divider XD and VCO divider VD must be set in the PLL_DIV_REG to the correct values (typical value 0x6180) and the VCO and charge pump can be switched on with $\text{CLK_CTRL_REG}[4]=\text{VCO_ON}$ set to '1' and $\text{CLK_CTRL_REG}[1] = \text{CP_ON}$ set to '1'.

3. The software has to wait 0.5 ms before PLL_BYPASSn may be set to '1' to switch the PLL clock as the main system clock.
4. The host must wait at least 16 SPI clock cycles before a new SPI command may be entered.

To disable the PLL:

1. Switching back to XTALmode can be done at any time by setting PLL_BYPASSn to '0'.
2. In the next SPI command the VCO and Charge Pump may be switched off.

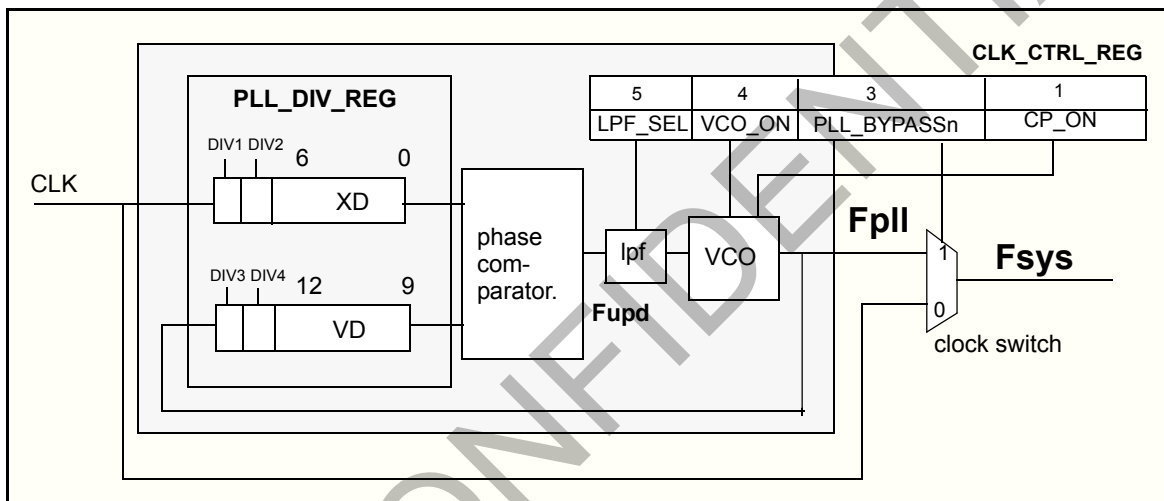


Figure 63: PLL Block Diagram

24.4 DMA CONTROLLER

The Direct Memory Access (DMA) controller has a single channel used for fast autonomous data transfers between the CODEC RAM and any peripherals on the system bus. A transfer can be triggered by the 8 kHz/16 kHz sample clock from the CODEC.

Features

- Indirect addressing mode
 - Peripheral to Memory
 - Memory to Peripheral
 - Memory to Memory
- Manual start or 8/16 kHz triggered operation

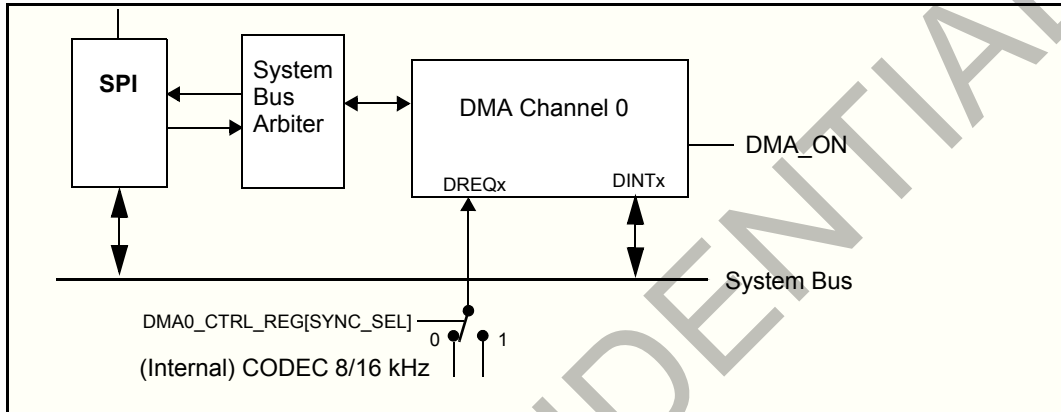


Figure 64: DMA Controller Block Diagram

24.4.1 DMA Channel Operation

A DMA channel is switched on with bit **DMA_ON**. This bit is automatically reset if the DMA transfer is finished.

The DMA channels can either be triggered by software (manual mode) or by an hardware interrupt source. If **DREQ_MODE** is '0', then a DMA channel is immediately triggered. If **DREQ_MODE** is '1' the DMA channel can be triggered by the 8/16 kHz from:

- CODEC if **DMA0_CTRL_REG[SYNC_SEL] = 0**

A DMA trigger can be used to copy data:

- From register to memory to peripheral (e.g CODEC_IN_OUT_REG)
- From CODEC to memory (e.g CODEC_IN_OUT_REG)
- From memory to memory

If DMA starts, data is transferred between address **DMA_A_START_REG** added to **DMA_A_IDX_REG** and **DMA_B_START_REG** added to **DMA_B_IDX_REG**. The direction is set with bit **DIR**. In Fly-by mode registers "A" points to memory and registers "B" are don't care.

If the DMA start condition is still true (**DREQ** in Figure 64) after the DMA cycle, the DMA continues. The DMA stops if **DREQ** is low or if **DMA_LEN_REG** is equal to **DMA_A_IDX_REG**. This condition also clears the **DMA_ON** bit.

If bit **CIRCULAR** is set to 1, the DMA controller automatically resets the **DMA_A_IDX_REGx** and **DMA_B_IDX_REG** and thus continues from its starting

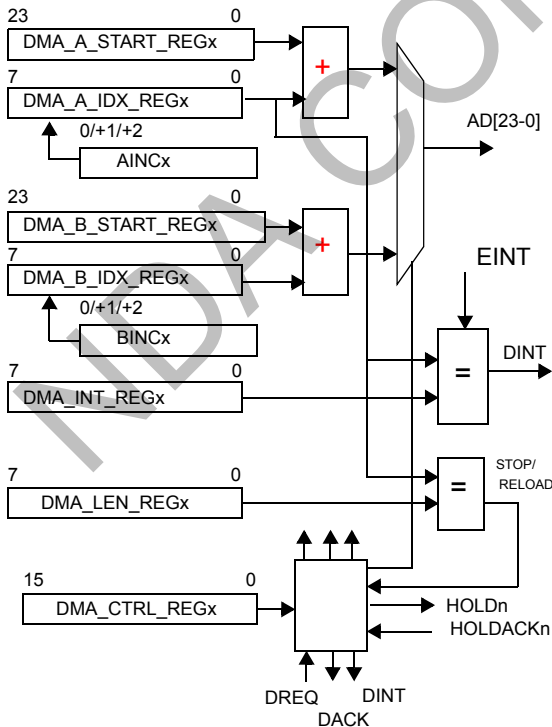


Figure 65: DMA Channel Diagram

address without stopping. If the DMA controller is started with **DREQ_MODE** =0, the DMA will always stop, regardless of the state of CIRCULAR.

Only 16-bit words can be transferred by the DMA controller. The DMA_A_IDX_REG and DMA_B_IDX_REG can automatically be post-incremented with 0 or 2 in bits **AINC[1-0]** and **BINC[1-0]**.

24.4.2 DMA Arbitration

The DMA channel shares the bus with SPI. If the DMA is master on the bus, the SPI read or write commands are not disturbed.

24.5 BANDGAP REFERENCE CIRCUIT

The bandgap reference circuit is used for internal reference voltages and bias currents of the CODEC.

To adjust the bandgap voltage an internal comparator is used as shown in [Figure 66](#). This comparator is enabled with **BG_COMP_PD** = 0. Next the **BANDGAP_REG[VI]** can be tuned until the **BG_COMP[1-0]** bit field reads '01'. If multiple values result in '01', the middle one should be selected. For power saving it is recommended to set **BG_COMP_PD** to '1' after the measurement.

The reference current can be trimmed using bits 8-6. As a reference the typical **lavd2** CODEC current of 4.5 mA can be used.

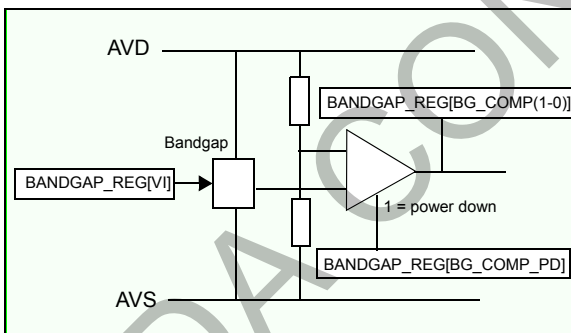


Figure 66: Bandgap Measurement Circuit

24.6 CODEC INPUT/OUTPUT PORTS (GPIO)

The CODEC has two programmable bidirectional GPIO ports as shown in Figure 67.

Features

- 2 pins: GPIO2, GPIO3
- Push-pull output configuration

- Selectable pull-up, pull-down resistors per pin
- VDDIO voltage 1.75 V to 3.45 V
- Programmable reference voltage for VDDIO <= 2 V and VDDIO > 2 V

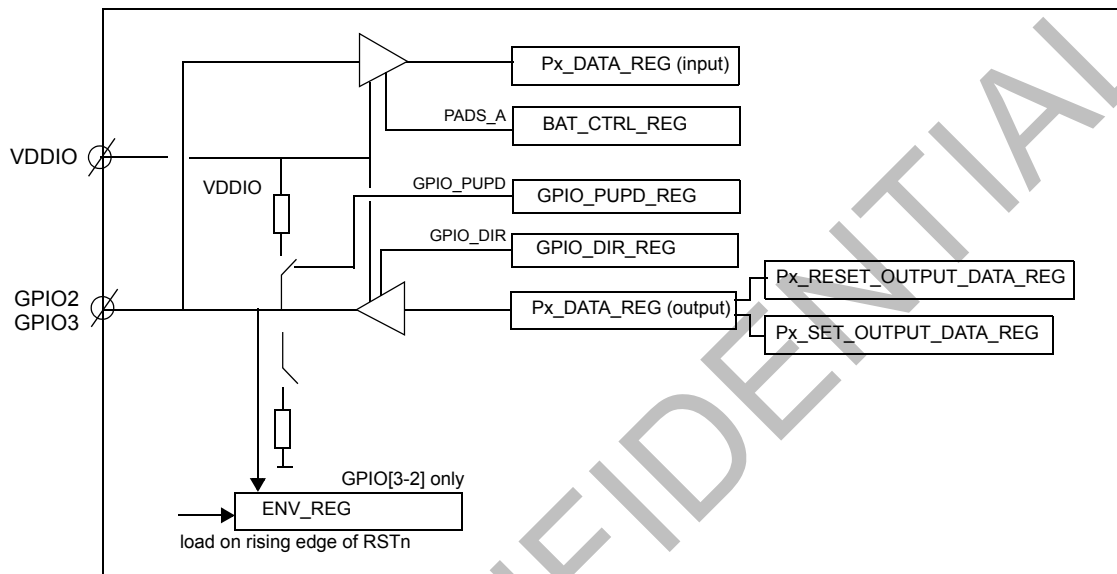


Figure 67: CODEC GPIO Ports

24.6.1 Pull-Up/Pull-Down Resistors

From startup, all digital I/Os are input and the pull-up resistors are enabled. The pull-up resistors are always disabled if the port is switched to output or if the GPIO_PUPD_REG[x] is set to '0'.

24.6.2 Port Direction

The port I/O direction is set with GPIO_DIR_REG. All output ports have a push-pull configuration. The GPIO output **level** can be set in two ways:

- Writing to GPIO_DATA_REG.
- Writing to GPIO_SET_DATA_REG to set a port pin.
Writing to GPIO_RESET_DATA_REG to reset a port pin.

24.6.3 Reference Voltage for Input Level

Register BAT_CTRL_REG[PADS_A] adjusts the reference voltage for PAD input stages. This register must be set to a higher value **before** a higher VDDIO is applied and set to a lower value **after** a lower VDDIO is applied. Reversing this order may damage the PADS.

0x: For VDDIO: 1.75 V to 2.0 V

1x: For VDDIO: 2.01 V to 3.45 V (default)

If PADS_A is kept at the reset value and VDDIO is lower than 2 V, the input levels cannot be guaranteed.

24.7 CODEC AND ANALOG FRONTEND

The DA14582 contains a 1.8 V CODEC and a fully differential Analog Front-End which can be configured for handsfree speakerphone applications with a corded handset.

24.7.1 Loudspeaker Connection

The loudspeaker stage can be fully configured through register **CODEC_LSR_REG**. The **LSRN_MODE**, and **LSRP_MODE** bits are used to select differential or single ended. (see [Figure 69](#))

Dynamic loudspeakers with an impedance as low as 28 Ω can be connected, as well as ceramic loudspeakers with an equivalent resistance of 600 Ω and 30 μF capacitance can be connected without additional circuitry. The loudspeaker attenuation can be controlled with **CODEC_LSR_REG[LSRATT]** from +5 dB to -9 dB in 8 steps of ~ 2 dB/step.

The LSR+ or LSR- can be configured as single output to the line interface. The unused LSR+ or LSR- pin can be configured as a voltage reference pin (REF) with 0.9 V output voltage.

24.7.2 Microphone Connection

The microphone stage can be fully configured through register **CODEC_MIC_REG**. The **MIC_MODE** bits are used to select differential or single-ended modes.

MICp, **MICn** are the terminals for a differential microphone connection. In single-ended applications **MICn** can be muted by setting **MICN_AGND_OUT** to '1'.

The microphone gain can be adjusted in the **CODEC_MIC_REG[MIC_GAIN]** from 0 dB to +30 dB in steps of 2 dB.

24.7.3 Microphone Supply

Several types of microphones can be used. The supply current for the microphone is fed through differential reference voltages, independent of the supply voltage.

For active microphones a voltage source with high power supply rejection ratio (PSRR) is provided on supply pins **VREFp/VREFm**. Filtering of internal and external reference voltages is provided with external capacitors on the **VBUF** pin. The time constants can be set with internal resistors. With **VBUF_INIT** set to '1' (reset value), 100 k Ω resistors are selected. If **VBUF_INIT** is '0' (recommended), 1 M Ω series resistance is selected.

24.7.4 CODEC Side Tone

For telephony applications a side tone can be enabled between the microphone and loudspeaker audio path.

The side tone is disabled after reset. To enable the side tone, **CODEC_ADDA_REG[SIDE_TONE_ON]** must be set to '1'. The side tone feedback level can be set with **CODEC_ADDA_REG[SIDE_TONE]** bits from 0 dB to -42 dB in steps of 6 dB.

If addition of the side tone signal would result in an overflow, the signal is automatically limited to a maximum c.q. minimum value.

24.7.5 CODEC Offset Compensation

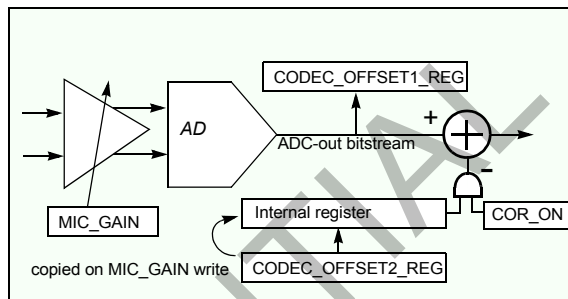


Figure 68: CODEC Offset Compensation

Although the **MIC_GAIN** amplifier offset error is very low, it can be compensated in the digital domain. Therefore the offset error for each gain setting must be measured at start-up while the **MIC_MUTE** is set to '0' (not muted), by reading out **CODEC_OFFSET_REG1** e.g. 100 times and averaging these values. Background noise may be present during the measurement.

For an applied **MIC_GAIN** setting, the corresponding average offset error value must then be written to **CODEC_OFFSET_REG2** to compensate the offset error. The value in this register becomes active in a seamless way after the new **MIC_GAIN** value is written to the **CODEC_MIC_REG**.

This synchronization process guarantees simultaneous activation of the new **MIC_GAIN** and corresponding **CODEC_OFFSET_REG2** value. During synchronization, which takes approximately 156 μs , **CODEC_TEST_CTRL_REG[COR_STAT]** becomes '1'. As soon as **COR_STAT** becomes '0', a new value may be loaded in **CODEC_OFFSET_REG2**. The **COR_ON** bit must be switched on **before** the initial load of **CODEC_OFFSET_REG2** and **MIC_GAIN**.

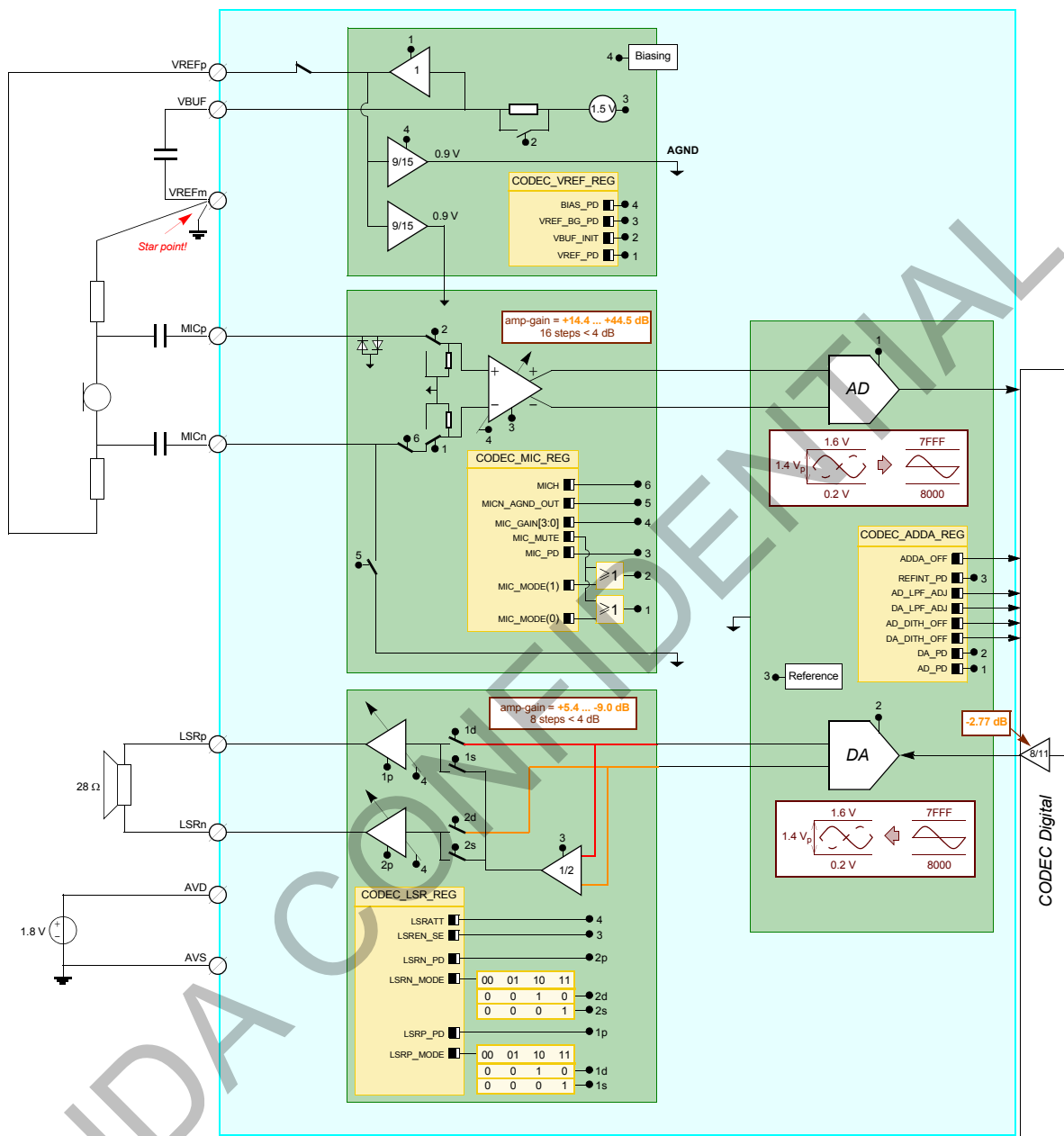


Figure 69: Sigma-Delta CODEC with Analog Front-End Block Diagram

24.8 CODEC MEMORY MAP

Table 32: CODEC Memory Map Accessible from SPI

| Address | Description |
|------------------------|----------------------|
| 0x000000 0x000FFF | shared RAM (4 kB) |
| 0x0001200 0x0001FFF | Registers |

Note 18: 1 kB = 1024 bytes. 1 Mbit = 1024*1024 bits. 1 GB = 1024*1024*1024 bytes.

Note 19: For Exchange memory configurations please refer to [Figure 71](#).

NDA CONFIDENTIAL

25 BLE Core

The BLE (Bluetooth low energy) core is a qualified Bluetooth baseband controller compatible with the Bluetooth low energy 4.2 specification and it is in charge of packet encoding/decoding and frame scheduling.

The block diagram is presented in Figure 70.

Features

- All device classes support (Broadcaster, Central, Observer, Peripheral)
- All packet types (Advertising / Data / Control)
- Encryption (AES / CCM)
- Bit stream processing (CRC, Whitening)
- FDMA/TDMA/events formatting and synchronization
- Frequency hopping calculation
- Operating clock 16 MHz or 8 MHz
- Low power modes supporting 32.0 kHz or 32.768 kHz
- Supports power down of the baseband during the protocol's idle periods
- AHB Slave interface for register file access
- AHB Slave interface for Exchange Memory access of CPU via BLE core
- AHB Master interface for direct access of BLE core to Exchange Memory space

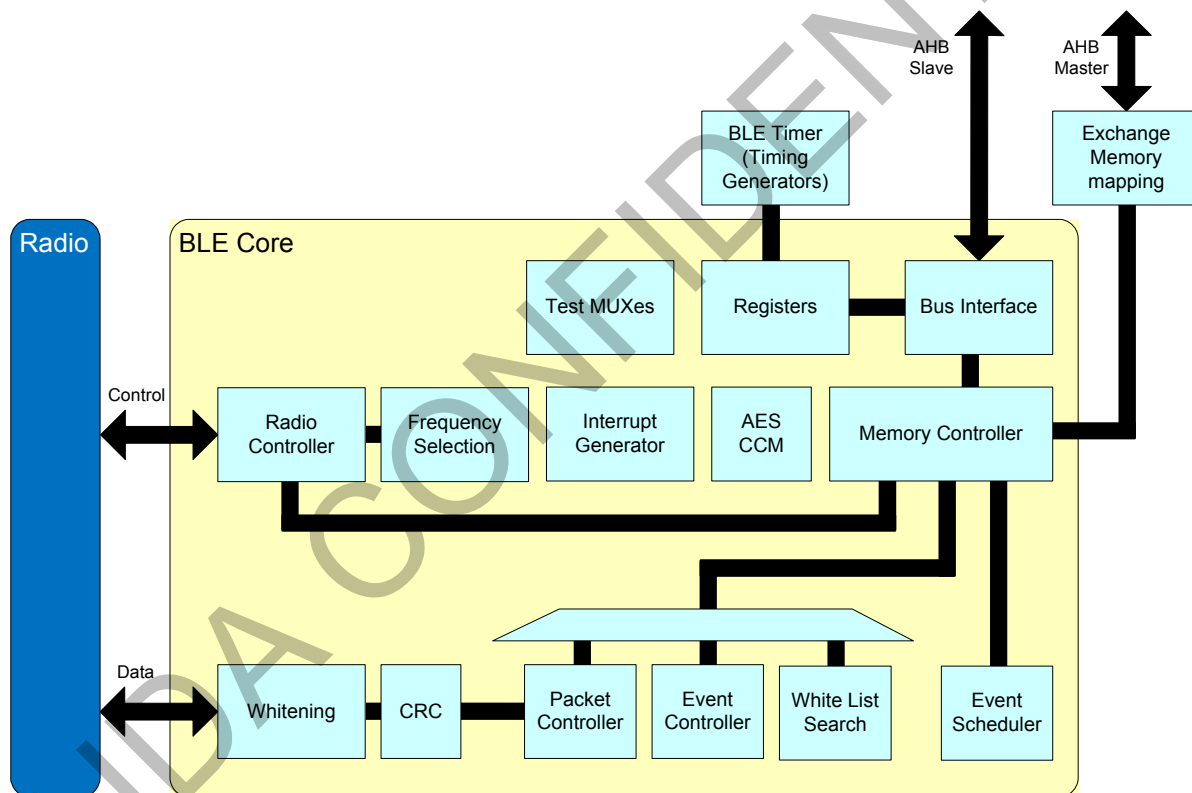


Figure 70: BLE Core Block Diagram

25.1 EXCHANGE MEMORY

The BLE Core requires access to a memory space named “Exchange Memory” to store control structures and frame buffers. The access to Exchange Memory is performed via the AHB Master interface. The mapping of the BLE Core address space to the System Bus address space is controlled via the register bit field GP_CONTROL_REG[EM_MAP]. Figure 71 illustrates all the address mapping options.



Figure 71: Exchange Memory Mapping Possibilities

There are 24 different cases of mapping the Exchange Memory onto the 5 physical memories (4 Retention RAMs and 1 SysRAM) in pages of 2 kB each. They should be selected according to the application needs regarding the amount of the Exchange Memory. So, for example, Case 15 provides 40 kB of SysRAM and 10 kB of Exchange Memory (8 kB using the Retention and another 2 kB page of the SysRAM cell). Hence, exchange memory requirements can be met by programming the respective case in the EM_MAP field of the GP_CONTROL_REG.

If the BLE core access an Exchange Memory area that is out of the boundaries specified by the GP_CONTROL_REG[EM_MAP], then EMACCERRSTAT bit field is asserted, causing BLE_ERROR_IRQ to be raised. The Interrupt Service Routine can detect the error condition by accessing the EMACCERRSTAT and can acknowledge by writing a "1" to EMACCERRACK. BLE_ERROR_IRQ is raised when either EMACCERRSTAT or ERRORINTSTAT are asserted.

25.2 PROGRAMMING BLE WAKE UP IRQ

Once BLE core switches to "BLE Deep Sleep mode" the only way to correctly exit from this state is by initially generating the BLE_WAKEUP_LP_IRQ and consecutively the BLE_SLP_IRQ. This sequence must be followed regardless of the cause of the termination of the "BLE Deep Sleep mode", i.e. regardless if the BLE Timer expired or BLE Timer has been stopped due to the assertion of BLE_WAKEUP_REQ.

The assertion and de-assertion of BLE_WAKEUP_LP_IRQ is fully controlled via the BLE_ENBPRESET_REG bit fields. Detailed description is following:

TWIRQ_SET: Number of "ble_lp_clk" cycles before the expiration of the BLE Timer, when the BLE_WAKEUP_LP_IRQ must be asserted. It is recommended to select a TWIRQ_SET value larger than the time required for the XTAL16_TRIM_READY_DELAY event, plus the IRQ Handler execution time. If the programmed value of TWIRQ_SET is less than the minimum recommended value, then the system will wake up but the actual BLE sleep duration (refer to BLE_DEEPSLSTAT_REG) will be larger than the programmed sleep duration (refer to BLE_DEEPSLWKUP_REG).

TWIRQ_RESET: Number of "ble_lp_clk" cycles before the expiration of the sleep period, when the BLE_WAKEUP_LP_IRQ will be de-asserted. It is recommended to always set to "1".

TWEXT: Determines the high period of BLE_WAKEUP_LP_IRQ, in the case of an external wake up event (refer to GP_CONTROL_REG[BLE_WAKEUP_REQ]). Minimum value is "TWIRQ_RESET + X", where X is the number of "ble_lp_clk" clock cycles that BLE_WAKEUP_LP_IRQ will be held high. Recommended value is "TWIRQ_RESET + 1". Note that as soon as GP_CONTROL_REG[BLE_WAKEUP_REQ] is set to "1" the BLE_WAKEUP_LP_IRQ will be asserted.

Minimum BLE Sleep Duration: The minimum value of BLE_DEEPSLWKUP_REG[DEEPSLTIME] time, measured in "ble_lp_clk" cycles, is the maximum of (a) "TWIRQ_SET + 1" and (b) the SW execution time from setting BLE_DEEPSLCTRL_REG[DEEP_SLEEP_ON] up to preparing CPU to accept the BLE_WAKEUP_LP_IRQ (e.g. to call the ARM instruction WFI). If programmed DEEPSLTIME is less than the aforementioned minimum value, then BLE_WAKEUP_LP_IRQ Handler may execute sooner than the call of ARM WFI instruction for example, causing SW instability.

25.3 SWITCH FROM ACTIVE MODE TO DEEP SLEEP MODE

Software can set the BLE core into the "BLE Deep Sleep mode", by first programming the timing of BLE_WAKEUP_LP_IRQ generation, then program the desired sleep duration at BLE_DEEPSLWKUP_REG and finally set the register bit BLE_DEEPSLCTRL_REG[DEEP_SLEEP_ON]. The BLE Core will switch to the "ble_lp_clk" (32.0 kHz or 32.768 kHz) in order to maintain its internal 625 μs timing reference. Software must poll the state of BLE_CNTL2_REG[RADIO_PWRDN_ALLOW] to detect the completion of this mode transition. Once the "ble_lp_clk" is used for base time reference, SW must disable the BLE clocks ("ble_master1_clk", "ble_master2_clk" and "ble_crypt_clk") by setting to "0" the CLK_RADIO_REG[BLE_ENABLE] register bit. Finally, SW can optionally power down the Radio Subsystem by using the PMU_CTRL_REG[RADIO_SLEEP] and the Peripheral and System power domains as well.

Figure 72 presents the waveforms while entering in BLE Deep Sleep mode. In this case, SW, as soon as it detects that RADIO_PWRDOWN_ALLOW is "1", it sets the PMU_CTRL_REG[RADIO_SLEEP] to power down the Radio Subsystem. At the following figures, the corresponding BLE Core signals are marked with red while Radio Subsystem is in power down state.

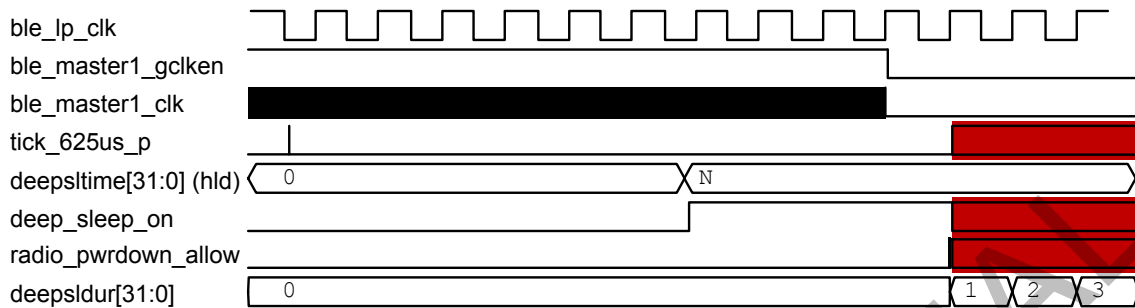


Figure 72 Entering into BLE Deep Sleep Mode

25.4 SWITCH FROM DEEP SLEEP MODE TO ACTIVE MODE

There are two possibilities for BLE Core to terminate the BLE Deep Sleep mode:

1. Termination at the end of a predetermined time.
2. Termination on software wake-up request, due to an external event.

25.4.1 Switching at Anchor Points

Figure 75 shows a typical deep sleep phase that is terminated at predetermined time. After a configurable time before the scheduled wake up time (configured via BLE_ENBPRESET_REG register bit fields), the BLE Timer asserts the BLE_WAKEUP_LP_IRQ in order to wake-up the CPU (powering up the System Power Domain). The BLE_WAKEUP_LP_IRQ Interrupt Handler will prepare the code environment and the XTAL16M oscillator stabilization (refer to SYS_STAT_REG[XTAL16_SETTLED]) and will decide when the BLE Core will be ready to exit from the BLE Deep Sleep mode.

Once the SW decides that BLE Core can wake up, it must enable the BLE clocks (via CLK_RADIO_REG[BLE_ENABLE]) and power up the Radio Power Domain (refer to PMU_CTRL_REG[RADIO_SLEEP] and SYS_STAT_REG[RAD_IS_UP]).

After the expiration of the sleep period (as specified in BLE_DEEPSLWKUP_REG[DEEPSLTIME]) the BLE Timer will not exit the BLE Deep Sleep mode until it detect that BLE Core is powered up. That means that if the SW requires more time to power up the BLE Core, then the final sleep duration (provided by BLE_DEEPSLSTAT_REG) will be larger than the pre-programmed value.

When BLE Timer is expired, BLE clocks are enabled and BLE Core (Radio Subsystem) is powered up, the BLE Core exists the "BLE Core Deep Sleep mode" and asserts the BLE_SLP_IRQ.

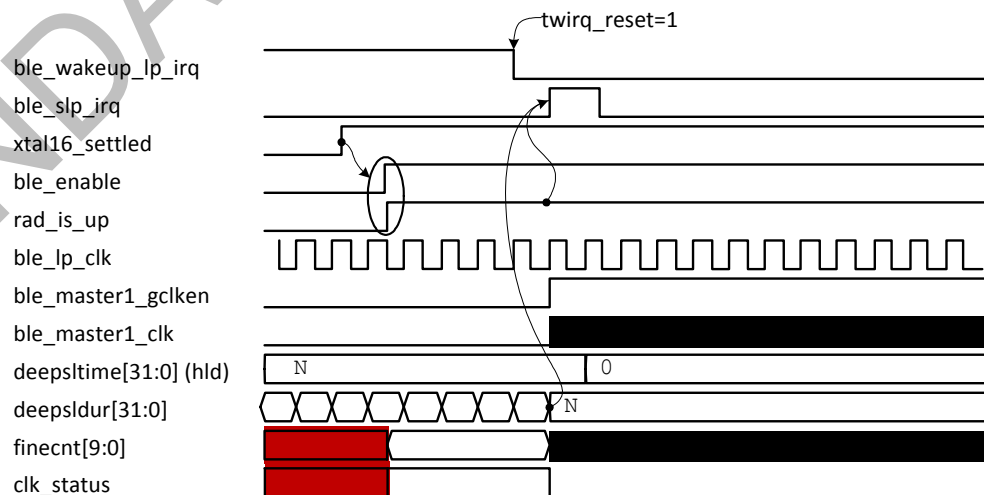


Figure 73 Exit BLE Deep Sleep Mode at Predetermined Time (Zoom In)

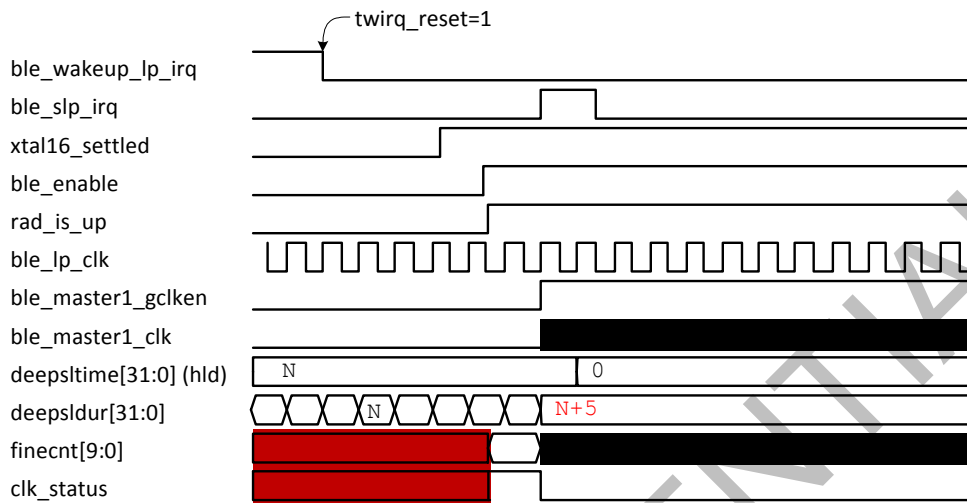


Figure 74 Exit BLE Deep Sleep Mode Later than the Predetermined Time (Zoom In)



Figure 75 Exit BLE Deep Sleep Mode at Predetermined Time (Zoom Out)

25.4.2 Switching Due to an External Event

Figure 76 shows a wake up from a deep sleep period forced by the assertion of register bit GP_CONTROL_REG[BLE_WAKEUP_REQ].

Assume that the system is in Deep Sleep state, i.e. all Power Domains have been switched off, and both the Wake-up Timer and Wake-up Controller have been programmed appropriately. Then assume that an event is detected at one of the GPIOs, causing the System Power Domain to wake-up due to WKUP_QUADDEC_IRQ. In that case, the SW will decide to wake-up the BLE core, then it should set to

“1” the GP_CONTROL_REG[BLE_WAKEUP_REQ] in order to force the wake up sequence.

In Figure 76 the BLE_WAKEUP_REQ is raised by the software as soon as possible, causing BLE_WAKEUP_LP_IRQ Handler to be executed as soon as possible. It is also possible to raise BLE_WAKEUP_REQ after the detection of XTAL16_TRIM_READY, causing both BLE_WAKEUP_LP_IRQ and BLE_SLP_IRQ Handlers to execute sequentially. The decision depends on the software structure and the application.

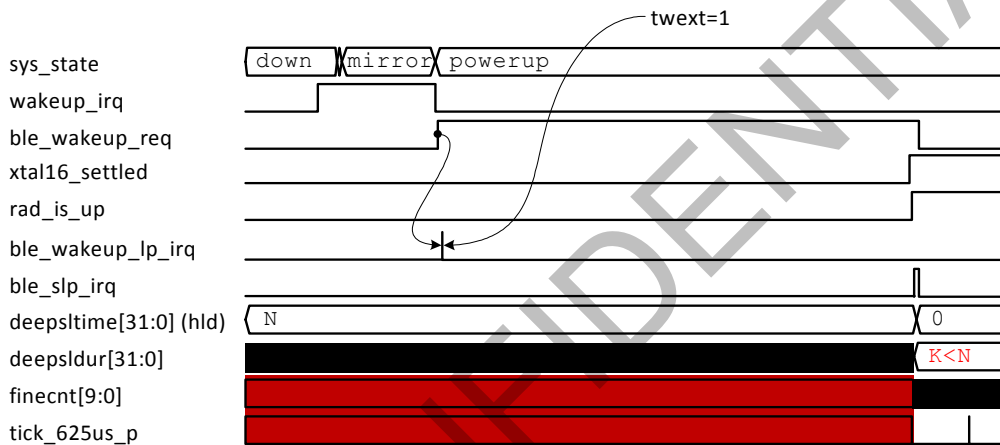


Figure 76 Exit BLE Deep Sleep Mode Due to External Event

As soon as bit field BLE_WAKEUP_REQ is set to “1” the BLE_WAKEUP_LP_IRQ will be asserted. In that case, the high period of BLE_WAKEUP_LP_IRQ is controlled via TWEXT. The recommended value of TWEXT is "TWIRQ_RESET + 1", meaning that BLE_WAKEUP_LP_IRQ will remain high for one “ble_llp_clk” period.

As long as the BLE_WAKEUP_REQ is high, entering the sleep mode is prohibited. Please note that BLE_WAKEUP_REQ event can be disabled by setting BLE_DEEPSLCNTL_REG[EXTWKUPDSB].

26 Radio

The Radio Transceiver implements the RF part of the Bluetooth low energy protocol. Together with the Bluetooth 4.2 PHY layer, this provides a 93 dB RF link budget for reliable wireless communication.

All RF blocks are supplied by on-chip low-drop out-regulators (LDOs). The bias scheme is programmable per block and optimized for minimum power consumption.

The Bluetooth LE radio block diagram is given in Figure 77. It comprises the Receiver, Transmitter, Synthesizer, Rx/Tx combiner block, and Biasing LDOs.

Features

- Single ended RFIO interface, 50 Ω matched
- Alignment free operation
- -93 dBm receiver sensitivity
- 0 dBm transmit output power
- Ultra low power consumption
- Fast frequency tuning minimizes overhead

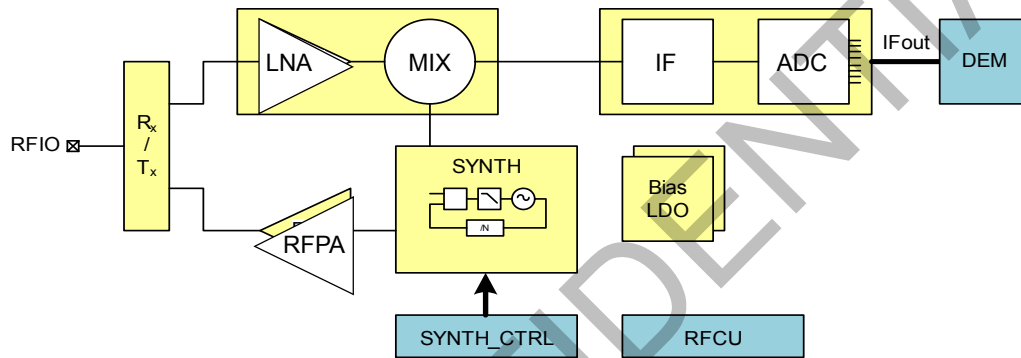


Figure 77: Bluetooth Radio Block Diagram

26.1 RECEIVER

The RX frontend consists of a selective matching network, a low noise amplifier (LNA) and an image rejection down conversion mixer. The LNA gain is controlled by the AGC.

The intermediate frequency (IF) part of the receiver comprises a complex filter and 2 variable gain amplifiers. This provides the necessary signal conditioning prior to digitalization. The digital demodulator block (DEM) provides a synchronous bit stream.

26.2 SYNTHESIZER

The RF Synthesizer generates the quadrature LO signal for the mixer, but also generates the modulated TX output signal. The VCO runs at twice the required frequency and a dedicated divide-by-2 circuit generates the 2.4 GHz signals in the required phase relations. Its frequency is controlled by a classic 3rd order type II PLL with a passive loop filter, operated in fractional-N mode. The reference frequency is the 16 MHz crystal clock. The multi-modulus divider has a nominal divide ratio of 153 which is varied by a $\Sigma\Delta$ modulator. The modulation of the TX frequency is performed by 2-point modulation. The fractional divide ratio also contains the shaped TX data stream. A second modulation path feeds the TX data stream directly to the VCO. The latter path is automatically calibrated from time to time to align the low and high frequency parts of the 2-point modulation scheme.

26.3 TRANSMITTER

The RF power amplifier (RFPA) is an extremely efficient Class-D structure, providing typically 0 dBm to the antenna. It is fed by the VCO's divide-by-2 circuit and delivers its TX power to the antenna pin through the combined RX/TX matching circuit.

26.4 RFIO

The RX/TX combiner block is a unique feature of the DA14582. It makes sure that the received power is applied to the LNA with minimum losses towards the RFPA. In TX mode, the LNA poses a minimal load for the RFPA and its input pins are protected from the RFPA. In both modes, the single ended RFIO port is matched to 50 Ω , in order to provide the simplest possible interfacing to the antenna on the printed circuit board.

26.5 BIASING

All RF blocks are supplied by on-chip low-drop out-regulators (LDOs). The bias scheme is programmable per block and optimized for minimum power consumption.

26.6 CONTROL

The radio control unit (RFCU) controls the block timing and configuration registers. The BLE interfaces directly with the RFCU. The DA14582 can be put in test mode using a standard Bluetooth tester (e.g. Rohde & Schwarz CBT with K57 option) by connecting the antenna terminal for the RF link and the UART as described in section 14.7.

27 Registers

This section contains a detailed view of the DA14582 registers. It is organized as follows: An overview table is presented initially, which depicts all register names, addresses and descriptions. A detailed bit level description of each register follows.

Note: The DA14582 does not support DC-DC converter Boost mode.

The register file of the ARM Cortex-M0 can be found in the following documents, available on the ARM website:

Devices Generic User Guide:

DUI0497A_cortex_m0_r0p0_generic_ug.pdf

Technical Reference Manual:

DDI0432C_cortex_m0_r0p0_trm.pdf

These documents contain the register descriptions for the Nested Vectored Interrupt Controller (NVIC), the System Control Block (SCB) and the System Timer (SysTick).

NDA CONFIDENTIAL

Table 33: Register map

| Address | Port | Description |
|------------|--------------------------|--|
| 0x40000000 | BLE_RWBTLCTRL_REG | BLE Control register |
| 0x40000004 | BLE_VERSION_REG | Version register |
| 0x40000008 | BLE_RWBTLCONF_REG | Configuration register |
| 0x4000000C | BLE_INTCNTL_REG | Interrupt controller register |
| 0x40000010 | BLE_INTSTAT_REG | Interrupt status register |
| 0x40000014 | BLE_INTRAWSTAT_REG | Interrupt raw status register |
| 0x40000018 | BLE_INTACK_REG | Interrupt acknowledge register |
| 0x4000001C | BLE_BASETIMECNT_REG | Base time reference counter |
| 0x40000020 | BLE_FINETIMECNT_REG | Fine time reference counter |
| 0x40000024 | BLE_BDADDR_L_REG | BLE device address LSB register |
| 0x40000028 | BLE_BDADDR_U_REG | BLE device address MSB register |
| 0x4000002C | BLE_CURRENTRXDESCPTR_REG | Rx Descriptor Pointer for the Receive Buffer Chained List |
| 0x40000030 | BLE_DEEPSLCTRL_REG | Deep-Sleep control register |
| 0x40000034 | BLE_DEEPSLWUP_REG | Time (measured in Low Power clock cycles) in Deep Sleep Mode before waking-up the device |
| 0x40000038 | BLE_DEEPSLSTAT_REG | Duration of the last deep sleep phase register |
| 0x4000003C | BLE_ENBPRESSET_REG | Time in low power oscillator cycles register |
| 0x40000040 | BLE_FINECNTCORR_REG | Phase correction value register |
| 0x40000044 | BLE_BASETIMECNTCORR_REG | Base Time Counter |
| 0x40000050 | BLE_DIAGCNTL_REG | Diagnostics Register |
| 0x40000054 | BLE_DIAGSTAT_REG | Debug use only |
| 0x40000058 | BLE_DEBUGADDMAX_REG | Upper limit for the memory zone |
| 0x4000005C | BLE_DEBUGADDMIN_REG | Lower limit for the memory zone |
| 0x40000060 | BLE_ERRORYPESTAT_REG | Error Type Status registers |
| 0x40000064 | BLE_SWPROFILING_REG | Software Profiling register |
| 0x40000070 | BLE_RADIOCNTL0_REG | Radio interface control register |
| 0x40000074 | BLE_RADIOCNTL1_REG | Radio interface control register |
| 0x40000080 | BLE_RADIOWRUPDN_REG | RX/TX power up/down phase register |
| 0x40000090 | BLE_ADVCHMAP_REG | Advertising Channel Map |
| 0x400000A0 | BLE_ADVTIM_REG | Advertising Packet Interval |
| 0x400000A4 | BLE_ACTSCANSTAT_REG | Active scan register |
| 0x400000B0 | BLE_WLPUBADDPTR_REG | Start address of public devices list |
| 0x400000B4 | BLE_WLPRIVADDPTR_REG | Start address of private devices list |
| 0x400000B8 | BLE_WLNBDEV_REG | Devices in white list |
| 0x400000C0 | BLE_AESCNTL_REG | Start AES register |
| 0x400000C4 | BLE_AESKEY31_0_REG | AES encryption key |
| 0x400000C8 | BLE_AESKEY63_32_REG | AES encryption key |
| 0x400000CC | BLE_AESKEY95_64_REG | AES encryption key |
| 0x400000D0 | BLE_AESKEY127_96_REG | AES encryption key |
| 0x400000D4 | BLE_AESPTR_REG | Pointer to the block to encrypt/decrypt |
| 0x400000D8 | BLE_TXMICVAL_REG | AES / CCM plain MIC value |
| 0x400000DC | BLE_RXMICVAL_REG | AES / CCM plain MIC value |
| 0x400000E0 | BLE_RFTESTCNTL_REG | RF Testing Register |

Table 33: Register map

| Address | Port | Description |
|------------|---------------------|---|
| 0x40000F0 | BLE_TIMGENCNTL_REG | Timing Generator Register |
| 0x40000F4 | BLE_GROSSTIMTGT_REG | Gross Timer Target value |
| 0x40000F8 | BLE_FINETIMTGT_REG | Fine Timer Target value |
| 0x40000FC | BLE_SAMPLECLK_REG | Samples the Base Time Counter |
| 0x4000200 | BLE_CNTL2_REG | BLE Control Register 2 |
| 0x4000204 | BLE_RF_DIAGIRQ_REG | BLE/RF Diagnostic IRQ Control Register |
| 0x4000800 | OTPC_MODE_REG | Mode register |
| 0x4000804 | OTPC_PCTRL_REG | Bit-programming control register |
| 0x4000808 | OTPC_STAT_REG | Status register |
| 0x400080C | OTPC_AHBADR_REG | AHB master start address |
| 0x40008010 | OTPC_CELADR_REG | Macrocell start address |
| 0x40008014 | OTPC_NWORDS_REG | Number of words |
| 0x40008018 | OTPC_FFPRT_REG | Ports access to fifo logic |
| 0x4000801C | OTPC_FFRD_REG | Latest read data from the OTPC_FFPRT_REG |
| 0x50000000 | CLK_AMBA_REG | HCLK, PCLK, divider and clock gates |
| 0x50000002 | CLK_FREQ_TRIM_REG | Xtal frequency trimming register |
| 0x50000004 | CLK_PER_REG | Peripheral divider register |
| 0x50000008 | CLK_RADIO_REG | Radio PLL control register |
| 0x50000010 | PMU_CTRL_REG | Power Management Unit control register |
| 0x50000012 | SYS_CTRL_REG | System Control register |
| 0x50000014 | SYS_STAT_REG | System status register |
| 0x50000016 | TRIM_CTRL_REG | Control trimming of the XTAL16M |
| 0x50000020 | CLK_32K_REG | 32 kHz oscillator register |
| 0x50000022 | CLK_16M_REG | 16 MHz RC-oscillator register |
| 0x50000024 | CLK_RCX20K_REG | 20 kHz RXC-oscillator control register |
| 0x5000002A | ANA_STATUS_REG | Status bit of analog (power management) circuits |
| 0x50000100 | WKUP_CTRL_REG | Control register for the wakeup counter |
| 0x50000102 | WKUP_COMPARE_REG | Number of events before wakeup interrupt |
| 0x50000104 | WKUP_RESET_IRQ_REG | Reset wakeup interrupt |
| 0x50000106 | WKUP_COUNTER_REG | Actual number of events of the wakeup counter |
| 0x50000108 | WKUP_RESET_CNTR_REG | Reset the event counter |
| 0x5000010A | WKUP_SELECT_P0_REG | Select which inputs from P0 port can trigger wkup counter |
| 0x5000010C | WKUP_SELECT_P1_REG | Select which inputs from P1 port can trigger wkup counter |
| 0x5000010E | WKUP_SELECT_P2_REG | Select which inputs from P2 port can trigger wkup counter |
| 0x50000110 | WKUP_SELECT_P3_REG | Select which inputs from P3 port can trigger wkup counter |
| 0x50000112 | WKUP_POL_P0_REG | Select the sensitivity polarity for each P0 input |
| 0x50000114 | WKUP_POL_P1_REG | Select the sensitivity polarity for each P1 input |
| 0x50000116 | WKUP_POL_P2_REG | Select the sensitivity polarity for each P2 input |
| 0x50000118 | WKUP_POL_P3_REG | Select the sensitivity polarity for each P3 input |
| 0x50000200 | QDEC_CTRL_REG | Quad Decoder control register |
| 0x50000202 | QDEC_XCNT_REG | Counter value of the X Axis |

Table 33: Register map

| Address | Port | Description |
|------------|----------------------|---|
| 0x5000204 | QDEC_YCNT_REG | Counter value of the Y Axis |
| 0x5000206 | QDEC_CLOCKDIV_REG | Clock divider register |
| 0x5000208 | QDEC_CTRL2_REG | Quad Decoder control register |
| 0x500020A | QDEC_ZCNT_REG | Z_counter |
| 0x50001000 | UART_RBR_THR_DLL_REG | Receive Buffer Register |
| 0x50001004 | UART_IER_DLH_REG | Interrupt Enable Register |
| 0x50001008 | UART_IIR_FCR_REG | Interrupt Identification Register/FIFO Control Register |
| 0x5000100C | UART_LCR_REG | Line Control Register |
| 0x50001010 | UART_MCR_REG | Modem Control Register |
| 0x50001014 | UART_LSR_REG | Line Status Register |
| 0x50001018 | UART_MSR_REG | Modem Status Register |
| 0x5000101C | UART_SCR_REG | Scratchpad Register |
| 0x50001020 | UART_LPDLL_REG | Low Power Divisor Latch Low |
| 0x50001024 | UART_LPDH_REG | Low Power Divisor Latch High |
| 0x50001030 | UART_SRBR_STHR0_REG | Shadow Receive/Transmit Buffer Register |
| 0x50001034 | UART_SRBR_STHR1_REG | Shadow Receive/Transmit Buffer Register |
| 0x50001038 | UART_SRBR_STHR2_REG | Shadow Receive/Transmit Buffer Register |
| 0x5000103C | UART_SRBR_STHR3_REG | Shadow Receive/Transmit Buffer Register |
| 0x50001040 | UART_SRBR_STHR4_REG | Shadow Receive/Transmit Buffer Register |
| 0x50001044 | UART_SRBR_STHR5_REG | Shadow Receive/Transmit Buffer Register |
| 0x50001048 | UART_SRBR_STHR6_REG | Shadow Receive/Transmit Buffer Register |
| 0x5000104C | UART_SRBR_STHR7_REG | Shadow Receive/Transmit Buffer Register |
| 0x50001050 | UART_SRBR_STHR8_REG | Shadow Receive/Transmit Buffer Register |
| 0x50001054 | UART_SRBR_STHR9_REG | Shadow Receive/Transmit Buffer Register |
| 0x50001058 | UART_SRBR_STHR10_REG | Shadow Receive/Transmit Buffer Register |
| 0x5000105C | UART_SRBR_STHR11_REG | Shadow Receive/Transmit Buffer Register |
| 0x50001060 | UART_SRBR_STHR12_REG | Shadow Receive/Transmit Buffer Register |
| 0x50001064 | UART_SRBR_STHR13_REG | Shadow Receive/Transmit Buffer Register |
| 0x50001068 | UART_SRBR_STHR14_REG | Shadow Receive/Transmit Buffer Register |
| 0x5000106C | UART_SRBR_STHR15_REG | Shadow Receive/Transmit Buffer Register |
| 0x5000107C | UART_USR_REG | UART Status register. |
| 0x50001080 | UART_TFL_REG | Transmit FIFO Level |
| 0x50001084 | UART_RFL_REG | Receive FIFO Level. |
| 0x50001088 | UART_SRR_REG | Software Reset Register. |
| 0x5000108C | UART_SRTS_REG | Shadow Request to Send |
| 0x50001090 | UART_SBCR_REG | Shadow Break Control Register |
| 0x50001094 | UART_SDMAM_REG | Shadow DMA Mode |
| 0x50001098 | UART_SFE_REG | Shadow FIFO Enable |
| 0x5000109C | UART_SRT_REG | Shadow RCVR Trigger |
| 0x500010A0 | UART_STET_REG | Shadow TX Empty Trigger |
| 0x500010A4 | UART_HTX_REG | Halt TX |
| 0x500010F4 | UART_CPR_REG | Component Parameter Register |
| 0x500010F8 | UART_UCV_REG | Component Version |
| 0x500010FC | UART_CTR_REG | Component Type Register |

Table 33: Register map

| Address | Port | Description |
|------------|-----------------------|---|
| 0x50001100 | UART2_RBR_THR_DLL_REG | Receive Buffer Register |
| 0x50001104 | UART2_IER_DLH_REG | Interrupt Enable Register |
| 0x50001108 | UART2_IIR_FCR_REG | Interrupt Identification Register/FIFO Control Register |
| 0x5000110C | UART2_LCR_REG | Line Control Register |
| 0x50001110 | UART2_MCR_REG | Modem Control Register |
| 0x50001114 | UART2_LSR_REG | Line Status Register |
| 0x50001118 | UART2_MSR_REG | Modem Status Register |
| 0x5000111C | UART2_SCR_REG | Scratchpad Register |
| 0x50001120 | UART2_LPDLL_REG | Low Power Divisor Latch Low |
| 0x50001124 | UART2_LPDH_REG | Low Power Divisor Latch High |
| 0x50001130 | UART2_SRBR_STHR0_REG | Shadow Receive/Transmit Buffer Register |
| 0x50001134 | UART2_SRBR_STHR1_REG | Shadow Receive/Transmit Buffer Register |
| 0x50001138 | UART2_SRBR_STHR2_REG | Shadow Receive/Transmit Buffer Register |
| 0x5000113C | UART2_SRBR_STHR3_REG | Shadow Receive/Transmit Buffer Register |
| 0x50001140 | UART2_SRBR_STHR4_REG | Shadow Receive/Transmit Buffer Register |
| 0x50001144 | UART2_SRBR_STHR5_REG | Shadow Receive/Transmit Buffer Register |
| 0x50001148 | UART2_SRBR_STHR6_REG | Shadow Receive/Transmit Buffer Register |
| 0x5000114C | UART2_SRBR_STHR7_REG | Shadow Receive/Transmit Buffer Register |
| 0x50001150 | UART2_SRBR_STHR8_REG | Shadow Receive/Transmit Buffer Register |
| 0x50001154 | UART2_SRBR_STHR9_REG | Shadow Receive/Transmit Buffer Register |
| 0x50001158 | UART2_SRBR_STHR10_REG | Shadow Receive/Transmit Buffer Register |
| 0x5000115C | UART2_SRBR_STHR11_REG | Shadow Receive/Transmit Buffer Register |
| 0x50001160 | UART2_SRBR_STHR12_REG | Shadow Receive/Transmit Buffer Register |
| 0x50001164 | UART2_SRBR_STHR13_REG | Shadow Receive/Transmit Buffer Register |
| 0x50001168 | UART2_SRBR_STHR14_REG | Shadow Receive/Transmit Buffer Register |
| 0x5000116C | UART2_SRBR_STHR15_REG | Shadow Receive/Transmit Buffer Register |
| 0x5000117C | UART2_USR_REG | UART Status register. |
| 0x50001180 | UART2_TFL_REG | Transmit FIFO Level |
| 0x50001184 | UART2_RFL_REG | Receive FIFO Level. |
| 0x50001188 | UART2_SRR_REG | Software Reset Register. |
| 0x5000118C | UART2_SRTS_REG | Shadow Request to Send |
| 0x50001190 | UART2_SBCR_REG | Shadow Break Control Register |
| 0x50001194 | UART2_SDMAM_REG | Shadow DMA Mode |
| 0x50001198 | UART2_SFE_REG | Shadow FIFO Enable |
| 0x5000119C | UART2_SRT_REG | Shadow RCVR Trigger |
| 0x500011A0 | UART2_STET_REG | Shadow TX Empty Trigger |
| 0x500011A4 | UART2_HTX_REG | Halt TX |
| 0x500011F4 | UART2_CPR_REG | Component Parameter Register |
| 0x500011F8 | UART2_UCV_REG | Component Version |
| 0x500011FC | UART2_CTR_REG | Component Type Register |
| 0x50001200 | SPI_CTRL_REG | SPI control register 0 |
| 0x50001202 | SPI_RX_TX_REG0 | SPI RX/TX register0 |
| 0x50001204 | SPI_RX_TX_REG1 | SPI RX/TX register1 |
| 0x50001206 | SPI_CLEAR_INT_REG | SPI clear interrupt register |

Table 33: Register map

| Address | Port | Description |
|------------|--------------------------|--|
| 0x50001208 | SPI_CTRL_REG1 | SPI control register 1 |
| 0x50001300 | I2C_CON_REG | I2C Control Register |
| 0x50001304 | I2C_TAR_REG | I2C Target Address Register |
| 0x50001308 | I2C_SAR_REG | I2C Slave Address Register |
| 0x50001310 | I2C_DATA_CMD_REG | I2C Rx/Tx Data Buffer and Command Register |
| 0x50001314 | I2C_SS_SCL_HCNT_REG | Standard Speed I2C Clock SCL High Count Register |
| 0x50001318 | I2C_SS_SCL_LCNT_REG | Standard Speed I2C Clock SCL Low Count Register |
| 0x5000131C | I2C_FS_SCL_HCNT_REG | Fast Speed I2C Clock SCL High Count Register |
| 0x50001320 | I2C_FS_SCL_LCNT_REG | Fast Speed I2C Clock SCL Low Count Register |
| 0x5000132C | I2C_INTR_STAT_REG | I2C Interrupt Status Register |
| 0x50001330 | I2C_INTR_MASK_REG | I2C Interrupt Mask Register |
| 0x50001334 | I2C_RAW_INTR_STAT_REG | I2C Raw Interrupt Status Register |
| 0x50001338 | I2C_RX_TL_REG | I2C Receive FIFO Threshold Register |
| 0x5000133C | I2C_TX_TL_REG | I2C Transmit FIFO Threshold Register |
| 0x50001340 | I2C_CLR_INTR_REG | Clear Combined and Individual Interrupt Register |
| 0x50001344 | I2C_CLR_RX_UNDER_REG | Clear RX_UNDER Interrupt Register |
| 0x50001348 | I2C_CLR_RX_OVER_REG | Clear RX_OVER Interrupt Register |
| 0x5000134C | I2C_CLR_TX_OVER_REG | Clear TX_OVER Interrupt Register |
| 0x50001350 | I2C_CLR_RD_REQ_REG | Clear RD_REQ Interrupt Register |
| 0x50001354 | I2C_CLR_TX_ABRT_REG | Clear TX_ABRT Interrupt Register |
| 0x50001358 | I2C_CLR_RX_DONE_REG | Clear RX_DONE Interrupt Register |
| 0x5000135C | I2C_CLR_ACTIVITY_REG | Clear ACTIVITY Interrupt Register |
| 0x50001360 | I2C_CLR_STOP_DET_REG | Clear STOP_DET Interrupt Register |
| 0x50001364 | I2C_CLR_START_DET_REG | Clear START_DET Interrupt Register |
| 0x50001368 | I2C_CLR_GEN_CALL_REG | Clear GEN_CALL Interrupt Register |
| 0x5000136C | I2C_ENABLE_REG | I2C Enable Register |
| 0x50001370 | I2C_STATUS_REG | I2C Status Register |
| 0x50001374 | I2C_TXFLR_REG | I2C Transmit FIFO Level Register |
| 0x50001378 | I2C_RXFLR_REG | I2C Receive FIFO Level Register |
| 0x5000137C | I2C_SDA_HOLD_REG | I2C SDA Hold Time Length Register |
| 0x50001380 | I2C_TX_ABRT_SOURCE_REG | I2C Transmit Abort Source Register |
| 0x50001394 | I2C_SDA_SETUP_REG | I2C SDA Setup Register |
| 0x50001398 | I2C_ACK_GENERAL_CALL_REG | I2C ACK General Call Register |
| 0x5000139C | I2C_ENABLE_STATUS_REG | I2C Enable Status Register |
| 0x500013A0 | I2C_IC_FS_SPKLEN_REG | I2C SS and FS spike suppression limit Size |
| 0x50001400 | GPIO_IRQ0_IN_SEL_REG | GPIO interrupt selection for GPIO_IRQ0 |
| 0x50001402 | GPIO_IRQ1_IN_SEL_REG | GPIO interrupt selection for GPIO_IRQ1 |
| 0x50001404 | GPIO_IRQ2_IN_SEL_REG | GPIO interrupt selection for GPIO_IRQ2 |
| 0x50001406 | GPIO_IRQ3_IN_SEL_REG | GPIO interrupt selection for GPIO_IRQ3 |
| 0x50001408 | GPIO_IRQ4_IN_SEL_REG | GPIO interrupt selection for GPIO_IRQ4 |
| 0x5000140C | GPIO_DEBOUNCE_REG | debounce counter value for GPIO inputs |
| 0x5000140E | GPIO_RESET_IRQ_REG | GPIO interrupt reset register |
| 0x50001410 | GPIO_INT_LEVEL_CTRL_REG | high or low level select for GPIO interrupts |
| 0x50001412 | KBRD_IRQ_IN_SEL0_REG | GPIO interrupt selection for KBRD_IRQ for P0 |

Table 33: Register map

| Address | Port | Description |
|------------|----------------------|---|
| 0x50001414 | KBRD_IRQ_IN_SEL1_REG | GPIO interrupt selection for KBRD_IRQ for P1 and P2 |
| 0x50001416 | KBRD_IRQ_IN_SEL2_REG | GPIO interrupt selection for KBRD_IRQ for P3 |
| 0x50001500 | GP_ADC_CTRL_REG | General Purpose ADC Control Register |
| 0x50001502 | GP_ADC_CTRL2_REG | General Purpose ADC Second Control Register |
| 0x50001504 | GP_ADC_OFFP_REG | General Purpose ADC Positive Offset Register |
| 0x50001506 | GP_ADC_OFFN_REG | General Purpose ADC Negative Offset Register |
| 0x50001508 | GP_ADC_CLEAR_INT_REG | General Purpose ADC Clear Interrupt Register |
| 0x5000150A | GP_ADC_RESULT_REG | General Purpose ADC Result Register |
| 0x5000150C | GP_ADC_DELAY_REG | General Purpose ADC Delay Register |
| 0x5000150E | GP_ADC_DELAY2_REG | General Purpose ADC Second Delay Register |
| 0x50001600 | CLK_REF_SEL_REG | Select clock for oscillator calibration |
| 0x50001602 | CLK_REF_CNT_REG | Count value for oscillator calibration |
| 0x50001604 | CLK_REF_VAL_L_REG | XTAL16M reference cycles, lower 16 bits |
| 0x50001606 | CLK_REF_VAL_H_REG | XTAL16M reference cycles, upper 16 bits |
| 0x50003000 | P0_DATA_REG | P0 Data input / output register |
| 0x50003002 | P0_SET_DATA_REG | P0 Set port pins register |
| 0x50003004 | P0_RESET_DATA_REG | P0 Reset port pins register |
| 0x50003006 | P00_MODE_REG | P00 Mode Register |
| 0x50003008 | P01_MODE_REG | P01 Mode Register |
| 0x5000300A | P02_MODE_REG | P02 Mode Register |
| 0x5000300C | P03_MODE_REG | P03 Mode Register |
| 0x5000300E | P04_MODE_REG | P04 Mode Register |
| 0x50003010 | P05_MODE_REG | P05 Mode Register |
| 0x50003012 | P06_MODE_REG | P06 Mode Register |
| 0x50003014 | P07_MODE_REG | P07 Mode Register |
| 0x50003020 | P1_DATA_REG | P1 Data input / output register |
| 0x50003022 | P1_SET_DATA_REG | P1 Set port pins register |
| 0x50003024 | P1_RESET_DATA_REG | P1 Reset port pins register |
| 0x50003026 | P10_MODE_REG | P10 Mode Register |
| 0x50003028 | P11_MODE_REG | P11 Mode Register |
| 0x5000302A | P12_MODE_REG | P12 Mode Register |
| 0x5000302C | P13_MODE_REG | P13 Mode Register |
| 0x5000302E | P14_MODE_REG | P14 Mode Register |
| 0x50003030 | P15_MODE_REG | P15 Mode Register |
| 0x50003040 | P2_DATA_REG | P2 Data input / output register |
| 0x50003042 | P2_SET_DATA_REG | P2 Set port pins register |
| 0x50003044 | P2_RESET_DATA_REG | P2 Reset port pins register |
| 0x50003046 | P20_MODE_REG | P20 Mode Register |
| 0x50003048 | P21_MODE_REG | P21 Mode Register |
| 0x5000304A | P22_MODE_REG | P22 Mode Register |
| 0x5000304C | P23_MODE_REG | P23 Mode Register |
| 0x5000304E | P24_MODE_REG | P24 Mode Register |
| 0x50003050 | P25_MODE_REG | P25 Mode Register |
| 0x50003052 | P26_MODE_REG | P26 Mode Register |

Table 33: Register map

| Address | Port | Description |
|------------|----------------------|---|
| 0x50003054 | P27_MODE_REG | P27 Mode Register |
| 0x50003056 | P28_MODE_REG | P28 Mode Register |
| 0x50003058 | P29_MODE_REG | P29 Mode Register |
| 0x50003070 | P01_PADPWR_CTRL_REG | Ports 0 and 1 Output Power Control Register |
| 0x50003072 | P2_PADPWR_CTRL_REG | Port 2 Output Power Control Register |
| 0x50003074 | P3_PADPWR_CTRL_REG | Port 3 Output Power Control Register |
| 0x50003080 | P3_DATA_REG | P3 Data input / output register |
| 0x50003082 | P3_SET_DATA_REG | P3 Set port pins register |
| 0x50003084 | P3_RESET_DATA_REG | P3 Reset port pins register |
| 0x50003086 | P30_MODE_REG | P30 Mode Register |
| 0x50003088 | P31_MODE_REG | P31 Mode Register |
| 0x5000308A | P32_MODE_REG | P32 Mode Register |
| 0x5000308C | P33_MODE_REG | P33 Mode Register |
| 0x5000308E | P34_MODE_REG | P34 Mode Register |
| 0x50003090 | P35_MODE_REG | P35 Mode Register |
| 0x50003092 | P36_MODE_REG | P36 Mode Register |
| 0x50003094 | P37_MODE_REG | P37 Mode Register |
| 0x50003100 | WATCHDOG_REG | Watchdog timer register. |
| 0x50003102 | WATCHDOG_CTRL_REG | Watchdog control register. |
| 0x50003200 | CHIP_ID1_REG | Chip identification register 1. |
| 0x50003201 | CHIP_ID2_REG | Chip identification register 2. |
| 0x50003202 | CHIP_ID3_REG | Chip identification register 3. |
| 0x50003203 | CHIP_SWC_REG | Software compatibility register. |
| 0x50003204 | CHIP_REVISION_REG | Chip revision register. |
| 0x50003300 | SET_FREEZE_REG | Controls freezing of various timers/counters. |
| 0x50003302 | RESET_FREEZE_REG | Controls unfreezing of various timers/counters. |
| 0x50003304 | DEBUG_REG | Various debug information register. |
| 0x50003306 | GP_STATUS_REG | General purpose system status register. |
| 0x50003308 | GP_CONTROL_REG | General purpose system control register. |
| 0x50003400 | TIMER0_CTRL_REG | Timer0 control register |
| 0x50003402 | TIMER0_ON_REG | Timer0 on control register |
| 0x50003404 | TIMER0_RELOAD_M_REG | 16 bits reload value for Timer0 |
| 0x50003406 | TIMER0_RELOAD_N_REG | 16 bits reload value for Timer0 |
| 0x50003408 | PWM2_DUTY_CYCLE | Duty Cycle for PWM2 |
| 0x5000340A | PWM3_DUTY_CYCLE | Duty Cycle for PWM3 |
| 0x5000340C | PWM4_DUTY_CYCLE | Duty Cycle for PWM4 |
| 0x5000340E | TRIPLE_PWM_FREQUENCY | Frequency for PWM 2,3 and 4 |
| 0x50003410 | TRIPLE_PWM_CTRL_REG | PWM 2 3 4 Control |

Table 34: BLE_RWBTLCTRL_REG (0x40000000)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-----------------|--|-------|
| 31 | r/w | MASTER_SOFT_RST | Reset the complete system except registers and timing generator, when written with a 1. Resets at 0 when action is performed. No action happens if it is written with 0. | 0x0 |

Table 34: BLE_RWBTLCTRL_REG (0x40000000)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-------------------|--|-------|
| 30 | r/w | MASTER_TGSOFT_RST | Reset the timing generator, when written with a 1. Resets at 0 when action is performed. No action happens if it is written with 0. | 0x0 |
| 29 | r/w | REG_SOFT_RST | Reset the complete register block, when written with a 1. Resets at 0 when action is performed. No action happens if it is written with 0. | 0x0 |
| 28:27 | - | - | Reserved | 0x0 |
| 26 | r/w | RFTEST_ABORT | Abort the current RF Testing defined as per CS-FORMAT when written with a 1. Resets at 0 when action is performed. No action happens if it is written with 0. Note that when RFTEST_ABORT is requested. 1) In case of infinite Tx, the Packet Controller FSM stops at the end of the current byte in process, and processes accordingly the packet CRC. 2) In case of Infinite Rx, the Packet Controller FSM either stops as the end of the current Packet reception (if Access address has been detected), or simply stop the processing switching off the RF. | 0x0 |
| 25 | r/w | ADVERT_ABORT | Abort the current Advertising event when written with a 1. Resets at 0 when action is performed. No action happens if it is written with 0. | 0x0 |
| 24 | r/w | SCAN_ABORT | Abort the current scan window when written with a 1. Resets at 0 when action is performed. No action happens if it is written with 0. | 0x0 |
| 23 | - | - | Reserved | 0x0 |
| 22 | r/w | MD_DSB | 0: Normal operation of MD bits management 1: Allow a single Tx/Rx exchange whatever the MD bits are | 0x0 |
| 21 | r/w | SN_DSB | 0: Normal operation of Sequence number 1: Sequence Number Management disabled: value forced by SW from Tx Descriptor value ignored in Rx, meaning that no SN error reported. | 0x0 |
| 20 | r/w | NESN_DSB | 0: Normal operation of Sequence number 1: Sequence Number Management disabled: value forced by SW from Tx Descriptor value ignored in Rx, meaning that no SN error reported. | 0x0 |
| 19 | r/w | CRYPT_DSB | 0: Normal operation. Encryption / Decryption enabled. 1: Encryption / Decryption disabled. Note that if CS-CRYPT_EN is set, then MIC is generated, and only data encryption is disabled, meaning data sent are plain data. | 0x0 |
| 18 | r/w | WHIT_DSB | 0: Normal operation. Whitening enabled. 1: Whitening disabled. | 0x0 |
| 17 | r/w | CRC_DSB | 0: Normal operation. CRC removed from data stream. 1: CRC stripping disabled on Rx packets, CRC replaced by 0x000 in Tx | 0x0 |
| 16 | r/w | HOP_REMAP_DSB | 0: Normal operation. Frequency Hopping Remapping algorithm enabled. 1: Frequency Hopping Remapping algorithm disabled | 0x0 |
| 15:13 | - | - | Reserved | 0x0 |

Table 34: BLE_RWBTLCNTL_REG (0x40000000)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|---------------|---|-------|
| 12 | r/w | TXWINDOFFSEL | Applicable only if device is in Initiator mode 0: Window Offset field in CONNECT_REQ comes from Tx Data Buffer 1: Window Offset field in CONNECT_REQ comes from Event Controller processing and is replaced in real time by Packet Controller | 0x0 |
| 11 | r/w | RXDESCPTRSEL | 0: Selects Rx Descriptor Pointer value from Control Structure 1: Selects Rx Descriptor Pointer value from CURRENTRX-DESCPTR register | 0x0 |
| 10 | r/w | ADVERRFILT_EN | Advertising Channels Error Filtering Enable control 0: BLE Core reports all errors to BLE Software 1: BLE Core reports only correctly received packet, without error to BLE Software | 0x0 |
| 9 | r/w | WLSYNC_EN | 0: WLAN synchronization pulse generation disabled 1: WLAN synchronization pulse generation enabled | 0x0 |
| 8 | r/w | RWBLE_EN | 0: Disable BLE Core Exchange Table pre-fetch mechanism. 1: Enable BLE Core Exchange table pre-fetch mechanism. | 0x0 |
| 7:4 | r/w | RXWINSZDEF | Default Rx Window size in us. Used when device a) is master connected b) performs its second receipt. 0 is not a valid value. Recommended value is 10. | 0x0 |
| 3 | - | - | Reserved | 0x0 |
| 2:0 | r/w | SYNCERR | Indicates the maximum number of errors allowed to recognize the synchronization word. | 0x0 |

Table 35: BLE_VERSION_REG (0x40000004)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|---|-------|
| 31:24 | r | TYP | BLE Core Type - 0x6 means BT4.0 (i.e correspond LL version assigned number) | 0x0 |
| 23:16 | r | REL | BLE Core version - Major release number.(Correspond to FS v1.11) | 0x0 |
| 15:8 | r | UPG | BLE Core upgrade - Upgrade number. (Correspond to FS v1.11) | 0x0 |
| 7:0 | r | BUILD | BLE Core Build - Build number | 0x0 |

Table 36: BLE_RWBTLCONF_REG (0x40000008)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-----------|--|-------|
| 31:30 | - | - | Reserved | 0x0 |
| 29:24 | r | ADD_WIDTH | Value of the BLE_ADDRESS_WIDTH parameter converted into binary. | 0xE |
| 23 | - | - | Reserved | 0x0 |
| 22:16 | r | RFIF | Supported radio interfaces. 0001000: on-chip radio others: reserved | 0x8 |
| 15:14 | - | - | Reserved | 0x0 |
| 13:8 | r | CLK_SEL | Operating Frequency (in MHz). This field is a copy of the BLE_CNTL2_REG[BLE_CLK_SEL] value. | 0x0 |
| 7:6 | - | - | Reserved | 0x0 |

Table 36: BLE_RWBLECONF_REG (0x40000008)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|----------|---|-------|
| 5 | r | DMMODE | 0: BLE Core is used as a standalone BLE device 1: BLE Core is used in a Dual Mode device | 0x0 |
| 4 | r | INTMODE | 0: Interrupts are edge level generated, i.e pulse. 1: Interrupts are trigger level generated, i.e stays active at 1 till acknowledgement | 0x1 |
| 3 | r | WLAN | 0: WLAN Coexistence mechanism not present 1: WLAN Coexistence mechanism present | 0x0 |
| 2 | r | USEDDBG | 0: Diagnostic port not instantiated 1: Diagnostic port instantiated | 0x1 |
| 1 | r | USECRYPT | 0: Encryption block not present 1: Encryption block present | 0x1 |
| 0 | r | BUSWIDTH | Processor bus width: 0: 16 bits 1: 32 bits | 0x1 |

Table 37: BLE_INTCNTL_REG (0x4000000C)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-------------------|---|-------|
| 31:16 | r/w | INTCSCNTL | Selection of the CS counter that generates an interrupt. For example, if INTCNTL[3] is set, an interrupt is sent each time CS counter equals 3. | 0xFF |
| 15 | r/w | CSCNTDEVMSK | CSCNT interrupt mask during event. This bit allows to enable CSCNT interrupt generation during events (i.e advertising, scanning, initiating, and connection) 0: CSCNT Interrupt not generated during events. 1: CSCNT Interrupt generated during events. | 0x1 |
| 14:9 | - | - | Reserved | 0x0 |
| 8 | r/w | RADIOCNTRLINTMSK | Radio Controller interrupt mask 0: Interrupt not generated 1: Interrupt generated | 0x0 |
| 7 | r/w | FINETGTIMINTMSK | Fine Target Timer Mask 0: Interrupt not generated 1: Interrupt generated | 0x0 |
| 6 | r/w | GROSSTGTIMINT-MSK | Gross Target Timer Mask 0: Interrupt not generated 1: Interrupt generated | 0x0 |
| 5 | r/w | ERRORINTMSK | Error Interrupt Mask 0: Interrupt not generated 1: Interrupt generated | 0x0 |
| 4 | r/w | CRYPTINTMSK | Encryption / Decryption Interrupt Mask 0: Interrupt not generated 1: Interrupt generated | 0x1 |
| 3 | r/w | EVENTINTMSK | End of event Interrupt Mask 0: Interrupt not generated 1: Interrupt generated | 0x1 |
| 2 | r/w | SLPINTMSK | Sleep Mode Interrupt Mask 0: Interrupt not generated 1: Interrupt generated | 0x1 |
| 1 | r/w | RXINTMSK | Rx Interrupt Mask 0: Interrupt not generated 1: Interrupt generated | 0x1 |

Table 37: BLE_INTCNTL_REG (0x4000000C)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-------------|--|-------|
| 0 | r/w | CSCNTINTMSK | 625usec Base Time Interrupt Mask 0: Interrupt not generated 1: Interrupt generated | 0x1 |

Table 38: BLE_INTSTAT_REG (0x40000010)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------------|---|-------|
| 31:9 | - | - | Reserved | 0x0 |
| 8 | r | RADIOCNTLINT-STAT | Radio Controller interrupt status 0: No Gross Target Timer interrupt. 1: A Gross Target Timer interrupt is pending. | 0x0 |
| 7 | r | FINETGTIMINTSTAT | Masked Fine Target Timer Error interrupt status 0: No Fine Target Timer interrupt. 1: A Fine Target Timer interrupt is pending. | 0x0 |
| 6 | r | GROSSTGTIMINT-STAT | Masked Gross Target Timer interrupt status 0: No Gross Target Timer interrupt. 1: A Gross Target Timer interrupt is pending. | 0x0 |
| 5 | r | ERRORINTSTAT | Masked Error interrupt status 0: No Error interrupt. 1: An Error interrupt is pending. | 0x0 |
| 4 | r | CRYPTINTSTAT | Masked Encryption/Decryption interrupt status 0: No Encryption / Decryption interrupt. 1: An Encryption / Decryption interrupt is pending. | 0x0 |
| 3 | r | EVENTINTSTAT | Masked End of Event interrupt status 0: No End of Advertising / Scanning / Connection interrupt. 1: An End of Advertising / Scanning / Connection interrupt is pending. | 0x0 |
| 2 | r | SLPINTSTAT | Masked Sleep interrupt status 0: No End of Sleep Mode interrupt. 1: An End of Sleep Mode interrupt is pending. | 0x0 |
| 1 | r | RXINTSTAT | Masked Packet Reception interrupt status 0: No Rx interrupt. 1: An Rx interrupt is pending. | 0x0 |
| 0 | r | CSCNTINTSTAT | Masked 625us base time reference interrupt status 0: No 625us Base Time interrupt. 1: A 625us Base Time interrupt is pending. | 0x0 |

Table 39: BLE_INTRAWSTAT_REG (0x40000014)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-----------------------|--|-------|
| 31:9 | - | - | Reserved | 0x0 |
| 8 | r | RADIOCNTLIN-TRAWSTAT | Radio Controller interrupt raw status 0: No Gross Target Timer interrupt. 1: A Gross Target Timer interrupt is pending. | 0x0 |
| 7 | r | FINETGTIMINTRAW-STAT | Fine Target Timer Error interrupt raw status 0: No Fine Target Timer interrupt. 1: A Fine Target Timer interrupt is pending. | 0x0 |
| 6 | r | GROSSTGTIMIN-TRAWSTAT | Gross Target Timer interrupt raw status 0: No Gross Target Timer interrupt. 1: A Gross Target Timer interrupt is pending. | 0x0 |

Table 39: BLE_INTRAWSTAT_REG (0x40000014)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------------|--|-------|
| 5 | r | ERRORINTRAW-STAT | Error interrupt raw status 0: No Error interrupt. 1: An Error interrupt is pending. | 0x0 |
| 4 | r | CRYPTINTRAWSTAT | Encryption/Decryption interrupt raw status 0: No Encryption / Decryption interrupt. 1: An Encryption / Decryption interrupt is pending. | 0x0 |
| 3 | r | EVENTINTRAWSTAT | End of Event interrupt raw status 0: No End of Advertising / Scanning / Connection interrupt. 1: An End of Advertising / Scanning / Connection interrupt is pending. | 0x0 |
| 2 | r | SLPINTRAWSTAT | Sleep interrupt raw status 0: No End of Sleep Mode interrupt. 1: An End of Sleep Mode interrupt is pending. | 0x0 |
| 1 | r | RXINTRAWSTAT | Packet Reception interrupt raw status 0: No Rx interrupt. 1: An Rx interrupt is pending. | 0x0 |
| 0 | r | CSCNTINTRAW-STAT | 625us base time reference interrupt raw status 0: No 625us Base Time interrupt. 1: A 625us Base Time interrupt is pending. | 0x0 |

Table 40: BLE_INTACK_REG (0x40000018)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-------------------|---|-------|
| 31:9 | - | - | Reserved | 0x0 |
| 8 | w | RADIOCNTRLINTACK | Radio Controller interrupt acknowledgement bit Software writing 1 acknowledges the Error interrupt. This bit resets RADIOCNTRLINTSTAT and RADIOCNTRLINTRAW-STAT flags. | 0x0 |
| 7 | w | FINETGTIMINTACK | Fine Target Timer interrupt acknowledgement bit Software writing 1 acknowledges the Error interrupt. This bit resets FINETGTIMINTSTAT and FINETGTIMINTRAWSTAT flags. | 0x0 |
| 6 | w | GROSSTGTIMIN-TACK | Gross Target Timer interrupt acknowledgement bit Software writing 1 acknowledges the Error interrupt. This bit resets GROSSTGTIMINTSTAT and GROSSTGTIMIN-TRAWSTAT flags. | 0x0 |
| 5 | w | ERRORINTACK | Error interrupt acknowledgement bit Software writing 1 acknowledges the Error interrupt. This bit resets ERRORINTSTAT and ERRORINTRAWSTAT flags. | 0x0 |
| 4 | w | CRYPTINTACK | Encryption/Decryption interrupt acknowledgement bit Software writing 1 acknowledges the Encryption / Decryption interrupt. This bit resets CRYPTINTSTAT and CRYPTIN-TRAWSTAT flags. | 0x0 |
| 3 | w | EVENTINTACK | End of Event interrupt acknowledgment bit Software writing 1 acknowledges the End of Advertising / Scanning / Connection interrupt. This bit resets SLPINT-STAT and SLPINTRAWSTAT flags. | 0x0 |
| 2 | w | SLPINTACK | End of Deep Sleep interrupt acknowledgment bit Software writing 1 acknowledges the End of Sleep Mode interrupt. This bit resets SLPINTSTAT and SLPINTRAW-STAT flags. | 0x0 |
| 1 | w | RXINTACK | Packet Reception interrupt acknowledgment bit Software writing 1 acknowledges the Rx interrupt. This bit resets RXINTSTAT and RXINTRAWSTAT flags. | 0x0 |

Table 40: BLE_INTACK_REG (0x40000018)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-------------|---|-------|
| 0 | w | CSCNTINTACK | 625us base time reference interrupt acknowledgment bit Software writing 1 acknowledges the CLKN interrupt. This bit resets CLKINTSTAT and CLKINTRAWSTAT flags. | 0x0 |

Table 41: BLE_BASETIMECNT_REG (0x4000001C)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-------------|---|-------|
| 31:27 | - | - | Reserved | 0x0 |
| 26:0 | r | BASETIMECNT | Value of the 625us base time reference counter. Updated each time BLE_SAMPLECLK_REG[SAMP] is written. Used by the SW in order to synchronize with the HW. | 0x0 |

Table 42: BLE_FINETIMECNT_REG (0x40000020)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|---------|--|-------|
| 31:10 | - | - | Reserved | 0x0 |
| 9:0 | r | FINECNT | Value of the current usec fine time reference counter. Updated each usec. Used by the SW in order to synchronize with the HW, and obtain a more precise sleep duration | 0x0 |

Table 43: BLE_BDADDR_L_REG (0x40000024)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------|--|-------|
| 31:0 | r/w | BDADDR_L | Bluetooth Low Energy Device Address. LSB part. | 0x0 |

Table 44: BLE_BDADDR_U_REG (0x40000028)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-----------|--|-------|
| 31:17 | - | - | Reserved | 0x0 |
| 16 | r/w | PRIV_NPUB | Bluetooth Low Energy Device Address privacy indicator 0: Public Bluetooth Device Address 1: Private Bluetooth Device Address | 0x0 |
| 15:0 | r/w | BDADDR_U | Bluetooth Low Energy Device Address. MSB part. | 0x0 |

Table 45: BLE_CURRENT_RX_DESCRIPTOR_REG (0x4000002C)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-----------------------|--|-------|
| 31:14 | - | - | Reserved | 0x0 |
| 13:0 | r/w | CURRENT_RX_DESCRIPTOR | Rx Descriptor Pointer that determines the starting point of the Receive Buffer Chained List. | 0x0 |

Table 46: BLE_DEEPSLEEP_CNTL_REG (0x40000030)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|------------|--|-------|
| 31 | r/w | EXTWKUPDSB | External Wake-Up disable 0: BLE Core can be woken by external wake-up 1: BLE Core cannot be woken up by external wake-up | 0x0 |
| 30:16 | - | - | Reserved | 0x0 |

Table 46: BLE_DEEPSLCNTL_REG (0x40000030)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------------|---|-------|
| 15 | r | DEEP_SLEEP_STAT | Indicator of current Deep Sleep clock mux status: 0: BLE Core is not yet in Deep Sleep Mode 1: BLE Core is in Deep Sleep Mode (only Low Power Clock is running) | 0x0 |
| 14:5 | - | - | Reserved | 0x0 |
| 4 | r/w | SOFT_WAKEUP_REQ | Wake Up Request from BLE Software. Applies when system is in Deep Sleep Mode. It wakes up the BLE Core when written with a 1. Always read as 0. No action happens if it is written with 0. | 0x0 |
| 3 | r/w | DEEP_SLEEP_CORR_EN | 625us base time reference integer and fractional part correction. Applies when system has been woken-up from Deep Sleep Mode. It enables Fine Counter and Base Time counter when written with a 1. Always read as 0. No action happens if it is written with 0. | 0x0 |
| 2 | r/w | DEEP_SLEEP_ON | 0: BLE Core in normal active mode 1: Request BLE Core to switch in deep sleep mode. This bit is reset on DEEP_SLEEP_STAT falling edge. | 0x0 |
| 1:0 | r/w | DEEP_SLEEP_IRQ_EN | Always set to "3" when DEEP_SLEEP_ON is set to "1". It controls the generation of BLE_WAKEUP_LP_IRQ. | 0x0 |

Table 47: BLE_DEEPSLWKUP_REG (0x40000034)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------|---|-------|
| 31:0 | r/w | DEEPSLTIME | Determines the time in Low Power Clock cycles to spend in Deep Sleep Mode before waking-up the device. This ensures a maximum of 37 hours and 16mn sleep mode capabilities at 32kHz. This ensures a maximum of 36 hours and 16mn sleep mode capabilities at 32.768kHz. If DEEPSLTIME is set to zero, the Deep Sleep Time duration is considered as infinite, and only wake up requests can restore active behavior BLE Software must ensure DEEPSLTIME value to be greater than 2 in order to cope with control resynchronization requirements | 0x0 |

Table 48: BLE_DEEPSLSTAT_REG (0x40000038)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-----------|---|-------|
| 31:0 | r | DEEPSLDUR | Actual duration of the last deep sleep phase measured in Low Power Clock cycles. DEEPSLDUR is set to zero at the beginning of the deep sleep phase, and is incremented at each Low Power Clock cycle until the end of the deep sleep phase. | 0x0 |

Table 49: BLE_ENBPRESET_REG (0x4000003C)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|--|-------|
| 31:21 | r/w | TWEXT | Minimum and recommended value is "TWIRQ_RESET + 1". In the case of wake-up due to an external wake-up request, TWEXT specifies the time delay in low power oscillator cycles to deassert BLE_WAKEUP_LP_IRQ. Refer also to GP_CONTROL_REG[BLE_WAKEUP_REQ]. Range is [0...64 ms] for 32kHz; [0...62.5 ms] for 32.768kHz | 0x0 |

Table 49: BLE_ENBPRESET_REG (0x4000003C)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-------------|--|-------|
| 20:10 | r/w | TWIRQ_SET | Minimum value is "TWIRQ_RESET + 1". Time in low power oscillator cycles to set BLE_WAKEUP_LP_IRQ before the BLE sleep timer expiration. Refer also to BLE_DEEPSLWKUP_REG[DEEPSLTIME]. Range is [0...64 ms] for 32kHz; [0...62.5 ms] for 32.768kHz | 0x0 |
| 9:0 | r/w | TWIRQ_RESET | Recommended value is 1. Time in low power oscillator cycles to reset BLE_WAKEUP_LP_IRQ before the BLE sleep timer expiration. Refer also to BLE_DEEPSLWKUP_REG[DEEPSLTIME]. Range is [0...32 ms] for 32kHz; [0...31.25 ms] for 32.768kHz. | 0x0 |

Table 50: BLE_FINECNTCORR_REG (0x40000040)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-------------|--|-------|
| 31:10 | - | - | Reserved | 0x0 |
| 9:0 | r/w | FINECNTCORR | Phase correction value for the 625usec reference counter (i.e Fine Counter) in us. | 0x0 |

Table 51: BLE_BASETIMECNTCORR_REG (0x40000044)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|------------------|-------------------------------------|-------|
| 31:27 | - | - | Reserved | 0x0 |
| 26:0 | r/w | BASETIMECNT-CORR | Base Time Counter correction value. | 0x0 |

Table 52: BLE_DIAGCNTL_REG (0x40000050)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|----------|--|-------|
| 31 | r/w | DIAG3_EN | 0: Disable diagnostic port 3 output. All outputs are set to 0. 1: Enable diagnostic port 3 output. | 0x0 |
| 30 | - | - | Reserved | 0x0 |
| 29:24 | r/w | DIAG3 | Only relevant when DIAG3_EN = 1. Selection of the outputs that must be driven to the diagnostic port 3. | 0x0 |
| 23 | r/w | DIAG2_EN | 0: Disable diagnostic port 2 output. All outputs are set to 0. 1: Enable diagnostic port 2 output. | 0x0 |
| 22 | - | - | Reserved | 0x0 |
| 21:16 | r/w | DIAG2 | Only relevant when DIAG2_EN = 1. Selection of the outputs that must be driven to the diagnostic port 2. | 0x0 |
| 15 | r/w | DIAG1_EN | 0: Disable diagnostic port 1 output. All outputs are set to 0. 1: Enable diagnostic port 1 output. | 0x0 |
| 14 | - | - | Reserved | 0x0 |
| 13:8 | r/w | DIAG1 | Only relevant when DIAG1_EN = 1. Selection of the outputs that must be driven to the diagnostic port 1. | 0x0 |
| 7 | r/w | DIAG0_EN | 0: Disable diagnostic port 0 output. All outputs are set to 0. 1: Enable diagnostic port 0 output. | 0x0 |
| 6 | - | - | Reserved | 0x0 |

Table 52: BLE_DIAGCNTL_REG (0x40000050)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|--|-------|
| 5:0 | r/w | DIAG0 | Only relevant when DIAG0_EN = 1. Selection of the outputs that must be driven to the diagnostic port 0. | 0x0 |

Table 53: BLE_DIAGSTAT_REG (0x40000054)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-----------|---|-------|
| 31:24 | r | DIAG3STAT | Directly connected to ble_dbg3[7:0] output. Debug use only. | 0x0 |
| 23:16 | r | DIAG2STAT | Directly connected to ble_dbg2[7:0] output. Debug use only. | 0x0 |
| 15:8 | r | DIAG1STAT | Directly connected to ble_dbg1[7:0] output. Debug use only. | 0x0 |
| 7:0 | r | DIAG0STAT | Directly connected to ble_dbg0[7:0] output. Debug use only. | 0x0 |

Table 54: BLE_DEBUGADDMAX_REG (0x40000058)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|----------------------------------|-------|
| 31:16 | - | - | Reserved | 0x0 |
| 15:0 | r/w | ADDMAX | Upper limit for the memory zone. | 0x0 |

Table 55: BLE_DEBUGADMIN_REG (0x4000005C)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|----------------------------------|-------|
| 31:16 | - | - | Reserved | 0x0 |
| 15:0 | r/w | ADMIN | Lower limit for the memory zone. | 0x0 |

Table 56: BLE_ERRORYPESTAT_REG (0x40000060)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-------------------|--|-------|
| 31:12 | - | - | Reserved | 0x0 |
| 11 | r | CSFORMAT_ERROR | Indicates whether CS-FORMAT has been programmed with an invalid value: this is a major software programming failure. 0: No error 1: Error occurred | 0x0 |
| 10 | r | CSTXPTR_ERROR | Indicates whether CS-TXPTR is null, this is a major software programming failure. 0: No error 1: Error occurred | 0x0 |
| 9 | - | - | Reserved | 0x0 |
| 8 | r | RADIO_EMACC_ERROR | Radio Controller Exchange Memory access error, happens when Exchange Memory access are not served in time and data are corrupted. 0: No error 1: Error occurred | 0x0 |
| 7 | r | LLCHMAP_ERROR | Link Layer Channel Map error, happens when actual number of CS-LLCHMAP bit set to one is different from CS-NBCH-GOOD at the beginning of Frequency Hopping process 0: No error 1: Error occurred | 0x0 |

Table 56: BLE_ERRORTYPESTAT_REG (0x40000060)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|---------------------|---|-------|
| 6 | r | IFS_UNDERRUN | Inter Frame Space Under run, occurs if IFS time is not enough to update and read Control Structure/Descriptors, and/or White List parsing is not finished and/or Decryption time is too long to be finished on time 0: No error 1: Error occurred | 0x0 |
| 5 | r | RXCRYPT_ERROR | Real Time Decryption Error, happens when decryption is not finished before IFS time 0: No error 1: Error occurred | 0x0 |
| 4 | r | WHITELIST_ERROR | White List Timeout Error, occurs if White List parsing is not finished on time 0: No error 1: Error occurred | 0x0 |
| 3 | r | APFM_ERROR | Anticipated Pre-Fetch Mechanism error, happens when 3 consecutive Exchange Table entry have been programmed, 0: no error 1: Error occurred | 0x0 |
| 2 | r | TXDESC_ERROR | Tx Descriptor Error, happens when fetched Tx Descriptor has TXDONE bit not set 0: No error 1: Error occurred | 0x0 |
| 1 | r | PKTCNTL_EMACC_ERROR | Packet Controller Exchange Memory access error, happens when Exchange Memory access are not served in time and Tx/Rx data are corrupted 0: No error 1: Error occurred | 0x0 |
| 0 | r | TXCRYPT_ERROR | Real Time Encryption Error, happens when encryption is not finished before Tx Payload has to be sent 0: No error 1: Error occurred | 0x0 |

Table 57: BLE_SWPROFILING_REG (0x40000064)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-----------|--|-------|
| 31:0 | r/w | SWPROFVAL | Software Profiling register: used by BLE Software for profiling purpose: this value is copied on Diagnostic port | 0x0 |

Table 58: BLE_RADIOCNTL0_REG (0x40000070)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-----------|--|--------|
| 31:23 | - | - | Reserved | 0x0 |
| 22 | r/w | DPCORR_EN | Enable the use of delayed DC compensated data path in Radio Correlator block 1: Enable 0: Disable Must be set to "0". | 0x0 |
| 21:0 | - | - | Reserved | 0x2102 |

Table 59: BLE_RADIOCNTL1_REG (0x40000074)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|-------------|-------|
| 31:21 | - | - | Reserved | 0x0 |

Table 59: BLE_RADIOCNTRL1_REG (0x40000074)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|---|-------|
| 20:16 | r/w | XRFSEL | Extended radio selection field, Must be set to '00011'. | 0x0 |
| 15:0 | - | - | Reserved | 0x0 |

Table 60: BLE_RADIOPWUPDN_REG (0x40000080)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-------------|--|-------|
| 31 | - | - | Reserved | 0x0 |
| 30:24 | r/w | RTRIP_DELAY | Defines round trip delay value. This value correspond to the addition of data latency in Tx and data latency in Rx. Value is in us. | 0x0 |
| 23:16 | r/w | RXPWRUP | This register holds the length in us of the Rx power up phase for the current radio device. Default value is 210 us (reset value). Operating range depends of the selected radio. | 0xD2 |
| 15:12 | - | - | Reserved | 0x0 |
| 11:8 | r/w | TXPWRDN | This register extends the length in us of the Tx power down phase for the current radio device. Default value is 3us (reset value). Operating range depends of the selected radio. | 0x3 |
| 7:0 | r/w | TXPWRUP | This register holds the length in us of the Tx power up phase for the current radio device. Default value is 210 us (reset value). Operating range depends of the selected radio. | 0xD2 |

Table 61: BLE_ADVCHMAP_REG (0x40000090)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------|--|-------|
| 31:3 | - | - | Reserved | 0x0 |
| 2:0 | r/w | ADVCHMAP | Advertising Channel Map, defined as per the advertising connection settings. Contains advertising channels index 37 to 39. If ADVCHMAP[i] equals: 0: Do not use data channel i+37. 1: Use data channel i+37. | 0x7 |

Table 62: BLE_ADVTIM_REG (0x400000A0)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|--|-------|
| 31:14 | - | - | Reserved | 0x0 |
| 13:0 | r/w | ADVINT | Advertising Packet Interval defines the time interval in between two ADV_XXX packet sent. Value is in usec. Value to program depends of the used Advertising Packet type and the device filtering policy. Please refer to Table 3-10 for details about ADVINT programming range. | 0x0 |

Table 63: BLE_ACTSCANSTAT_REG (0x400000A4)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|------------|---|-------|
| 31:25 | - | - | Reserved | 0x0 |
| 24:16 | r/w | BACKOFF | Active scan mode back-off counter initialization value. | 0x0 |
| 15:9 | - | - | Reserved | 0x0 |
| 8:0 | r/w | UPPERLIMIT | Active scan mode upper limit counter value. | 0x0 |

Table 64: BLE_WLPUBADDPTR_REG (0x400000B0)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-------------|---|-------|
| 31:16 | - | - | Reserved | 0x0 |
| 15:0 | r/w | WLPUBADDPTR | Start address pointer of the public devices white list. | 0x0 |

Table 65: BLE_WLPRIVADDPTR_REG (0x400000B4)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------------|--|-------|
| 31:16 | - | - | Reserved | 0x0 |
| 15:0 | r/w | WLPRIVADDPTR | Start address pointer of the private devices white list. | 0x0 |

Table 66: BLE_WLNBDEV_REG (0x400000B8)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-----------|--|-------|
| 31:16 | - | - | Reserved | 0x0 |
| 15:8 | r/w | NBPRIVDEV | Number of private devices in the white list. | 0x0 |
| 7:0 | r/w | NBPUBDEV | Number of public devices in the white list. | 0x0 |

Table 67: BLE_AESCNTRL_REG (0x400000C0)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-----------|--|-------|
| 31:30 | - | - | Reserved | 0x0 |
| 0 | r/w | AES_START | Writing a 1 starts AES-128 ciphering process. This bit is reset once the process is finished (i.e BLE_CRYPT_IRQ interrupt occurs, even masked) | 0x0 |

Table 68: BLE_AESKEY31_0_REG (0x400000C4)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------|--|-------|
| 31:0 | r/w | AESKEY31_0 | AES encryption 128-bit key. Bit 31 down to 0 | 0x0 |

Table 69: BLE_AESKEY63_32_REG (0x400000C8)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-------------|---|-------|
| 31:0 | r/w | AESKEY63_32 | AES encryption 128-bit key. Bit 63 down to 32 | 0x0 |

Table 70: BLE_AESKEY95_64_REG (0x400000CC)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-------------|---|-------|
| 31:0 | r/w | AESKEY95_64 | AES encryption 128-bit key. Bit 95 down to 64 | 0x0 |

Table 71: BLE_AESKEY127_96_REG (0x400000D0)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------|--|-------|
| 31:0 | r/w | AESKEY127_96 | AES encryption 128-bit key. Bit 127 down to 96 | 0x0 |

Table 72: BLE_AESPTR_REG (0x400000D4)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|-------------|-------|
| 31:16 | - | - | Reserved | 0x0 |

Table 72: BLE_AESPTR_REG (0x400000D4)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|--|-------|
| 15:0 | r/w | AESPTR | Pointer to the memory zone where the block to encrypt/decrypt is stored. | 0x0 |

Table 73: BLE_TXMICVAL_REG (0x400000D8)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------|---|-------|
| 31:0 | r | TXMICVAL | AES / CCM plain MIC value. Valid on BLE_CRYPT_IRQ interrupt (even masked) | 0x0 |

Table 74: BLE_RXMICVAL_REG (0x400000DC)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------|---|-------|
| 31:0 | r | RXMICVAL | AES / CCM plain MIC value. Valid on BLE_CRYPT_IRQ interrupt (even masked) | 0x0 |

Table 75: BLE_RFTESTCNTL_REG (0x400000E0)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-------------|--|-------|
| 31 | r/w | INFINITERX | Applicable for all frame format 0: Normal mode of operation 1: Infinite Rx window | 0x0 |
| 30:16 | - | - | Reserved | 0x0 |
| 15 | r/w | INFINITETX | Applicable for all frame format 0: Normal mode of operation. 1: Infinite Tx packet / Normal start of a packet but endless payload In case of infinite Tx payload, and when PRBS source is not selected, then RFTESTCNTL-TXLENGTH field provides the length of the pattern to repeat in the payload. | 0x0 |
| 14 | r/w | TXLENGTHSRC | Applicable only in Tx/Rx RF Test mode 0: Normal mode of operation: TXDESC-TXADVLEN controls the Tx packet payload size 1: Uses RFTESTCNTL-TXLENGTH packet length (can support up to 512 bytes transmit) | 0x0 |
| 13 | r/w | PRBSTYPE | Applicable only in Tx/Rx RF Test mode 0: Tx Packet Payload are PRBS9 type 1: Tx Packet Payload are PRBS15 type PRBS9 is defined as $p(x)=1+x^5+x^9$. The LFSR used for the PRBS9 generator must be initialized with 0x1FF value. PRBS15 is defined as $p(x)=1+x+x^2+x^{12}+x^{13}+x^{14}$. The LFSR used for the PRBS15 generator must be initialized with 0x7FFF value. | 0x0 |
| 12 | r/w | TXPLDSRC | Applicable only in Tx/Rx RF Test mode 0: Tx Packet Payload source is the Control Structure 1: Tx Packet Payload are PRBS generator | 0x0 |
| 11:9 | - | - | Reserved | 0x0 |
| 8:0 | r/w | TXLENGTH | Applicable only in Tx/Rx RF Test mode Tx packet length in number of byte | 0x0 |

Table 76: BLE_TIMGENCTL_REG (0x400000F0)

| Bit | Mode | Symbol | Description | Reset |
|------|------|---------------|--|-------|
| 31:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | PREFTECH_TIME | Defines Exchange Table pre-fetch instant in us | 0x0 |

Table 77: BLE_GROSSTIMTGT_REG (0x400000F4)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-------------|--|-------|
| 31:16 | - | - | Reserved | 0x0 |
| 15:0 | r/w | GROSSTARGET | Gross Timer Target value on which a BLE_GROSSTGTIM_IRQ must be generated. This timer has a precision of 10ms: interrupt is generated only when GROSSTARGET[15:0] = BASETIMECNT[19:4] and BASETIMECNT[3:0] = 0. | 0x0 |

Table 78: BLE_FINETIMTGT_REG (0x400000F8)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|------------|--|-------|
| 31:27 | - | - | Reserved | 0x0 |
| 26:0 | r/w | FINETARGET | Fine Timer Target value on which a BLE_FINETGTIM_IRQ must be generated. This timer has a precision of 625us: interrupt is generated only when FINETARGET = BASETIMECNT | 0x0 |

Table 79: BLE_SAMPLECLK_REG (0x400000FC)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|--|-------|
| 31:30 | - | - | Reserved | 0x0 |
| 0 | r/w | SAMP | Writing a 1 samples the Base Time Counter value in BASETIMECNT register. Resets at 0 when action is performed. | 0x0 |

Table 80: BLE_CNTL2_REG (0x40000200)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------------|---|-------|
| 31:22 | - | - | Reserved | 0x0 |
| 21 | r/w | BLE_RSSI_SEL | 0: Select Peak-hold RSSI value (default). 1: Select current Average RSSI value. | 0x0 |
| 20 | r | WAKEUPLPSTAT | The status of the BLE_WAKEUP_LP_IRQ. The Interrupt Service Routine of BLE_WAKEUP_LP_IRQ should return only when the WAKEUPLPSTAT is cleared. Note that BLE_WAKEUP_LP_IRQ is automatically acknowledged after the power up of the Radio Subsystem, plus one Low Power Clock period. | 0x0 |
| 19 | r/w | SW_RPL_SPI | Keep to 0. | 0x0 |
| 18 | r/w | BB_ONLY | Keep to 0. | 0x0 |
| 17 | r/w | RADIO_ONLY | Keep to 0. | 0x0 |
| 16:15 | - | - | Reserved | 0x0 |

Table 80: BLE_CNTL2_REG (0x40000200)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-------------------|---|-------|
| 14:9 | r/w | BLE_CLK_SEL | <p>BLE Clock Select.</p> <p>Specifies the BLE master clock absolute frequency in MHz. Typical values are 16 and 8.</p> <p>Value depends on the selected XTAL frequency and the value of CLK_RADIO_REG[BLE_DIV] bitfield. For example, if XTAL oscillates at 16MHz and CLK_RADIO_REG[BLE_DIV] = 1 (divide by 2), then BLE master clock frequency is 8MHz and BLE_CLK_SEL should be set to value 8.</p> <p>The selected BLE master clock frequency (affected by BLE_DIV and BLE_CLK_SEL) must be modified and set only during the initialization time, i.e. before setting BLE_RWBTL2_CNTL2_REG[RWBLE_EN] to 1. Refer also to BLE_RWBTL2_CONF_REG[CLK_SEL].</p> | 0x0 |
| 8 | r | RADIO_PWRDN_ALLOW | <p>This active high signal indicates when it is allowed for the BLE core (embedded in the Radio sub-System power domain) to be powered down.</p> <p>After the assertion of the BLE_DEEPSLCNTL2_REG[DEEP_SLEEP_ON] a hardware sequence based on the Low Power clock will cause the assertion of RADIO_PWRDN_ALLOW. The RADIO_PWRDN_ALLOW will be cleared to "0" when the BLE core exits from the sleep state, i.e. when the BLE_SLP_IRQ will be asserted.</p> | 0x0 |
| 7 | r/w | MON_LP_CLK | <p>The SW can only write a "0" to this bit.</p> <p>Whenever a positive edge of the low power clock used by the BLE Timers is detected, then the HW will automatically set this bit to "1". This functionality will not work if BLE Timer is in reset state (refer to CLK_RADIO_REG[BLE_LP_RESET]).</p> <p>This bit can be used for SW synchronization, to debug the low power clock, etc.</p> | 0x0 |
| 6 | r | BLE_CLK_STAT | <p>0: BLE uses low power clock</p> <p>1: BLE uses master clock</p> | 0x0 |
| 5 | r/w | DIAGPORT_REVERSE | <p>BLE/RADIO Diagnostic Port Reverse order.</p> <p>When this bit is "1", the mapping of the diagnostic bus DIAGPORT[7:0] (controlled by DIAGPORT_SEL) to GPIOs (controlled by Pxy_MODE_REG[PID]) is reversed. The mapping is:</p> <p>If "0" then DIAGPORT[7] is mapped to P0[7], etc. DIAGPORT[4] is mapped to P0[4], DIAGPORT[3] is mapped to P0[3] and P1[3], etc. and DIAGPORT[0] is mapped to P0[0] and P1[0].</p> <p>If "1" then DIAGPORT[7] is mapped to P0[0] and P1[0], etc. DIAGPORT[4] is mapped to P0[3] and P1[3], DIAGPORT[3] is mapped to P0[4], etc. and DIAGPORT[0] is mapped to P0[7].</p> | 0 |

Table 80: BLE_CNTL2_REG (0x40000200)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------------|---|-------|
| 4:3 | r/w | DIAGPORT_SEL | <p>BLE/RADIO Diagnostic Port Selection.</p> <p>Controls the multiplexing of the internal diagnostic signals towards the 8-bit diagnostic bus DIAGPORT[7:0]. The DIAGPORT[7:0] bit order may or may not be reversed by using the DIAGPORT_REVERSE bitfield and then it will be directed to the GPIOs P0[7:0] and P1[3:0]. (Note that the P1[3:0] diagnostic signals are the same with P0[3:0] signals.)</p> <p>The DIAGPORT[7:0] value, depending on the DIAGPORT_SEL value, is: 00: {BLE_DIAG2[7:5], BLE_DIAG1[4:3], BLE_DIAG0[2:0]} 01: {BLE_DIAG2[7:5], BLE_DIAG1[4:3], BLE_DIAG0[2], wakeup_lp_irq, deep_sleep_stat_32k} 10: RADIO_DIAG0[7:0] 11: RADIO_DIAG1[7:0]</p> | 0x0 |
| 2 | r/w | EMACCERRMSK | <p>Exchange Memory Access Error Mask: When cleared to "0" the EM_ACC_ERR will not cause an BLE_ERROR_IRQ interrupt. When set to "1" an BLE_ERROR_IRQ will be generated as long as EM_ACC_ERR is "1".</p> | 0x1 |
| 1 | w | EMACCERRACK | <p>Exchange Memory Access Error Acknowledge. When the SW writes a "1" to this bit then the EMACCERRSTAT bit will be cleared. When the SW writes "0" it will have no affect. The read value is always "0".</p> | 0x0 |
| 0 | r | EMACCERRSTAT | <p>Exchange Memory Access Error Status: The bit is read-only and can be cleared only by writing a "1" at EMACCERRACK bitfield. This bit will be set to "1" by the hardware when the controller will access an EM page that is not mapped according to the EM_MAPPING value. When this bit is "1" then the BLE_ERROR_IRQ will be asserted as long as EMACCERRMSK is "1".</p> | 0x0 |

Table 81: BLE_RF_DIAGIRQ_REG (0x40000204)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|----------------|-------------------------|-------|
| 31 | r | DIAGIRQ_STAT_3 | Same as DIAGIRQ_STAT_0. | 0 |
| 30 | r/w | DIAGIRQ_EDGE_3 | Same as DIAGIRQ_EDGE_0. | 0 |
| 29:27 | r/w | DIAGIRQ_BSEL_3 | Same as DIAGIRQ_BSEL_0. | 0 |
| 26:25 | r/w | DIAGIRQ_WSEL_3 | Same as DIAGIRQ_WSEL_0. | 0 |
| 24 | r/w | DIAGIRQ_MASK_3 | Same as DIAGIRQ_MASK_0. | 0 |
| 23 | r | DIAGIRQ_STAT_2 | Same as DIAGIRQ_STAT_0. | 0 |
| 22 | r/w | DIAGIRQ_EDGE_2 | Same as DIAGIRQ_EDGE_0. | 0 |
| 21:19 | r/w | DIAGIRQ_BSEL_2 | Same as DIAGIRQ_BSEL_0. | 0 |
| 18:17 | r/w | DIAGIRQ_WSEL_2 | Same as DIAGIRQ_WSEL_0. | 0 |
| 16 | r/w | DIAGIRQ_MASK_2 | Same as DIAGIRQ_MASK_0. | 0 |
| 15 | r | DIAGIRQ_STAT_1 | Same as DIAGIRQ_STAT_0. | 0 |
| 14 | r/w | DIAGIRQ_EDGE_1 | Same as DIAGIRQ_EDGE_0. | 0 |
| 13:11 | r/w | DIAGIRQ_BSEL_1 | Same as DIAGIRQ_BSEL_0. | 0 |
| 10:9 | r/w | DIAGIRQ_WSEL_1 | Same as DIAGIRQ_WSEL_0. | 0 |
| 8 | r/w | DIAGIRQ_MASK_1 | Same as DIAGIRQ_MASK_0. | 0 |

Table 81: BLE_RF_DIAGIRQ_REG (0x40000204)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|----------------|--|-------|
| 7 | r | DIAGIRQ_STAT_0 | Diagnostic IRQ Status 0 This bit is read only. It is automatically cleared to "0" on each read of the BLE_RF_DIAGIRQ_REG register. It is automatically asserted to "1" on each detection of the selected edge, of the selected bit, of the selected word. | 0 |
| 6 | r/w | DIAGIRQ_EDGE_0 | Diagnostic IRQ Edge 0 Selects the edge of the selected bit (refer to DIAGIRQ_BSEL_0) that will trigger the assertion of DIAGIRQ_STAT_0. If '0' then the positive edge is selected, when "1" the negative edge is selected. | 0 |
| 5:3 | r/w | DIAGIRQ_BSEL_0 | Diagnostic IRQ Bit Select 0 Selects the bit of the 8-bit bus (as selected by the DIAGIRQ_WSEL_0) that will be used for the IRQ generation. | 0 |
| 2:1 | r/w | DIAGIRQ_WSEL_0 | Diagnostic IRQ Word Select 0 Selects the 8-bit diagnostic bus that will be used for the IRQ generation. 00: Selects the BLE_DIAG0 01: Selects the BLE_DIAG1 10: Selects the RADIO_DIAG0 11: Selects the RADIO_DIAG1 | 0 |
| 0 | r/w | DIAGIRQ_MASK_0 | Diagnostic IRQ Mask 0 When set to "1" a BLE_RF_DIAG_IRQ will be generated on each rise of the DIAGIRQ_STAT_0 bit. When cleared to "0" no IRQ will be generated. | 0 |

Table 82: OTPC_MODE_REG (0x40008000)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|------------------------|--|-------|
| 31:30 | - | - | Reserved | 0x0 |
| 29:28 | r/w | OTPC_MODE_PRG_PORT_MUX | Selects the source that is connected to the prg_port port of the controller. 00 - {16'd0, BANDGAP_REG[15:0]} 01 - {RF_RSSI_COMP_CTRL_REG[15:0], 8'd0, RFIO_CTRL1_REG[7:0]} 10 - {3'd0, RF_LNA_CTRL3_REG[4:0], RF_LNA_CTRL2_REG[11:0], RF_LNA_CTRL1_REG[11:0]} 11 - {28'd0, RF_VCO_CTRL_REG[3:0]} See OTPC_MODE_PRG_PORT_SEL about the use of the prg_port | 0x0 |
| 27:9 | - | - | Reserved | 0x0 |
| 8 | r/w | OTPC_MODE_PRG_FAST | Defines the timing that will be used for all the programming activities (APROG, MPROG and TWR) 0 - Selects the normal timing 1 - Selects the fast timing | 0 |

Table 82: OTPC_MODE_REG (0x40008000)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------------------|--|-------|
| 7 | r/w | OTPC_MODE_PRG_PORT_SEL | Selects an alternative data source for the programming of the OTP macrocells, when the controller is configured in APROG mode. 0 - The fifo will be used as the data source. The fifo will be filled with a way defined by the register OTPC_MODE_USE_DMA. The number of words that will be programmed is defined by OTPC_NWORDS. 1 - Only one word will programmed. The value of the word is contained in the prg_port port of the controller. The values of the registers OTPC_MODE_USE_DMA, OTPC_NWORDS and the contents of the FIFO will not be used. | 0x0 |
| 6 | r/w | OTPC_MODE_TWO_CC_ACC | Defines the duration of each read from the OTP macrocells. 0 - Reads 16 bits of data every one clock cycle. 1 - Reads 16 bits of data every two clock cycles. | 0x0 |
| 5 | r/w | OTPC_MODE_FIFO_FLUSH | Writing 1, removes any content from the FIFO. This bit returns automatically to 0. | 0x0 |
| 4 | r/w | OTPC_MODE_USE_DMA | Selects the use of the dma, when the controller is configured in one of the modes: AREAD or APROG. 0 - DMA is not used. The data should be transferred from/to controller through OTPC_FFPRT_REG 1 - DMA is used. Data transfers from/to controller are performed automatically. The AHB base address should be configured in OTPC_AHBADR_REG before the selection of the mode. If programming of the OTPC_MODE_REG is performed through the serial interface, the OTPC_MODE_USE_DMA will be set to 0 automatically. If the controller is in APROG mode and the OTPC_MODE_PRG_PORT_SEL is enabled, the dma will stay inactive. | 0x0 |
| 3 | - | - | Reserved | 0x0 |
| 2:0 | r/w | OTPC_MODE_MODE | Defines the mode of operation of the OTPC controller. The encoding of the modes is as follows: 000 - STBY mode 001 - MREAD mode 010 - MPROG mode 011 - AREAD mode 100 - APROG mode 101 - Test mode. Reserved 110 - Test mode. Reserved 111 - Test mode. Reserved To manually move between modes, always return to STBY mode first. | 0x0 |

Table 83: OTPC_PCTRL_REG (0x40008004)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-----------------|---|-------|
| 31:28 | - | - | Reserved | 0x0 |
| 27 | r/w | OTPC_PCTRL_ENU | Enables the programming in the upper bank of the OTP. 0 - Programming sequence is not applied in the upper bank. 1 - Programming sequence is applied in the upper bank. | 0x0 |
| 26 | r/w | OTPC_PCTRL_BITU | Defines the value of the selected bit in the upper bank, after the programming sequence. | 0x0 |

Table 83: OTPC_PCTRL_REG (0x40008004)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-------------------|--|-------|
| 25 | r/w | OTPC_PCTRL_ENL | Enables the programming in the lower bank. 0 - The programming sequence is not applied in the lower bank. 1 - The programming sequence is applied in the lower bank. | 0x0 |
| 24 | r/w | OTPC_PCTRL_BITL | Defines the value of the selected bit in the lower bank, after the programming sequence. | 0x0 |
| 23 | r/w | OTPC_PCTRL_BSE LU | Selects between the U1 and U0 byte for the programming sequence in the upper bank. 0 - Program the U0 byte 1 - Program the U1 byte | 0x0 |
| 22:20 | r/w | OTPC_PCTRL_BAD RU | Selects the bit inside the Ux (x=0,1) byte, which will be programmed in the upper bank. | 0x0 |
| 19 | r/w | OTPC_PCTRL_BSE LL | Selects between the L1 and L0 byte for the programming sequence in the lower bank. 0 - Program the L0 byte 1 - Program the L1 byte | 0x0 |
| 18:16 | r/w | OTPC_PCTRL_BAD RL | Selects the bit inside the Lx (x=0,1) byte, which will be programmed in the lower bank. | 0x0 |
| 15:13 | - | - | Reserved | 0x0 |
| 12:0 | r/w | OTPC_PCTRL_WAD DR | Defines the address of a 32 bits word {U1,L1,U0,L0} in the macrocells, where one or two bits will be programmed. There are two macrocell banks, with 8 bits each. Each bank contribute with two memory positions for each 32 bits word. The Ux, Lx represent the bytes of the upper and lower bank respectively. | 0x0 |

Table 84: OTPC_STAT_REG (0x40008008)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|------------------|--|-------|
| 31:29 | - | - | Reserved | 0x0 |
| 28:16 | r | OTPC_STAT_NWORDS | Contains the current value of the words to be processed. | 0 |
| 15 | r | OTPC_STAT_TERR_U | Indicates the upper bank as the source of a test error. This value is valid when OTPC_STAT_TERROR is valid. 0 - There is no test error in the upper bank 1 - A test error has occurred in the upper bank | 0x0 |
| 14 | r | OTPC_STAT_TERR_L | Indicates the lower bank as the source of a test error. The value is valid when OTPC_STAT_TERROR is valid. 0 - There is no test error in the lower bank 1 - A test error has occurred in the lower bank | 0x0 |
| 13 | r | OTPC_STAT_PERR_U | Indicates the upper bank as the source of a programming error. The value is valid when OTPC_STAT_PERROR is valid. 0 - There is no programming error in the upper bank 1 - A programming error has occurred in the upper bank | 0x0 |
| 12 | r | OTPC_STAT_PERR_L | Indicates the lower bank as the source of a programming error. The value is valid when OTPC_STAT_PERROR is valid. 0 - There is no programming error in the lower bank 1 - A programming error has occurred in the lower bank | 0x0 |
| 11:8 | r | OTPC_STAT_FWORDS | Indicates the number of words which contained in the fifo of the controller. | 0x0 |
| 7:5 | - | - | Reserved | 0x0 |

Table 84: OTPC_STAT_REG (0x40008008)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-------------------|--|-------|
| 4 | r | OTPC_STAT_ARDY | Monitors the progress of read or programming operations while in the AREAD or APROG modes. 0 - The controller is busy while reading or programming (AREAD or APROG modes). 1 - The controller is not busy in AREAD or APROG mode. | 0x1 |
| 3 | r | OTPC_STAT_TERR OR | Indicates the result of a test sequence. Should be checked after the end of a TBLANK, TDEC and TWR mode (OTPC_STAT_TRDY= 1). 0 - The test sequence ends with no error. 1 - The test sequence has failed. | 0x0 |
| 2 | r | OTPC_STAT_TRDY | Indicates the state of a test mode. Should be used to monitor the progress of the TBLANK, TDEC and TWR modes. 0 - The controller is busy. A test mode is in progress. 1 - There is no active test mode. | 0x1 |
| 1 | r | OTPC_STAT_PERR OR | Indicates that an error has occurred during the bit-programming process. 0 - No error during the bit-programming process. 1 - The process of bit-programming failed. When the controller is in MPROG mode, this bit should be checked after the end of the programming process (OTPC_STAT_PRDY= 1). During APROG mode, the value of this field is normal to change periodically. Upon finishing the operation in the APROG mode (OTPC_STAT_ARDY= 1), this field indicates if the programming has failed or ended successfully. | 0x0 |
| 0 | r | OTPC_STAT_PRDY | Indicates the state of a bit-programming process. 0 - The controller is busy. A bit-programming is in progress 1 - The logic which performs bit-programming is idle. When the controller is in MPROG mode, this bit should be used to monitor the progress of a programming request. During APROG mode, the value of this field it is normal to changing periodically. | 0x1 |

Table 85: OTPC_AHBADR_REG (0x4000800C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-------------|--|-------|
| 31:2 | r/w | OTPC_AHBADR | The AHB address used by the AHB master interface of the controller (bits [31:2]). | 0x0 |
| 1:0 | - | - | Reserved | 0x0 |

Table 86: OTPC_CELADR_REG (0x40008010)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-------------|--|-------|
| 31:13 | - | - | Reserved | 0x0 |
| 12:0 | r/w | OTPC_CELADR | Defines a word address inside the macrocell. Used in modes AREAD and APROG and is automatically updated. | 0x0 |

Table 87: OTPC_NWORDS_REG (0x40008014)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|-------------|-------|
| 31:13 | - | - | Reserved | 0x0 |

Table 87: OTPC_NWORDS_REG (0x40008014)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-------------|--|-------|
| 12:0 | r/w | OTPC_NWORDS | The number of words (minus one) for reading/programming during the AREAD/APROG mode. If in APROG mode, and the OTPC_MODE_PRG_PORT_SEL is enabled (=1), this register will not be used and will stay unchanged. During mirroring, this register reflects the current amount of data that will be copied. It keeps its value until be written by the software with a new value. The number of the words that remaining to be processed by the controller is contained in the field OTPC_STAT_NWORDS. | 0x0 |

Table 88: OTPC_FFPRT_REG (0x40008018)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------|---|-------|
| 31:0 | r/w | OTPC_FFPRT | Provides access to the fifo through an access port. Write this register with the corresponding data, when the APROG mode is selected and the DMA is disabled. Read from this register the corresponding data, when the AREAD mode is selected and the DMA is disabled. Check OTPC_STAT_FWORDS register for data/space availability, before accessing the fifo. | 0x0 |

Table 89: OTPC_FFRD_REG (0x4000801C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-----------|---|-------|
| 31:0 | r | OTPC_FFRD | Contains the value read from the fifo, after a read of the OTPC_FFPRT_REG register. | 0x0 |

Table 90: CLK_AMBA_REG (0x50000000)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------|---|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7 | r/w | OTP_ENABLE | Clock enable for OTP controller | 0x0 |
| 6 | - | - | Reserved | 0x0 |
| 5:4 | r/w | PCLK_DIV | APB interface clock (PCLK). Divider is cascaded with HCLK_DIV. PCLK is HCLK divided by: 0x0: divide by 1 0x1: divide by 2 0x2: divide by 4 0x3: divide by 8 | 0x2 |
| 3:2 | - | - | Reserved | 0x0 |
| 1:0 | r/w | HCLK_DIV | AHB interface and microprocessor clock (HCLK). HCLK is source clock divided by: 0x0: divide by 1 0x1: divide by 2 0x2: divide by 4 0x3: divide by 8 | 0x2 |

Table 91: CLK_FREQ_TRIM_REG (0x50000002)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|-------------|-------|
| 15:11 | - | - | Reserved | 0x0 |

Table 91: CLK_FREQ_TRIM_REG (0x50000002)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------|---|-------|
| 10:8 | r/w | COARSE_ADJ | Xtal frequency course trimming register. 0x0: lowest frequency 0x7: highest frequency Increment or decrement the binary value with 1. Wait approximately 200 us to allow the adjustment to settle. | 0x0 |
| 7:0 | r/w | FINE_ADJ | Xtal frequency fine trimming register. 0x00: lowest frequency 0xFF: highest frequency | 0x0 |

Table 92: CLK_PER_REG (0x50000004)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-----------------|--|-------|
| 15 | r/w | QUAD_ENABLE | Enable the Quadrature clock | 0x0 |
| 14:12 | - | - | Reserved | 0x0 |
| 11 | r/w | SPI_ENABLE | Enable SPI clock | 0x0 |
| 10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | SPI_DIV | Division factor for SPI 0x0: divide by 1 0x1: divide by 2 0x2: divide by 4 0x3: divide by 8 | 0x0 |
| 7 | r/w | UART1_ENABLE | Enable UART1 clock | 0x0 |
| 6 | r/w | UART2_ENABLE | Enable UART2 clock | 0x0 |
| 5 | r/w | I2C_ENABLE | Enable I2C clock | 0x0 |
| 4 | r/w | WAKEUPCT_ENABLE | Enable Wakeup CaptureTimer clock | 0x0 |
| 3 | r/w | TMR_ENABLE | Enable TIMER0 and TIMER2 clock | 0x0 |
| 2 | - | - | Reserved | 0x0 |
| 1:0 | r/w | TMR_DIV | Division factor for TIMER0 0x0: divide by 1 0x1: divide by 2 0x2: divide by 4 0x3: divide by 8 | 0x0 |

Table 93: CLK_RADIO_REG (0x50000008)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------|--|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7 | r/w | BLE_ENABLE | Enable the BLE core clocks | 0x0 |
| 6 | r/w | BLE_LP_RESET | Reset for the BLE LP timer | 0x1 |
| 5:4 | r/w | BLE_DIV | Division factor for BLE core blocks 0x0: divide by 1 0x1: divide by 2 0x2: divide by 4 0x3: divide by 8 The programmed frequency should not be lower than 8 MHz and not faster than the programmed CPU clock frequency. Refer also to BLE_CNTL2_REG[BLE_CLK_SEL]. | 0x0 |
| 3 | r/w | RFCU_ENABLE | Enable the RF control Unit clock | 0x0 |
| 2 | - | - | Reserved | 0x0 |

Table 93: CLK_RADIO_REG (0x50000008)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|----------|--|-------|
| 1:0 | r/w | RFCU_DIV | Division factor for RF Control Unit 0x0: divide by 1 0x1: divide by 2 0x2: divide by 4 0x3: divide by 8 The programmed frequency must be exactly 8 MHz. | 0x0 |

Table 94: PMU_CTRL_REG (0x50000010)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-----------------|---|-------|
| 15:12 | - | - | Reserved | 0x0 |
| 11:8 | r/w | RETENTION_MODE | Select the retainability of the 4 retention RAM macros. '1' is retainable, '0' is power gated. (3) is RETRAM4 (2) is RETRAM3 (1) is RETRAM2 (0) is RETRAM1 | 0x0 |
| 7 | - | - | Reserved | 0x0 |
| 6 | - | - | Reserved | 0x0 |
| 5:4 | r/w | OTP_COPY_DIV | Sets the HCLK division during OTP mirroring | 0x0 |
| 2 | r/w | RADIO_SLEEP | Put the digital part of the radio in power down | 0x1 |
| 1 | r/w | PERIPH_SLEEP | Put all peripherals (I2C, UART, SPI, ADC) in power down | 0x1 |
| 0 | r/w | RESET_ON_WAKEUP | Perform a Hardware Reset after waking up. Booter will be started. | 0x0 |

Table 95: SYS_CTRL_REG (0x50000012)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-----------------|--|-------|
| 15 | w | SW_RESET | Writing a '1' to this bit will reset the device, except for: SYS_CTRL_REG CLK_FREQ_TRIM_REG ... | 0x0 |
| 9 | r/w | TIMEOUT_DISABLE | Disables timeout in Power statemachine. By default, the statemachine continues if after 2 ms the blocks are not started up. This can be read back from ANA_STATUS_REG. | 0x0 |
| 8 | - | - | Reserved | 0x0 |
| 7 | r/w | DEBUGGER_ENABLE | Enable the debugger. This bit is set by the booter according to the OTP header. If not set, the SWDIO and SW_CLK can be used as gpio ports. | 0x0 |
| 6 | r/w | OTPC_RESET_REQ | Reset request for the OTP controller. | 0x0 |
| 5 | r/w | PAD_LATCH_EN | Latches the control signals of the pads for state retention in power down mode. 0: Control signals are retained 1: Latch is transparent, pad can be recontrolled | 0x1 |
| 4 | r/w | OTP_COPY | Enables OTP to SysRAM copy action after waking up PD_SYS | 0x0 |
| 3 | r/w | CLK32_SOURCE | Sets the clock source of the 32 kHz clock 0 = RC-oscillator 1 = 32 kHz crystal oscillator | 0x0 |

Table 95: SYS_CTRL_REG (0x50000012)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 2 | r/w | RET_SYSRAM | Sets the development phase mode. The PD_SYS is not actually power gated (SysRAM is retained). No copy action to SysRAM is done when the system wakes up. For emulating startup time, the OTP_COPY bit still needs to be set. | 0x0 |
| 1:0 | r/w | REMAP_ADR0 | Controls which memory is located at address 0x0000 for execution. 0x0: ROM 0x1: OTP 0x2: SysRAM 0x3: RetRAM | 0x0 |

Table 96: SYS_STAT_REG (0x50000014)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-------------------|--|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7 | r | XTAL16_SETTLED | Indicates that XTAL16 has had > 2 ms of settle time | 0x0 |
| 6 | r | XTAL16_TRIM_READY | Indicates that XTAL trimming mechanism is ready, i.e. the trimming equals CLK_FREQ_TRIM_REG. | 0x1 |
| 5 | r | DBG_IS_UP | Indicates that PD_DBG is functional | 0x0 |
| 4 | r | DBG_IS_DOWN | Indicates that PD_DBG is in power down | 0x1 |
| 3 | r | PER_IS_UP | Indicates that PD_PER is functional | 0x0 |
| 2 | r | PER_IS_DOWN | Indicates that PD_PER is in power down | 0x1 |
| 1 | r | RAD_IS_UP | Indicates that PD_RAD is functional | 0x0 |
| 0 | r | RAD_IS_DOWN | Indicates that PD_RAD is in power down | 0x1 |

Table 97: TRIM_CTRL_REG (0x50000016)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-------------|---|-------|
| 7:4 | r/w | TRIM_TIME | Defines the delay between XTAL16M enable and applying the CLK_FREQ_TRIM_REG in steps of 250 us. 0x0: apply directly 0x1: wait between 0 and 250 us 0x2: wait between 250 us and 500 us etc. (Note 20) | 0xA |
| 3:0 | r/w | SETTLE_TIME | Defines the delay between applying CLK_FREQ_TRIM_REG and XTAL16_SETTLED in steps of 250 us. 0x0: XTAL16_SETTLED is set directly 0x1: wait between 0 and 250 us 0x2: wait between 250 us and 500 us etc. | 0x2 |

Note 20: The period duration of 250 us is derived by dividing the RC16M clock signal by 4000. Consequently, the period duration may vary over temperature.

Table 98: CLK_32K_REG (0x50000020)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|-------------|-------|
| 15:13 | - | - | Reserved | 0x0 |

Table 98: CLK_32K_REG (0x50000020)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------------------|---|-------|
| 12 | r/w | XTAL32K_DISABLE_AMPREG | Setting this bit disables the amplitude regulation of the XTAL32kHz oscillator. Set this bit to '1' for an external clock applied at XTAL32Kp. Keep this bit '0' with a crystal between XTAL32Kp and XTAL32Km. | 0x0 |
| 11:8 | r/w | RC32K_TRIM | Controls the frequency of the RC32K oscillator. 0x0: lowest frequency 0x7: default 0xF: highest frequency | 0x7 |
| 7 | r/w | RC32K_ENABLE | Enables the 32 kHz RC oscillator | 0x1 |
| 6:3 | r/w | XTAL32K_CUR | Bias current for the 32kHz XTAL oscillator. 0x0: minimum 0x3: default 0xF: maximum For each application there is an optimal setting for which the startup behavior is optimal. | 0x3 |
| 2:1 | r/w | XTAL32K_RBIAS | Setting for the bias resistor of the 32 kHz XTAL oscillator. 0x0: maximum 0x3: minimum Preferred setting will be provided by Dialog. | 0x2 |
| 0 | r/w | XTAL32K_ENABLE | Enables the 32 kHz XTAL oscillator | 0x0 |

Table 99: CLK_16M_REG (0x50000022)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------------------------------|--|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9 | r/w | XTAL16_NOISE_FILTER_ENABLE | Enables noise filter in 16 MHz crystal oscillator | 0x0 |
| 8 | r/w | XTAL16_BIAS_SAMPLE_HOLD_ENABLE | Enables Ibias sample/hold function in 16 MHz crystal oscillator. This bit should be set when the system wake up and reset before entering deep or extended sleep mode. | 0x0 |
| 7:5 | r/w | XTAL16_CUR_SET | Bias current for the 16 MHz XTAL oscillator. 0x0: minimum 0x7: maximum | 0x5 |
| 4:1 | r/w | RC16M_TRIM | Controls the frequency of the RC16M oscillator. 0x0: lowest frequency 0xF: highest frequency | 0x0 |
| 0 | r/w | RC16M_ENABLE | Enables the 16 MHz RC oscillator | 0x0 |

Table 100: CLK_RCX20K_REG (0x50000024)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|---------------|--|-------|
| 12 | r/w | RCX20K_SELECT | Selects RCX oscillator. 0 : RC32K oscillator 1: RCX oscillator | 0 |
| 11 | r/w | RCX20K_ENABLE | Enable the RCX oscillator | 0 |
| 10 | r/w | RCX20K_LOWF | Extra low frequency | 0 |
| 9:8 | r/w | RCX20K_BIAS | Bias control | 1 |
| 7:4 | r/w | RCX20K_NTC | Temperature control | 7 |

Table 100: CLK_RCX20K_REG (0x50000024)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-------------|--|-------|
| 3:0 | r/w | RCX20K_TRIM | Controls the frequency of the RCX oscillator. 0x0: lowest frequency 0x7: default 0xF: highest frequency | 8 |

Table 101: ANA_STATUS_REG (0x5000002A)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|------------------|--|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9 | r | BOOST_SELECTED | Indicates that DCDC is in boost mode | 0x0 |
| 8 | - | - | Reserved | 0x0 |
| 7 | r | BANDGAP_OK | Indicates that BANDGAP is OK | 0x1 |
| 6 | r | BOOST_VBAT_OK | Indicates that VBAT is above threshold while in BOOST converter mode. | 0x0 |
| 5 | r | LDO_ANA_OK | Indicates that LDO_ANA is in regulation. This LDO is used for the general-purpose ADC only | 0x0 |
| 4 | r | LDO_VDD_OK | Indicates that LDO_VDD is in regulation | 0x1 |
| 3 | r | LDO_OTP_OK | Indicates that LDO_OTP is in regulation | 0x0 |
| 2 | r | VDCDC_OK | Indicates that VDCDC is above threshold. | 0x0 |
| 1 | r | VBAT1V_OK | Indicates that VBAT1V is above threshold. | 0x0 |
| 0 | r | VBAT1V_AVAILABLE | Indicates that VBAT1V is available. | 0x0 |

Table 102: WKUP_CTRL_REG (0x50000100)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-----------------|--|-------|
| 15:14 | - | - | Reserved | 0x0 |
| 7 | r/w | WKUP_ENABLE_IRQ | 0: no interrupt will be enabled 1: if the event counter reaches the value set by WKUP_COMPARE_REG an IRQ will be generated | 0x0 |
| 6 | r/w | WKUP_SFT_KEYHIT | 0: no effect 1: emulate key hit. The event counter will increment by 1 (after debouncing if enabled). First make this bit 0 before any new key hit can be sensed. | 0x0 |
| 5:0 | r/w | WKUP_DEB_VALUE | Keyboard debounce time (N*1 ms with N = 1 to 63). 0x0: no debouncing 0x1 to 0x3F: 1 ms to 63 ms debounce time | 0x0 |

Table 103: WKUP_COMPARE_REG (0x50000102)

| Bit | Mode | Symbol | Description | Reset |
|------|------|---------|--|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | COMPARE | The number of events that have to be counted before the wakeup interrupt will be given | 0x0 |

Table 104: WKUP_RESET_IRQ_REG (0x50000104)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------|--|-------|
| 15:0 | w | WKUP_IRQ_RST | writing any value to this register will reset the interrupt. reading always returns 0. | 0x0 |

Table 105: WKUP_COUNTER_REG (0x50000106)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-------------|--|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r | EVENT_VALUE | This value represents the number of events that have been counted so far. It will be reset by resetting the interrupt. | 0x0 |

Table 106: WKUP_RESET_CNTR_REG (0x50000108)

| Bit | Mode | Symbol | Description | Reset |
|------|------|---------------|---|-------|
| 15:0 | w | WKUP_CNTR_RST | writing any value to this register will reset the event counter | 0x0 |

Table 107: WKUP_SELECT_P0_REG (0x5000010A)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|----------------|--|-------|
| 7:0 | r/w | WKUP_SELECT_P0 | 0: input P0x is not enabled for wakeup event counter 1: input P0x is enabled for wakeup event counter | 0x0 |

Table 108: WKUP_SELECT_P1_REG (0x5000010C)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|----------------|--|-------|
| 5:0 | r/w | WKUP_SELECT_P1 | 0: input P1x is not enabled for wakeup event counter 1: input P1x is enabled for wakeup event counter | 0x0 |

Table 109: WKUP_SELECT_P2_REG (0x5000010E)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|----------------|--|-------|
| 9:0 | r/w | WKUP_SELECT_P2 | 0: input P2x is not enabled for wakeup event counter 1: input P2x is enabled for wakeup event counter | 0x0 |

Table 110: WKUP_SELECT_P3_REG (0x50000110)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|----------------|--|-------|
| 7:0 | r/w | WKUP_SELECT_P3 | 0: input P3x is not enabled for wakeup event counter 1: input P3x is enabled for wakeup event counter | 0x0 |

Table 111: WKUP_POL_P0_REG (0x50000112)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-------------|---|-------|
| 7:0 | r/w | WKUP_POL_P0 | 0: enabled input P0x will increment the event counter if that input goes high 1: enabled input P0x will increment the event counter if that input goes low | 0x0 |

Table 112: WKUP_POL_P1_REG (0x50000114)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-------------|---|-------|
| 5:0 | r/w | WKUP_POL_P1 | 0: enabled input P1x will increment the event counter if that input goes high 1: enabled input P1x will increment the event counter if that input goes low | 0x0 |

Table 113: WKUP_POL_P2_REG (0x50000116)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-------------|---|-------|
| 9:0 | r/w | WKUP_POL_P2 | 0: enabled input P2x will increment the event counter if that input goes high 1: enabled input P2x will increment the event counter if that input goes low | 0x0 |

Table 114: WKUP_POL_P3_REG (0x50000118)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-------------|---|-------|
| 7:0 | r/w | WKUP_POL_P3 | 0: enabled input P3x will increment the event counter if that input goes high 1: enabled input P3x will increment the event counter if that input goes low | 0x0 |

Table 115: QDEC_CTRL_REG (0x50000200)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|---------------|--|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:3 | r/w | QD_IRQ_THRES | The number of events on either counter (X or Y) that need to be reached before an interrupt is generated. If 0 is written, then threshold is considered to be 1. | 0x2 |
| 2 | r | QD_IRQ_STATUS | Interrupt Status. If 1 an interrupt has occurred. | 0x0 |
| 1 | r/w | QD_IRQ_CLR | Writing 1 to this bit clears the interrupt. This bit is auto-cleared | 0x0 |
| 0 | r/w | QD_IRQ_MASK | 0: interrupt is masked 1: interrupt is enabled | 0x0 |

Table 116: QDEC_XCNT_REG (0x50000202)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-----------|--|-------|
| 15:0 | r | X_COUNTER | Contains a signed value of the events. Zero when channel is disabled | 0x0 |

Table 117: QDEC_YCNT_REG (0x50000204)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-----------|--|-------|
| 15:0 | r | Y_COUNTER | Contains a signed value of the events. Zero when channel is disabled | 0x0 |

Table 118: QDEC_CLOCKDIV_REG (0x50000206)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|---------------|---|-------|
| 9:0 | r/w | CLOCK_DIVIDER | Contains the number of the input clock cycles minus one, that are required to generate one logic clock cycle. | 0x0 |

Table 119: QDEC_CTRL2_REG (0x50000208)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|-------------|-------|
| 15:12 | - | - | Reserved | 0 |

Table 119: QDEC_CTRL2_REG (0x50000208)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------|---|-------|
| 11:8 | r/w | CHZ_PORT_SEL | Defines which GPIOs are mapped on Channel Z 0: none 1: P0[0] -> CHZ_A, P0[1] -> CHZ_B 2: P0[2] -> CHZ_A, P0[3] -> CHZ_B 3: P0[4] -> CHZ_A, P0[5] -> CHZ_B 4: P0[6] -> CHZ_A, P0[7] -> CHZ_B 5: P1[0] -> CHZ_A, P1[1] -> CHZ_B 6: P1[2] -> CHZ_A, P1[3] -> CHZ_B 7: P2[3] -> CHZ_A, P2[4] -> CHZ_B 8: P2[5] -> CHZ_A, P2[6] -> CHZ_B 9: P2[7] -> CHZ_A, P2[8] -> CHZ_B 10: P2[9] -> CHZ_A, P2[0] -> CHZ_B 11..15: None | 0 |
| 7:4 | r/w | CHY_PORT_SEL | Defines which GPIOs are mapped on Channel Y 0: none 1: P0[0] -> CHY_A, P0[1] -> CHY_B 2: P0[2] -> CHY_A, P0[3] -> CHY_B 3: P0[4] -> CHY_A, P0[5] -> CHY_B 4: P0[6] -> CHY_A, P0[7] -> CHY_B 5: P1[0] -> CHY_A, P1[1] -> CHY_B 6: P1[2] -> CHY_A, P1[3] -> CHY_B 7: P2[3] -> CHY_A, P2[4] -> CHY_B 8: P2[5] -> CHY_A, P2[6] -> CHY_B 9: P2[7] -> CHY_A, P2[8] -> CHY_B 10: P2[9] -> CHY_A, P2[0] -> CHY_B 11..15: None | 0 |
| 3:0 | r/w | CHX_PORT_SEL | Defines which GPIOs are mapped on Channel X 0: none 1: P0[0] -> CHX_A, P0[1] -> CHX_B 2: P0[2] -> CHX_A, P0[3] -> CHX_B 3: P0[4] -> CHX_A, P0[5] -> CHX_B 4: P0[6] -> CHX_A, P0[7] -> CHX_B 5: P1[0] -> CHX_A, P1[1] -> CHX_B 6: P1[2] -> CHX_A, P1[3] -> CHX_B 7: P2[3] -> CHX_A, P2[4] -> CHX_B 8: P2[5] -> CHX_A, P2[6] -> CHX_B 9: P2[7] -> CHX_A, P2[8] -> CHX_B 10: P2[9] -> CHX_A, P2[0] -> CHX_B 11..15: None | 0 |

Table 120: QDEC_ZCNT_REG (0x5000020A)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-----------|--|-------|
| 15:0 | r | Z_COUNTER | Contains a signed value of the events. Zero when channel is disabled | 0 |

Table 121: UART_RBR_THR_DLL_REG (0x50001000)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 121: UART_RBR_THR_DLL_REG (0x50001000)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-------------|---|-------|
| 7:0 | r/w | RBR_THR_DLL | <p>Receive Buffer Register: This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line Status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur. Transmit Holding Register: This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost. Divisor Latch (Low): This register makes up the lower 8-bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART. This register may only be accessed when the DLAB bit (LCR[7]) is set. The output baud rate is equal to the serial clock (sclk) frequency divided by sixteen times the value of the baud rate divisor, as follows: $\text{baud rate} = (\text{serial clock freq}) / (16 * \text{divisor})$ Note that with the Divisor Latch Registers (DLL and DLH) set to zero, the baud clock is disabled and no serial communications will occur. Also, once the DLL is set, at least 8 clock cycles of the slowest DW_apb_uart clock should be allowed to pass before transmitting or receiving data.</p> | 0x0 |

Table 122: UART_IER_DLH_REG (0x50001004)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------|--|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7 | r/w | PTIME_DLH7 | Interrupt Enable Register: PTIME, Programmable THRE Interrupt Mode Enable. This is used to enable/disable the generation of THRE Interrupt. 0 = disabled 1 = enabled Divisor Latch (High): Bit[7] of the 8 bit DLH register. | 0x0 |
| 6:4 | - | - | Reserved | 0x0 |
| 3 | r/w | EDSSI_DLH3 | Interrupt Enable Register: EDSSI, Enable Modem Status Interrupt. This is used to enable/disable the generation of Modem Status Interrupt. This is the fourth highest priority interrupt. 0 = disabled 1 = enabled Divisor Latch (High): Bit[3] of the 8 bit DLH register | 0x0 |

Table 122: UART_IER_DLH_REG (0x50001004)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 2 | r/w | ELSI_DHL2 | Interrupt Enable Register: ELSI, Enable Receiver Line Status Interrupt. This is used to enable/disable the generation of Receiver Line Status Interrupt. This is the highest priority interrupt. 0 = disabled 1 = enabled Divisor Latch (High): Bit[2] of the 8 bit DLH register. | 0x0 |
| 1 | r/w | ETBEI_DLH1 | Interrupt Enable Register: ETBEI, Enable Transmit Holding Register Empty Interrupt. This is used to enable/disable the generation of Transmitter Holding Register Empty Interrupt. This is the third highest priority interrupt. 0 = disabled 1 = enabled Divisor Latch (High): Bit[1] of the 8 bit DLH register. | 0x0 |
| 0 | r/w | ERBFI_DLH0 | Interrupt Enable Register: ERBFI, Enable Received Data Available Interrupt. This is used to enable/disable the generation of Received Data Available Interrupt and the Character Timeout Interrupt (if in FIFO mode and FIFO's enabled). These are the second highest priority interrupts. 0 = disabled 1 = enabled Divisor Latch (High): Bit[0] of the 8 bit DLH register. | 0x0 |

Table 123: UART_IIR_FCR_REG (0x50001008)

| Bit | Mode | Symbol | Description | Reset |
|------|------|---------|--|-------|
| 15:0 | r/w | IIR_FCR | <p>Interrupt Identification Register, reading this register; FIFO Control Register, writing to this register. Interrupt Identification Register: Bits[7:6], FIFO's Enabled (or FIFOSE): This is used to indicate whether the FIFO's are enabled or disabled. 00 = disabled. 11 = enabled. Bits[3:0], Interrupt ID (or IID): This indicates the highest priority pending interrupt which can be one of the following types: 0000 = modem status. 0001 = no interrupt pending. 0010 = THR empty. 0100 = received data available. 0110 = receiver line status. 0111 = busy detect. 1100 = character timeout. Bits[7:6], RCVR Trigger (or RT): This is used to select the trigger level in the receiver FIFO at which the Received Data Available Interrupt will be generated. In auto flow control mode it is used to determine when the rts_n signal will be de-asserted. It also determines when the dma_rx_req_n signal will be asserted when in certain modes of operation. The following trigger levels are supported: 00 = 1 character in the FIFO 01 = FIFO 1/4 full 10 = FIFO 1/2 full 11 = FIFO 2 less than full Bits[5:4], TX Empty Trigger (or TET): This is used to select the empty threshold level at which the THRE Interrupts will be generated when the mode is active. It also determines when the dma_tx_req_n signal will be asserted when in certain modes of operation. The following trigger levels are supported: 00 = FIFO empty 01 = 2 characters in the FIFO 10 = FIFO 1/4 full 11 = FIFO 1/2 full Bit[3], DMA Mode (or DMAM): This determines the DMA signalling mode used for the dma_tx_req_n and dma_rx_req_n output signals. 0 = mode 0 1 = mode 1 Bit[2], XMIT FIFO Reset (or XFIFOR): This resets the control portion of the transmit FIFO and treats the FIFO as empty. Note that this bit is 'self-clearing' and it is not necessary to clear this bit. Bit[1], RCVR FIFO Reset (or RFIFOR): This resets the control portion of the receive FIFO and treats the FIFO as empty. Note that this bit is 'self-clearing' and it is not necessary to clear this bit. Bit[0], FIFO Enable (or FIFOE): This enables/disables the transmit (XMIT) and receive (RCVR) FIFO's. Whenever the value of this bit is changed both the XMIT and RCVR controller portion of FIFO's will be reset.</p> | 0x0 |

Table 124: UART_LCR_REG (0x5000100C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-----------|--|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7 | r/w | UART_DLAB | <p>Divisor Latch Access Bit. This bit is used to enable reading and writing of the Divisor Latch register (DLL and DLH) to set the baud rate of the UART. This bit must be cleared after initial baud rate setup in order to access other registers.</p> | 0x0 |

Table 124: UART_LCR_REG (0x5000100C)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-----------|---|-------|
| 6 | r/w | UART_BC | Break Control Bit. This is used to cause a break condition to be transmitted to the receiving device. If set to one the serial output is forced to the spacing (logic 0) state. When not in Loopback Mode, as determined by MCR[4], the serial line is forced low until the Break bit is cleared. If active (MCR[6] set to one) the sir_out_n line is continuously pulsed. When in Loopback Mode, the break condition is internally looped back to the receiver and the sir_out_n line is forced low. | 0x0 |
| 5 | - | - | Reserved | 0x0 |
| 4 | r/w | UART_EPS | Even Parity Select. This is used to select between even and odd parity, when parity is enabled (PEN set to one). If set to one, an even number of logic 1s is transmitted or checked. If set to zero, an odd number of logic 1s is transmitted or checked. | 0x0 |
| 3 | r/w | UART_PEN | Parity Enable. This bit is used to enable and disable parity generation and detection in transmitted and received serial character respectively. 0 = parity disabled 1 = parity enabled | 0x0 |
| 2 | r/w | UART_STOP | Number of stop bits. This is used to select the number of stop bits per character that the peripheral transmits and receives. If set to zero, one stop bit is transmitted in the serial data. If set to one and the data bits are set to 5 (LCR[1:0] set to zero) one and a half stop bits is transmitted. Otherwise, two stop bits are transmitted. Note that regardless of the number of stop bits selected, the receiver checks only the first stop bit. 0 = 1 stop bit 1 = 1.5 stop bits when DLS (LCR[1:0]) is zero, else 2 stop bit | 0x0 |
| 1:0 | r/w | UART_DLS | Data Length Select. This is used to select the number of data bits per character that the peripheral transmits and receives. The number of bit that may be selected areas follows: 00 = 5 bits 01 = 6 bits 10 = 7 bits 11 = 8 bits | 0x0 |

Table 125: UART_MCR_REG (0x50001010)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-----------|---|-------|
| 15:7 | - | - | Reserved | 0x0 |
| 6 | r/w | UART_SIRE | SIR Mode Enable. This is used to enable/disable the IrDA SIR Mode features as described in "IrDA 1.0 SIR Protocol" on page 53. 0 = IrDA SIR Mode disabled 1 = IrDA SIR Mode enabled | 0x0 |
| 5 | r/w | UART_AFCE | Auto Flow Control Enable. When FIFOs are enabled and the Auto Flow Control Enable (AFCE) bit is set, hardware Auto Flow Control is enabled via CTS and RTS. 0 = Auto Flow Control Mode disabled 1 = Auto Flow Control Mode enabled | 0x0 |

Table 125: UART_MCR_REG (0x50001010)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-----------|---|-------|
| 4 | r/w | UART_LB | <p>LoopBack Bit.</p> <p>This is used to put the UART into a diagnostic mode for test purposes.</p> <p>If operating in UART mode (SIR_MODE not active, MCR[6] set to zero), data on the sout line is held high, while serial data output is looped back to the sin line, internally. In this mode all the interrupts are fully functional. Also, in loopback mode, the modem control inputs (dsr_n, cts_n, ri_n, dcd_n) are disconnected and the modem control outputs (dtr_n, rts_n, out1_n, out2_n) are looped back to the inputs, internally.</p> <p>If operating in infrared mode (SIR_MODE active, MCR[6] set to one), data on the sir_out_n line is held low, while serial data output is inverted and looped back to the sir_in line.</p> | 0x0 |
| 3 | r/w | UART_OUT2 | <p>OUT2.</p> <p>This is used to directly control the user-designated Output2 (out2_n) output. The value written to this location is inverted and driven out on out2_n, that is:</p> <p>0 = out2_n de-asserted (logic 1)</p> <p>1 = out2_n asserted (logic 0)</p> <p>Note that in Loopback mode (MCR[4] set to one), the out2_n output is held inactive high while the value of this location is internally looped back to an input.</p> | 0x0 |
| 2 | r/w | UART_OUT1 | <p>OUT1.</p> <p>This is used to directly control the user-designated Output1 (out1_n) output. The value written to this location is inverted and driven out on out1_n, that is:</p> <p>0 = out1_n de-asserted (logic 1)</p> <p>1 = out1_n asserted (logic 0)</p> <p>Note that in Loopback mode (MCR[4] set to one), the out1_n output is held inactive high while the value of this location is internally looped back to an input.</p> | 0x0 |
| 1 | r/w | UART_RTS | <p>Request to Send.</p> <p>This is used to directly control the Request to Send (rts_n) output. The Request To Send (rts_n) output is used to inform the modem or data set that the UART is ready to exchange data.</p> <p>When Auto Flow Control is disabled (MCR[5] set to zero), the rts_n signal is set low by programming MCR[1] (RTS) to a high. When Auto Flow Control is enabled (MCR[5] set to one) and FIFOs are enabled (FCR[0] set to one), the rts_n output is controlled in the same way, but is also gated with the receiver FIFO threshold trigger (rts_n is inactive high when above the threshold). The rts_n signal is de-asserted when MCR[1] is set low.</p> <p>Note that in Loopback mode (MCR[4] set to one), the rts_n output is held inactive (high) while the value of this location is internally looped back to an input.</p> | 0x0 |
| 0 | - | - | Reserved | 0x0 |

Table 126: UART_LSR_REG (0x50001014)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 126: UART_LSR_REG (0x50001014)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-----------|---|-------|
| 7 | r | UART_RFE | Receiver FIFO Error bit. This bit is only relevant when FIFOs are enabled (FCR[0] set to one). This is used to indicate if there is at least one parity error, framing error, or break indication in the FIFO. 0 = no error in RX FIFO 1 = error in RX FIFO This bit is cleared when the LSR is read and the character with the error is at the top of the receiver FIFO and there are no subsequent errors in the FIFO. | 0x0 |
| 6 | r | UART_TEMT | Transmitter Empty bit. If FIFOs enabled (FCR[0] set to one), this bit is set whenever the Transmitter Shift Register and the FIFO are both empty. If FIFOs are disabled, this bit is set whenever the Transmitter Holding Register and the Transmitter Shift Register are both empty. | 0x1 |
| 5 | r | UART_THRE | Transmit Holding Register Empty bit. If THRE mode is disabled (IER[7] set to zero) and regardless of FIFO's being implemented/enabled or not, this bit indicates that the THR or TX FIFO is empty. This bit is set whenever data is transferred from the THR or TX FIFO to the transmitter shift register and no new data has been written to the THR or TX FIFO. This also causes a THRE Interrupt to occur, if the THRE Interrupt is enabled. If both modes are active (IER[7] set to one and FCR[0] set to one respectively), the functionality is switched to indicate the transmitter FIFO is full, and no longer controls THRE interrupts, which are then controlled by the FCR[5:4] threshold setting. | 0x1 |
| 4 | r | UART_B1 | Break Interrupt bit. This is used to indicate the detection of a break sequence on the serial input data. If in UART mode (SIR_MODE == Disabled), it is set whenever the serial input, sin, is held in a logic '0' state for longer than the sum of start time + data bits + parity + stop bits. If in infrared mode (SIR_MODE == Enabled), it is set whenever the serial input, sir_in, is continuously pulsed to logic '0' for longer than the sum of start time + data bits + parity + stop bits. A break condition on serial input causes one and only one character, consisting of all zeros, to be received by the UART. In the FIFO mode, the character associated with the break condition is carried through the FIFO and is revealed when the character is at the top of the FIFO. Reading the LSR clears the BI bit. In the non-FIFO mode, the BI indication occurs immediately and persists until the LSR is read. | 0x0 |

Table 126: UART_LSR_REG (0x50001014)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|---------|---|-------|
| 3 | r | UART_FE | <p>Framing Error bit.</p> <p>This is used to indicate the occurrence of a framing error in the receiver. A framing error occurs when the receiver does not detect a valid STOP bit in the received data.</p> <p>In the FIFO mode, since the framing error is associated with a character received, it is revealed when the character with the framing error is at the top of the FIFO.</p> <p>When a framing error occurs, the UART tries to resynchronize. It does this by assuming that the error was due to the start bit of the next character and then continues receiving the other bit i.e. data, and/or parity and stop. It should be noted that the Framing Error (FE) bit (LSR[3]) is set if a break interrupt has occurred, as indicated by Break Interrupt (BI) bit (LSR[4]).</p> <p>0 = no framing error 1 = framing error</p> <p>Reading the LSR clears the FE bit.</p> | 0x0 |
| 2 | r | UART_PE | <p>Parity Error bit.</p> <p>This is used to indicate the occurrence of a parity error in the receiver if the Parity Enable (PEN) bit (LCR[3]) is set.</p> <p>In the FIFO mode, since the parity error is associated with a character received, it is revealed when the character with the parity error arrives at the top of the FIFO.</p> <p>It should be noted that the Parity Error (PE) bit (LSR[2]) is set if a break interrupt has occurred, as indicated by Break Interrupt (BI) bit (LSR[4]).</p> <p>0 = no parity error 1 = parity error</p> <p>Reading the LSR clears the PE bit.</p> | 0x0 |
| 1 | r | UART_OE | <p>Overrun error bit.</p> <p>This is used to indicate the occurrence of an overrun error. This occurs if a new data character was received before the previous data was read.</p> <p>In the non-FIFO mode, the OE bit is set when a new character arrives in the receiver before the previous character was read from the RBR. When this happens, the data in the RBR is overwritten. In the FIFO mode, an overrun error occurs when the FIFO is full and a new character arrives at the receiver. The data in the FIFO is retained and the data in the receive shift register is lost.</p> <p>0 = no overrun error 1 = overrun error</p> <p>Reading the LSR clears the OE bit.</p> | 0x0 |
| 0 | r | UART_DR | <p>Data Ready bit.</p> <p>This is used to indicate that the receiver contains at least one character in the RBR or the receiver FIFO.</p> <p>0 = no data ready 1 = data ready</p> <p>This bit is cleared when the RBR is read in non-FIFO mode, or when the receiver FIFO is empty, in FIFO mode.</p> | 0x0 |

Table 127: UART_MSR_REG (0x50001018)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 127: UART_MSR_REG (0x50001018)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-----------|--|-------|
| 7 | r | UART_DCD | <p>Data Carrier Detect.</p> <p>This is used to indicate the current state of the modem control line dcd_n. This bit is the complement of dcd_n. When the Data Carrier Detect input (dcd_n) is asserted it is an indication that the carrier has been detected by the modem or data set.</p> <p>0 = dcd_n input is de-asserted (logic 1) 1 = dcd_n input is asserted (logic 0)</p> <p>In Loopback Mode (MCR[4] set to one), DCD is the same as MCR[3] (Out2).</p> | 0x0 |
| 6 | r | UART_R1 | <p>Ring Indicator.</p> <p>This is used to indicate the current state of the modem control line ri_n. This bit is the complement of ri_n. When the Ring Indicator input (ri_n) is asserted it is an indication that a telephone ringing signal has been received by the modem or data set.</p> <p>0 = ri_n input is de-asserted (logic 1) 1 = ri_n input is asserted (logic 0)</p> <p>In Loopback Mode (MCR[4] set to one), RI is the same as MCR[2] (Out1).</p> | 0x0 |
| 5 | - | - | Reserved | 0x0 |
| 4 | r | UART_CTS | <p>Clear to Send.</p> <p>This is used to indicate the current state of the modem control line cts_n. This bit is the complement of cts_n. When the Clear to Send input (cts_n) is asserted it is an indication that the modem or data set is ready to exchange data with the UART Ctrl.</p> <p>0 = cts_n input is de-asserted (logic 1) 1 = cts_n input is asserted (logic 0)</p> <p>In Loopback Mode (MCR[4] = 1), CTS is the same as MCR[1] (RTS).</p> | 0x0 |
| 3 | r | UART_DDCD | <p>Delta Data Carrier Detect.</p> <p>This is used to indicate that the modem control line dcd_n has changed since the last time the MSR was read.</p> <p>0 = no change on dcd_n since last read of MSR 1 = change on dcd_n since last read of MSR</p> <p>Reading the MSR clears the DDCD bit. In Loopback Mode (MCR[4] = 1), DDCD reflects changes on MCR[3] (Out2). Note, if the DDCD bit is not set and the dcd_n signal is asserted (low) and a reset occurs (software or otherwise), then the DDCD bit is set when the reset is removed if the dcd_n signal remains asserted.</p> | 0x0 |
| 2 | r | UART_TERI | <p>Trailing Edge of Ring Indicator.</p> <p>This is used to indicate that a change on the input ri_n (from an active-low to an inactive-high state) has occurred since the last time the MSR was read.</p> <p>0 = no change on ri_n since last read of MSR 1 = change on ri_n since last read of MSR</p> <p>Reading the MSR clears the TERI bit. In Loopback Mode (MCR[4] = 1), TERI reflects when MCR[2] (Out1) has changed state from a high to a low.</p> | 0x0 |
| 1 | - | - | Reserved | 0x0 |

Table 127: UART_MSR_REG (0x50001018)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-----------|---|-------|
| 0 | r | UART_DCTS | Delta Clear to Send. This is used to indicate that the modem control line cts_n has changed since the last time the MSR was read. 0 = no change on cts_n since last read of MSR 1 = change on cts_n since last read of MSR Reading the MSR clears the DCTS bit. In Loopback Mode (MCR[4] = 1), DCTS reflects changes on MCR[1] (RTS). Note, if the DCTS bit is not set and the cts_n signal is asserted (low) and a reset occurs (software or otherwise), then the DCTS bit is set when the reset is removed if the cts_n signal remains asserted. | 0x0 |

Table 128: UART_SCR_REG (0x5000101C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------------------|---|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | UART_SCRATCH_P AD | This register is for programmers to use as a temporary storage space. It has no defined purpose in the UART Ctrl. | 0x0 |

Table 129: UART_LPDLL_REG (0x50001020)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------|---|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | UART_LPDLL | This register makes up the lower 8-bits of a 16-bit, read/write, Low Power Divisor Latch register that contains the baud rate divisor for the UART, which must give a baud rate of 115.2K. This is required for SIR Low Power (minimum pulse width) detection at the receiver. This register may be accessed only when the DLAB bit (LCR[7]) is set. The output low-power baud rate is equal to the serial clock (sclk) frequency divided by sixteen times the value of the baud rate divisor, as follows: Low power baud rate = (serial clock frequency)/(16* divisor) Therefore, a divisor must be selected to give a baud rate of 115.2K. NOTE: When the Low Power Divisor Latch registers (LPDLL and LPDLH) are set to 0, the low-power baud clock is disabled and no low-power pulse detection (or any pulse detection) occurs at the receiver. Also, once the LPDLL is set, at least eight clock cycles of the slowest UART Ctrl clock should be allowed to pass before transmitting or receiving data. | 0x0 |

Table 130: UART_LPDH_REG (0x50001024)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 130: UART_LPDLH_REG (0x50001024)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|--|-------|
| 7:0 | r/w | UART_LPDLH | <p>This register makes up the upper 8-bits of a 16-bit, read/write, Low Power Divisor Latch register that contains the baud rate divisor for the UART, which must give a baud rate of 115.2K. This is required for SIR Low Power (minimum pulse width) detection at the receiver. This register may be accessed only when the DLAB bit (LCR[7]) is set.</p> <p>The output low-power baud rate is equal to the serial clock (sclk) frequency divided by sixteen times the value of the baud rate divisor, as follows: Low power baud rate = (serial clock frequency)/(16* divisor) Therefore, a divisor must be selected to give a baud rate of 115.2K.</p> <p>NOTE: When the Low Power Divisor Latch registers (LPDLL and LPDLH) are set to 0, the low-power baud clock is disabled and no low-power pulse detection (or any pulse detection) occurs at the receiver. Also, once the LPDLH is set, at least eight clock cycles of the slowest UART Ctrl clock should be allowed to pass before transmitting or receiving data.</p> | 0x0 |

Table 131: UART_SRBR_STHR0_REG (0x50001030)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------|---|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 132: UART_SRBR_STHR1_REG (0x50001034)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------|---|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 133: UART_SRBR_STHR2_REG (0x50001038)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 133: UART_SRBR_STHR2_REG (0x50001038)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 134: UART_SRBR_STHR3_REG (0x5000103C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 134: UART_SRBR_STHR3_REG (0x5000103C)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 135: UART_SRBR_STHR4_REG (0x50001040)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 135: UART_SRBR_STHR4_REG (0x50001040)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 136: UART_SRBR_STHR5_REG (0x50001044)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 136: UART_SRBR_STHR5_REG (0x50001044)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 137: UART_SRBR_STHR6_REG (0x50001048)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 137: UART_SRBR_STHR6_REG (0x50001048)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 138: UART_SRBR_STHR7_REG (0x5000104C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 138: UART_SRBR_STHR7_REG (0x5000104C)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 139: UART_SRBR_STHR8_REG (0x50001050)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 139: UART_SRBR_STHR8_REG (0x50001050)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 140: UART_SRBR_STHR9_REG (0x50001054)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 140: UART_SRBR_STHR9_REG (0x50001054)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 141: UART_SRBR_STHR10_REG (0x50001058)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 141: UART_SRBR_STHR10_REG (0x50001058)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 142: UART_SRBR_STHR11_REG (0x5000105C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 142: UART_SRBR_STHR11_REG (0x5000105C)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 143: UART_SRBR_STHR12_REG (0x50001060)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 143: UART_SRBR_STHR12_REG (0x50001060)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 144: UART_SRBR_STHR13_REG (0x50001064)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 144: UART_SRBR_STHR13_REG (0x50001064)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 145: UART_SRBR_STHR14_REG (0x50001068)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 145: UART_SRBR_STHR14_REG (0x50001068)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 146: UART_SRBR_STHR15_REG (0x5000106C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 146: UART_SRBR_STHR15_REG (0x5000106C)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 147: UART_USR_REG (0x5000107C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-----------|---|-------|
| 15:5 | - | - | Reserved | 0x0 |
| 4 | r | UART_RFF | <p>Receive FIFO Full. This is used to indicate that the receive FIFO is completely full. 0 = Receive FIFO not full 1 = Receive FIFO Full This bit is cleared when the RX FIFO is no longer full.</p> | 0x0 |
| 3 | r | UART_RFNE | <p>Receive FIFO Not Empty. This is used to indicate that the receive FIFO contains one or more entries. 0 = Receive FIFO is empty 1 = Receive FIFO is not empty This bit is cleared when the RX FIFO is empty.</p> | 0x0 |
| 2 | r | UART_TFE | <p>Transmit FIFO Empty. This is used to indicate that the transmit FIFO is completely empty. 0 = Transmit FIFO is not empty 1 = Transmit FIFO is empty This bit is cleared when the TX FIFO is no longer empty.</p> | 0x1 |

Table 147: UART_USR_REG (0x5000107C)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-----------|--|-------|
| 1 | r | UART_TFNF | Transmit FIFO Not Full. This is used to indicate that the transmit FIFO is not full. 0 = Transmit FIFO is full 1 = Transmit FIFO is not full This bit is cleared when the TX FIFO is full. | 0x1 |
| 0 | - | - | Reserved | 0x0 |

Table 148: UART_TFL_REG (0x50001080)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------------------|---|-------|
| 15:0 | r | UART_TRANSMIT_FIFO_LEVEL | Transmit FIFO Level. This indicates the number of data entries in the transmit FIFO. | 0x0 |

Table 149: UART_RFL_REG (0x50001084)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-------------------------|---|-------|
| 15:0 | r | UART_RECEIVE_FIFO_LEVEL | Receive FIFO Level. This indicates the number of data entries in the receive FIFO. | 0x0 |

Table 150: UART_SRR_REG (0x50001088)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------|--|-------|
| 15:3 | - | - | Reserved | 0x0 |
| 2 | w | UART_XFR | XMIT FIFO Reset. This is a shadow register for the XMIT FIFO Reset bit (FCR[2]). This can be used to remove the burden on software having to store previously written FCR values (which are pretty static) just to reset the transmit FIFO. This resets the control portion of the transmit FIFO and treats the FIFO as empty. Note that this bit is 'self-clearing'. It is not necessary to clear this bit. | 0x0 |
| 1 | w | UART_RFR | RCVR FIFO Reset. This is a shadow register for the RCVR FIFO Reset bit (FCR[1]). This can be used to remove the burden on software having to store previously written FCR values (which are pretty static) just to reset the receive FIFO. This resets the control portion of the receive FIFO and treats the FIFO as empty. Note that this bit is 'self-clearing'. It is not necessary to clear this bit. | 0x0 |
| 0 | w | UART_UR | UART Reset. This asynchronously resets the UART Ctrl and synchronously removes the reset assertion. For a two clock implementation both pclk and sclk domains are reset. | 0x0 |

Table 151: UART_SRTS_REG (0x5000108C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:1 | - | - | Reserved | 0x0 |

Table 151: UART_SRTS_REG (0x5000108C)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-----------------------------|--|-------|
| 0 | r/w | UART_SHADOW_REQUEST_TO_SEND | <p>Shadow Request to Send.</p> <p>This is a shadow register for the RTS bit (MCR[1]), this can be used to remove the burden of having to perform a read-modify-write on the MCR. This is used to directly control the Request to Send (rts_n) output. The Request To Send (rts_n) output is used to inform the modem or data set that the UART Ctrl is ready to exchange data.</p> <p>When Auto Flow Control is disabled (MCR[5] = 0), the rts_n signal is set low by programming MCR[1] (RTS) to a high.</p> <p>When Auto Flow Control is enabled (MCR[5] = 1) and FIFOs are enabled (FCR[0] = 1), the rts_n output is controlled in the same way, but is also gated with the receiver FIFO threshold trigger (rts_n is inactive high when above the threshold).</p> <p>Note that in Loopback mode (MCR[4] = 1), the rts_n output is held inactive-high while the value of this location is internally looped back to an input.</p> | 0x0 |

Table 152: UART_SBCR_REG (0x50001090)

| Bit | Mode | Symbol | Description | Reset |
|------|------|---------------------------|---|-------|
| 15:1 | - | - | Reserved | 0x0 |
| 0 | r/w | UART_SHADOW_BREAK_CONTROL | <p>Shadow Break Control Bit.</p> <p>This is a shadow register for the Break bit (LCR[6]), this can be used to remove the burden of having to performing a read modify write on the LCR. This is used to cause a break condition to be transmitted to the receiving device.</p> <p>If set to one the serial output is forced to the spacing (logic 0) state. When not in Loopback Mode, as determined by MCR[4], the sout line is forced low until the Break bit is cleared.</p> <p>If SIR_MODE active (MCR[6] = 1) the sir_out_n line is continuously pulsed. When in Loopback Mode, the break condition is internally looped back to the receiver.</p> | 0x0 |

Table 153: UART_SDMAM_REG (0x50001094)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------------------|--|-------|
| 15:1 | - | - | Reserved | 0x0 |
| 0 | r/w | UART_SHADOW_DMA_MODE | <p>Shadow DMA Mode.</p> <p>This is a shadow register for the DMA mode bit (FCR[3]). This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the DMA Mode bit gets updated. This determines the DMA signalling mode used for the dma_tx_req_n and dma_rx_req_n output signals.</p> <p>0 = mode 0 1 = mode 1</p> | 0x0 |

Table 154: UART_SFE_REG (0x50001098)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:1 | - | - | Reserved | 0x0 |

Table 154: UART_SFE_REG (0x50001098)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-------------------------|---|-------|
| 0 | r/w | UART_SHADOW_FIFO_ENABLE | Shadow FIFO Enable. This is a shadow register for the FIFO enable bit (FCR[0]). This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the FIFO enable bit gets updated. This enables/disables the transmit (XMIT) and receive (RCVR) FIFOs. If this bit is set to zero (disabled) after being enabled then both the XMIT and RCVR controller portion of FIFOs are reset. | 0x0 |

Table 155: UART_SRT_REG (0x5000109C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------------------|--|-------|
| 15:2 | - | - | Reserved | 0x0 |
| 1:0 | r/w | UART_SHADOW_RCVR_TRIGGER | Shadow RCVR Trigger. This is a shadow register for the RCVR trigger bits (FCR[7:6]). This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the RCVR trigger bit gets updated. This is used to select the trigger level in the receiver FIFO at which the Received Data Available Interrupt is generated. It also determines when the dma_rx_req_n signal is asserted when DMA Mode (FCR[3]) = 1. The following trigger levels are supported: 00 = 1 character in the FIFO 01 = FIFO \hat{A} ¼ full 10 = FIFO \hat{A} ½ full 11 = FIFO 2 less than full | 0x0 |

Table 156: UART_STET_REG (0x500010A0)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------------------------|--|-------|
| 15:2 | - | - | Reserved | 0x0 |
| 1:0 | r/w | UART_SHADOW_TX_EMPTY_TRIGGER | Shadow TX Empty Trigger. This is a shadow register for the TX empty trigger bits (FCR[5:4]). This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the TX empty trigger bit gets updated. This is used to select the empty threshold level at which the THRE Interrupts are generated when the mode is active. The following trigger levels are supported: 00 = FIFO empty 01 = 2 characters in the FIFO 10 = FIFO \hat{A} ¼ full 11 = FIFO \hat{A} ½ full | 0x0 |

Table 157: UART_HTX_REG (0x500010A4)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:1 | - | - | Reserved | 0x0 |

Table 157: UART_HTX_REG (0x500010A4)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------------|---|-------|
| 0 | r/w | UART_HALT_TX | This register is use to halt transmissions for testing, so that the transmit FIFO can be filled by the master when FIFOs are implemented and enabled. 0 = Halt TX disabled 1 = Halt TX enabled Note, if FIFOs are implemented and not enabled, the setting of the halt TX register has no effect on operation. | 0x0 |

Table 158: UART_CPR_REG (0x500010F4)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|------------------------------|-------|
| 15:0 | r | CPR | Component Parameter Register | 0x0 |

Table 159: UART_UCV_REG (0x500010F8)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------------|----------------|
| 15:0 | r | UCV | Component Version | 0x33303 82A |

Table 160: UART_CTR_REG (0x500010FC)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------------------|----------------|
| 15:0 | r | CTR | Component Type Register | 0x44570 110 |

Table 161: UART2_RBR_THR_DLL_REG (0x50001100)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 161: UART2_RBR_THR_DLL_REG (0x50001100)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-------------|---|-------|
| 7:0 | r/w | RBR_THR_DLL | <p>Receive Buffer Register: This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line Status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur. Transmit Holding Register: This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost. Divisor Latch (Low): This register makes up the lower 8-bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART. This register may only be accessed when the DLAB bit (LCR[7]) is set. The output baud rate is equal to the serial clock (sclk) frequency divided by sixteen times the value of the baud rate divisor, as follows: baud rate = (serial clock freq) / (16 * divisor) Note that with the Divisor Latch Registers (DLL and DLH) set to zero, the baud clock is disabled and no serial communications will occur. Also, once the DLL is set, at least 8 clock cycles of the slowest DW_apb_uart clock should be allowed to pass before transmitting or receiving data.</p> | 0x0 |

Table 162: UART2_IER_DLH_REG (0x50001104)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------|--|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7 | r/w | PTIME_DLH7 | Interrupt Enable Register: PTIME, Programmable THRE Interrupt Mode Enable. This is used to enable/disable the generation of THRE Interrupt. 0 = disabled 1 = enabled Divisor Latch (High): Bit[7] of the 8 bit DLH register. | 0x0 |
| 6:4 | - | - | Reserved | 0x0 |
| 3 | r/w | EDSSI_DLH3 | Interrupt Enable Register: EDSSI, Enable Modem Status Interrupt. This is used to enable/disable the generation of Modem Status Interrupt. This is the fourth highest priority interrupt. 0 = disabled 1 = enabled Divisor Latch (High): Bit[3] of the 8 bit DLH register | 0x0 |

Table 162: UART2_IER_DLH_REG (0x50001104)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 2 | r/w | ELSI_DHL2 | Interrupt Enable Register: ELSI, Enable Receiver Line Status Interrupt. This is used to enable/disable the generation of Receiver Line Status Interrupt. This is the highest priority interrupt. 0 = disabled 1 = enabled Divisor Latch (High): Bit[2] of the 8 bit DLH register. | 0x0 |
| 1 | r/w | ETBEI_DLH1 | Interrupt Enable Register: ETBEI, Enable Transmit Holding Register Empty Interrupt. This is used to enable/disable the generation of Transmitter Holding Register Empty Interrupt. This is the third highest priority interrupt. 0 = disabled 1 = enabled Divisor Latch (High): Bit[1] of the 8 bit DLH register. | 0x0 |
| 0 | r/w | ERBFI_DLH0 | Interrupt Enable Register: ERBFI, Enable Received Data Available Interrupt. This is used to enable/disable the generation of Received Data Available Interrupt and the Character Timeout Interrupt (if in FIFO mode and FIFO's enabled). These are the second highest priority interrupts. 0 = disabled 1 = enabled Divisor Latch (High): Bit[0] of the 8 bit DLH register. | 0x0 |

Table 163: UART2_IIR_FCR_REG (0x50001108)

| Bit | Mode | Symbol | Description | Reset |
|------|------|---------|--|-------|
| 15:0 | r/w | IIR_FCR | <p>Interrupt Identification Register, reading this register; FIFO Control Register, writing to this register. Interrupt Identification Register: Bits[7:6], FIFO's Enabled (or FIFOSE): This is used to indicate whether the FIFO's are enabled or disabled. 00 = disabled. 11 = enabled. Bits[3:0], Interrupt ID (or IID): This indicates the highest priority pending interrupt which can be one of the following types: 0000 = modem status. 0001 = no interrupt pending. 0010 = THR empty. 0100 = received data available. 0110 = receiver line status. 0111 = busy detect. 1100 = character timeout. Bits[7:6], RCVR Trigger (or RT): This is used to select the trigger level in the receiver FIFO at which the Received Data Available Interrupt will be generated. In auto flow control mode it is used to determine when the rts_n signal will be de-asserted. It also determines when the dma_rx_req_n signal will be asserted when in certain modes of operation. The following trigger levels are supported: 00 = 1 character in the FIFO 01 = FIFO 1/4 full 10 = FIFO 1/2 full 11 = FIFO 2 less than full Bits[5:4], TX Empty Trigger (or TET): This is used to select the empty threshold level at which the THRE Interrupts will be generated when the mode is active. It also determines when the dma_tx_req_n signal will be asserted when in certain modes of operation. The following trigger levels are supported: 00 = FIFO empty 01 = 2 characters in the FIFO 10 = FIFO 1/4 full 11 = FIFO 1/2 full Bit[3], DMA Mode (or DMAM): This determines the DMA signalling mode used for the dma_tx_req_n and dma_rx_req_n output signals. 0 = mode 0 1 = mode 1 Bit[2], XMIT FIFO Reset (or XFIFOR): This resets the control portion of the transmit FIFO and treats the FIFO as empty. Note that this bit is 'self-clearing' and it is not necessary to clear this bit. Bit[1], RCVR FIFO Reset (or RFIFOR): This resets the control portion of the receive FIFO and treats the FIFO as empty. Note that this bit is 'self-clearing' and it is not necessary to clear this bit. Bit[0], FIFO Enable (or FIFOE): This enables/disables the transmit (XMIT) and receive (RCVR) FIFO's. Whenever the value of this bit is changed both the XMIT and RCVR controller portion of FIFO's will be reset.</p> | 0x0 |

Table 164: UART2_LCR_REG (0x5000110C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-----------|--|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7 | r/w | UART_DLAB | <p>Divisor Latch Access Bit. This bit is used to enable reading and writing of the Divisor Latch register (DLL and DLH) to set the baud rate of the UART. This bit must be cleared after initial baud rate setup in order to access other registers.</p> | 0x0 |

Table 164: UART2_LCR_REG (0x5000110C)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-----------|---|-------|
| 6 | r/w | UART_BC | Break Control Bit. This is used to cause a break condition to be transmitted to the receiving device. If set to one the serial output is forced to the spacing (logic 0) state. When not in Loopback Mode, as determined by MCR[4], the serial line is forced low until the Break bit is cleared. If active (MCR[6] set to one) the sir_out_n line is continuously pulsed. When in Loopback Mode, the break condition is internally looped back to the receiver and the sir_out_n line is forced low. | 0x0 |
| 5 | - | - | Reserved | 0x0 |
| 4 | r/w | UART_EPS | Even Parity Select. This is used to select between even and odd parity, when parity is enabled (PEN set to one). If set to one, an even number of logic 1s is transmitted or checked. If set to zero, an odd number of logic 1s is transmitted or checked. | 0x0 |
| 3 | r/w | UART_PEN | Parity Enable. This bit is used to enable and disable parity generation and detection in transmitted and received serial character respectively. 0 = parity disabled 1 = parity enabled | 0x0 |
| 2 | r/w | UART_STOP | Number of stop bits. This is used to select the number of stop bits per character that the peripheral transmits and receives. If set to zero, one stop bit is transmitted in the serial data. If set to one and the data bits are set to 5 (LCR[1:0] set to zero) one and a half stop bits is transmitted. Otherwise, two stop bits are transmitted. Note that regardless of the number of stop bits selected, the receiver checks only the first stop bit. 0 = 1 stop bit 1 = 1.5 stop bits when DLS (LCR[1:0]) is zero, else 2 stop bit | 0x0 |
| 1:0 | r/w | UART_DLS | Data Length Select. This is used to select the number of data bits per character that the peripheral transmits and receives. The number of bit that may be selected areas follows: 00 = 5 bits 01 = 6 bits 10 = 7 bits 11 = 8 bits | 0x0 |

Table 165: UART2_MCR_REG (0x50001110)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-----------|---|-------|
| 15:7 | - | - | Reserved | 0x0 |
| 6 | r/w | UART_SIRE | SIR Mode Enable. This is used to enable/disable the IrDA SIR Mode features as described in "IrDA 1.0 SIR Protocol" on page 53. 0 = IrDA SIR Mode disabled 1 = IrDA SIR Mode enabled | 0x0 |
| 5 | r/w | UART_AFCE | Auto Flow Control Enable. When FIFOs are enabled and the Auto Flow Control Enable (AFCE) bit is set, hardware Auto Flow Control is enabled via CTS and RTS. 0 = Auto Flow Control Mode disabled 1 = Auto Flow Control Mode enabled | 0x0 |

Table 165: UART2_MCR_REG (0x50001110)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-----------|---|-------|
| 4 | r/w | UART_LB | <p>LoopBack Bit.</p> <p>This is used to put the UART into a diagnostic mode for test purposes.</p> <p>If operating in UART mode (SIR_MODE not active, MCR[6] set to zero), data on the sout line is held high, while serial data output is looped back to the sin line, internally. In this mode all the interrupts are fully functional. Also, in loopback mode, the modem control inputs (dsr_n, cts_n, ri_n, dcd_n) are disconnected and the modem control outputs (dtr_n, rts_n, out1_n, out2_n) are looped back to the inputs, internally.</p> <p>If operating in infrared mode (SIR_MODE active, MCR[6] set to one), data on the sir_out_n line is held low, while serial data output is inverted and looped back to the sir_in line.</p> | 0x0 |
| 3 | r/w | UART_OUT2 | <p>OUT2.</p> <p>This is used to directly control the user-designated Output2 (out2_n) output. The value written to this location is inverted and driven out on out2_n, that is:</p> <p>0 = out2_n de-asserted (logic 1)</p> <p>1 = out2_n asserted (logic 0)</p> <p>Note that in Loopback mode (MCR[4] set to one), the out2_n output is held inactive high while the value of this location is internally looped back to an input.</p> | 0x0 |
| 2 | r/w | UART_OUT1 | <p>OUT1.</p> <p>This is used to directly control the user-designated Output1 (out1_n) output. The value written to this location is inverted and driven out on out1_n, that is:</p> <p>0 = out1_n de-asserted (logic 1)</p> <p>1 = out1_n asserted (logic 0)</p> <p>Note that in Loopback mode (MCR[4] set to one), the out1_n output is held inactive high while the value of this location is internally looped back to an input.</p> | 0x0 |
| 1 | r/w | UART_RTS | <p>Request to Send.</p> <p>This is used to directly control the Request to Send (rts_n) output. The Request To Send (rts_n) output is used to inform the modem or data set that the UART is ready to exchange data.</p> <p>When Auto Flow Control is disabled (MCR[5] set to zero), the rts_n signal is set low by programming MCR[1] (RTS) to a high. When Auto Flow Control is enabled (MCR[5] set to one) and FIFOs are enabled (FCR[0] set to one), the rts_n output is controlled in the same way, but is also gated with the receiver FIFO threshold trigger (rts_n is inactive high when above the threshold). The rts_n signal is de-asserted when MCR[1] is set low.</p> <p>Note that in Loopback mode (MCR[4] set to one), the rts_n output is held inactive (high) while the value of this location is internally looped back to an input.</p> | 0x0 |
| 0 | - | - | Reserved | 0x0 |

Table 166: UART2_LSR_REG (0x50001114)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 166: UART2_LSR_REG (0x50001114)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-----------|---|-------|
| 7 | r | UART_RFE | Receiver FIFO Error bit. This bit is only relevant when FIFOs are enabled (FCR[0] set to one). This is used to indicate if there is at least one parity error, framing error, or break indication in the FIFO. 0 = no error in RX FIFO 1 = error in RX FIFO This bit is cleared when the LSR is read and the character with the error is at the top of the receiver FIFO and there are no subsequent errors in the FIFO. | 0x0 |
| 6 | r | UART_TEMT | Transmitter Empty bit. If FIFOs enabled (FCR[0] set to one), this bit is set whenever the Transmitter Shift Register and the FIFO are both empty. If FIFOs are disabled, this bit is set whenever the Transmitter Holding Register and the Transmitter Shift Register are both empty. | 0x1 |
| 5 | r | UART_THRE | Transmit Holding Register Empty bit. If THRE mode is disabled (IER[7] set to zero) and regardless of FIFO's being implemented/enabled or not, this bit indicates that the THR or TX FIFO is empty. This bit is set whenever data is transferred from the THR or TX FIFO to the transmitter shift register and no new data has been written to the THR or TX FIFO. This also causes a THRE Interrupt to occur, if the THRE Interrupt is enabled. If both modes are active (IER[7] set to one and FCR[0] set to one respectively), the functionality is switched to indicate the transmitter FIFO is full, and no longer controls THRE interrupts, which are then controlled by the FCR[5:4] threshold setting. | 0x1 |
| 4 | r | UART_B1 | Break Interrupt bit. This is used to indicate the detection of a break sequence on the serial input data. If in UART mode (SIR_MODE == Disabled), it is set whenever the serial input, sin, is held in a logic '0' state for longer than the sum of start time + data bits + parity + stop bits. If in infrared mode (SIR_MODE == Enabled), it is set whenever the serial input, sir_in, is continuously pulsed to logic '0' for longer than the sum of start time + data bits + parity + stop bits. A break condition on serial input causes one and only one character, consisting of all zeros, to be received by the UART. In the FIFO mode, the character associated with the break condition is carried through the FIFO and is revealed when the character is at the top of the FIFO. Reading the LSR clears the BI bit. In the non-FIFO mode, the BI indication occurs immediately and persists until the LSR is read. | 0x0 |

Table 166: UART2_LSR_REG (0x50001114)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|---------|---|-------|
| 3 | r | UART_FE | <p>Framing Error bit.</p> <p>This is used to indicate the occurrence of a framing error in the receiver. A framing error occurs when the receiver does not detect a valid STOP bit in the received data.</p> <p>In the FIFO mode, since the framing error is associated with a character received, it is revealed when the character with the framing error is at the top of the FIFO.</p> <p>When a framing error occurs, the UART tries to resynchronize. It does this by assuming that the error was due to the start bit of the next character and then continues receiving the other bit i.e. data, and/or parity and stop. It should be noted that the Framing Error (FE) bit (LSR[3]) is set if a break interrupt has occurred, as indicated by Break Interrupt (BI) bit (LSR[4]).</p> <p>0 = no framing error 1 = framing error</p> <p>Reading the LSR clears the FE bit.</p> | 0x0 |
| 2 | r | UART_PE | <p>Parity Error bit.</p> <p>This is used to indicate the occurrence of a parity error in the receiver if the Parity Enable (PEN) bit (LCR[3]) is set.</p> <p>In the FIFO mode, since the parity error is associated with a character received, it is revealed when the character with the parity error arrives at the top of the FIFO.</p> <p>It should be noted that the Parity Error (PE) bit (LSR[2]) is set if a break interrupt has occurred, as indicated by Break Interrupt (BI) bit (LSR[4]).</p> <p>0 = no parity error 1 = parity error</p> <p>Reading the LSR clears the PE bit.</p> | 0x0 |
| 1 | r | UART_OE | <p>Overrun error bit.</p> <p>This is used to indicate the occurrence of an overrun error. This occurs if a new data character was received before the previous data was read.</p> <p>In the non-FIFO mode, the OE bit is set when a new character arrives in the receiver before the previous character was read from the RBR. When this happens, the data in the RBR is overwritten. In the FIFO mode, an overrun error occurs when the FIFO is full and a new character arrives at the receiver. The data in the FIFO is retained and the data in the receive shift register is lost.</p> <p>0 = no overrun error 1 = overrun error</p> <p>Reading the LSR clears the OE bit.</p> | 0x0 |
| 0 | r | UART_DR | <p>Data Ready bit.</p> <p>This is used to indicate that the receiver contains at least one character in the RBR or the receiver FIFO.</p> <p>0 = no data ready 1 = data ready</p> <p>This bit is cleared when the RBR is read in non-FIFO mode, or when the receiver FIFO is empty, in FIFO mode.</p> | 0x0 |

Table 167: UART2_MSR_REG (0x50001118)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 167: UART2_MSR_REG (0x50001118)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-----------|---|-------|
| 7 | r | UART_DCD | <p>Data Carrier Detect.</p> <p>This is used to indicate the current state of the modem control line dcd_n. This bit is the complement of dcd_n. When the Data Carrier Detect input (dcd_n) is asserted it is an indication that the carrier has been detected by the modem or data set.</p> <p>0 = dcd_n input is de-asserted (logic 1) 1 = dcd_n input is asserted (logic 0)</p> <p>In Loopback Mode (MCR[4] set to one), DCD is the same as MCR[3] (Out2).</p> | 0x0 |
| 6 | r | UART_R1 | <p>Ring Indicator.</p> <p>This is used to indicate the current state of the modem control line ri_n. This bit is the complement of ri_n. When the Ring Indicator input (ri_n) is asserted it is an indication that a telephone ringing signal has been received by the modem or data set.</p> <p>0 = ri_n input is de-asserted (logic 1) 1 = ri_n input is asserted (logic 0)</p> <p>In Loopback Mode (MCR[4] set to one), RI is the same as MCR[2] (Out1).</p> | 0x0 |
| 5 | - | - | Reserved | 0x0 |
| 4 | r | UART_CTS | <p>Clear to Send.</p> <p>This is used to indicate the current state of the modem control line cts_n. This bit is the complement of cts_n. When the Clear to Send input (cts_n) is asserted it is an indication that the modem or data set is ready to exchange data with the UART Ctrl.</p> <p>0 = cts_n input is de-asserted (logic 1) 1 = cts_n input is asserted (logic 0)</p> <p>In Loopback Mode (MCR[4] = 1), CTS is the same as MCR[1] (RTS).</p> | 0x0 |
| 3 | r | UART_DDCD | <p>Delta Data Carrier Detect.</p> <p>This is used to indicate that the modem control line dcd_n has changed since the last time the MSR was read.</p> <p>0 = no change on dcd_n since last read of MSR 1 = change on dcd_n since last read of MSR</p> <p>Reading the MSR clears the DDCD bit. In Loopback Mode (MCR[4] = 1), DDCD reflects changes on MCR[3] (Out2). Note, if the DDCD bit is not set and the dcd_n signal is asserted (low) and a reset occurs (software or otherwise), then the DDCD bit is set when the reset is removed if the dcd_n signal remains asserted.</p> | 0x0 |
| 2 | r | UART_TERI | <p>Trailing Edge of Ring Indicator.</p> <p>This is used to indicate that a change on the input ri_n (from an active-low to an inactive-high state) has occurred since the last time the MSR was read.</p> <p>0 = no change on ri_n since last read of MSR 1 = change on ri_n since last read of MSR</p> <p>Reading the MSR clears the TERI bit. In Loopback Mode (MCR[4] = 1), TERI reflects when MCR[2] (Out1) has changed state from a high to a low.</p> | 0x0 |
| 1 | - | - | Reserved | 0x0 |

Table 167: UART2_MSR_REG (0x50001118)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-----------|---|-------|
| 0 | r | UART_DCTS | Delta Clear to Send. This is used to indicate that the modem control line cts_n has changed since the last time the MSR was read. 0 = no change on cts_n since last read of MSR 1 = change on cts_n since last read of MSR Reading the MSR clears the DCTS bit. In Loopback Mode (MCR[4] = 1), DCTS reflects changes on MCR[1] (RTS). Note, if the DCTS bit is not set and the cts_n signal is asserted (low) and a reset occurs (software or otherwise), then the DCTS bit is set when the reset is removed if the cts_n signal remains asserted. | 0x0 |

Table 168: UART2_SCR_REG (0x5000111C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------------------|---|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | UART_SCRATCH_P AD | This register is for programmers to use as a temporary storage space. It has no defined purpose in the UART Ctrl. | 0x0 |

Table 169: UART2_LPDLL_REG (0x50001120)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------|---|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | UART_LPDLL | This register makes up the lower 8-bits of a 16-bit, read/write, Low Power Divisor Latch register that contains the baud rate divisor for the UART, which must give a baud rate of 115.2K. This is required for SIR Low Power (minimum pulse width) detection at the receiver. This register may be accessed only when the DLAB bit (LCR[7]) is set. The output low-power baud rate is equal to the serial clock (sclk) frequency divided by sixteen times the value of the baud rate divisor, as follows: Low power baud rate = (serial clock frequency)/(16* divisor) Therefore, a divisor must be selected to give a baud rate of 115.2K. NOTE: When the Low Power Divisor Latch registers (LPDLL and LPDLH) are set to 0, the low-power baud clock is disabled and no low-power pulse detection (or any pulse detection) occurs at the receiver. Also, once the LPDLL is set, at least eight clock cycles of the slowest UART Ctrl clock should be allowed to pass before transmitting or receiving data. | 0x0 |

Table 170: UART2_LPDLH_REG (0x50001124)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 170: UART2_LPDLH_REG (0x50001124)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|--|-------|
| 7:0 | r/w | UART_LPDLH | <p>This register makes up the upper 8-bits of a 16-bit, read/write, Low Power Divisor Latch register that contains the baud rate divisor for the UART, which must give a baud rate of 115.2K. This is required for SIR Low Power (minimum pulse width) detection at the receiver. This register may be accessed only when the DLAB bit (LCR[7]) is set.</p> <p>The output low-power baud rate is equal to the serial clock (sclk) frequency divided by sixteen times the value of the baud rate divisor, as follows: $\text{Low power baud rate} = (\text{serial clock frequency}) / (16 * \text{divisor})$ Therefore, a divisor must be selected to give a baud rate of 115.2K.</p> <p>NOTE: When the Low Power Divisor Latch registers (LPDLL and LPDLH) are set to 0, the low-power baud clock is disabled and no low-power pulse detection (or any pulse detection) occurs at the receiver. Also, once the LPDLH is set, at least eight clock cycles of the slowest UART Ctrl clock should be allowed to pass before transmitting or receiving data.</p> | 0x0 |

Table 171: UART2_SRBR_STHR0_REG (0x50001130)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------|---|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 172: UART2_SRBR_STHR1_REG (0x50001134)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------|---|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 173: UART2_SRBR_STHR2_REG (0x50001138)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 173: UART2_SRBR_STHR2_REG (0x50001138)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 174: UART2_SRBR_STHR3_REG (0x5000113C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 174: UART2_SRBR_STHR3_REG (0x5000113C)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 175: UART2_SRBR_STHR4_REG (0x50001140)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 175: UART2_SRBR_STHR4_REG (0x50001140)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 176: UART2_SRBR_STHR5_REG (0x50001144)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 176: UART2_SRBR_STHR5_REG (0x50001144)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 177: UART2_SRBR_STHR6_REG (0x50001148)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 177: UART2_SRBR_STHR6_REG (0x50001148)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 178: UART2_SRBR_STHR7_REG (0x5000114C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 178: UART2_SRBR_STHR7_REG (0x5000114C)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 179: UART2_SRBR_STHR8_REG (0x50001150)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 179: UART2_SRBR_STHR8_REG (0x50001150)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 180: UART2_SRBR_STHR9_REG (0x50001154)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 180: UART2_SRBR_STHR9_REG (0x50001154)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 181: UART2_SRBR_STHR10_REG (0x50001158)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 181: UART2_SRBR_STHR10_REG (0x50001158)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 182: UART2_SRBR_STHR11_REG (0x5000115C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 182: UART2_SRBR_STHR11_REG (0x5000115C)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 183: UART2_SRBR_STHR12_REG (0x50001160)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 183: UART2_SRBR_STHR12_REG (0x50001160)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 184: UART2_SRBR_STHR13_REG (0x50001164)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 184: UART2_SRBR_STHR13_REG (0x50001164)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 185: UART2_SRBR_STHR14_REG (0x50001168)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 185: UART2_SRBR_STHR14_REG (0x50001168)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 186: UART2_SRBR_STHR15_REG (0x5000116C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:8 | - | - | Reserved | 0x0 |

Table 186: UART2_SRBR_STHR15_REG (0x5000116C)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|---|-------|
| 7:0 | r/w | SRBR_STHRX | <p>Shadow Receive Buffer Register x: This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.</p> <p>Shadow Transmit Holding Register 0: This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled (FCR[0] set to zero) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled (FCR[0] set to one) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> | 0x0 |

Table 187: UART2_USR_REG (0x5000117C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-----------|---|-------|
| 15:5 | - | - | Reserved | 0x0 |
| 4 | r | UART_RFF | <p>Receive FIFO Full. This is used to indicate that the receive FIFO is completely full. 0 = Receive FIFO not full 1 = Receive FIFO Full This bit is cleared when the RX FIFO is no longer full.</p> | 0x0 |
| 3 | r | UART_RFNE | <p>Receive FIFO Not Empty. This is used to indicate that the receive FIFO contains one or more entries. 0 = Receive FIFO is empty 1 = Receive FIFO is not empty This bit is cleared when the RX FIFO is empty.</p> | 0x0 |
| 2 | r | UART_TFE | <p>Transmit FIFO Empty. This is used to indicate that the transmit FIFO is completely empty. 0 = Transmit FIFO is not empty 1 = Transmit FIFO is empty This bit is cleared when the TX FIFO is no longer empty.</p> | 0x1 |

Table 187: UART2_USR_REG (0x5000117C)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-----------|--|-------|
| 1 | r | UART_TFNF | Transmit FIFO Not Full. This is used to indicate that the transmit FIFO is not full. 0 = Transmit FIFO is full 1 = Transmit FIFO is not full This bit is cleared when the TX FIFO is full. | 0x1 |
| 0 | - | - | Reserved | 0x0 |

Table 188: UART2_TFL_REG (0x50001180)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------------------|---|-------|
| 15:0 | r | UART_TRANSMIT_FIFO_LEVEL | Transmit FIFO Level. This indicates the number of data entries in the transmit FIFO. | 0x0 |

Table 189: UART2_RFL_REG (0x50001184)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-------------------------|---|-------|
| 15:0 | r | UART_RECEIVE_FIFO_LEVEL | Receive FIFO Level. This indicates the number of data entries in the receive FIFO. | 0x0 |

Table 190: UART2_SRR_REG (0x50001188)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------|--|-------|
| 15:3 | - | - | Reserved | 0x0 |
| 2 | w | UART_XFR | XMIT FIFO Reset. This is a shadow register for the XMIT FIFO Reset bit (FCR[2]). This can be used to remove the burden on software having to store previously written FCR values (which are pretty static) just to reset the transmit FIFO. This resets the control portion of the transmit FIFO and treats the FIFO as empty. Note that this bit is 'self-clearing'. It is not necessary to clear this bit. | 0x0 |
| 1 | w | UART_RFR | RCVR FIFO Reset. This is a shadow register for the RCVR FIFO Reset bit (FCR[1]). This can be used to remove the burden on software having to store previously written FCR values (which are pretty static) just to reset the receive FIFO. This resets the control portion of the receive FIFO and treats the FIFO as empty. Note that this bit is 'self-clearing'. It is not necessary to clear this bit. | 0x0 |
| 0 | w | UART_UR | UART Reset. This asynchronously resets the UART Ctrl and synchronously removes the reset assertion. For a two clock implementation both pclk and sclk domains are reset. | 0x0 |

Table 191: UART2_SRTS_REG (0x5000118C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:1 | - | - | Reserved | 0x0 |

Table 191: UART2_SRTS_REG (0x5000118C)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|---------------------------------|---|-------|
| 0 | r/w | UART_SHADOW_R EQUEST_TO_SEND | Shadow Request to Send. This is a shadow register for the RTS bit (MCR[1]), this can be used to remove the burden of having to perform a read-modify-write on the MCR. This is used to directly control the Request to Send (rts_n) output. The Request To Send (rts_n) output is used to inform the modem or data set that the UART Ctrl is ready to exchange data. When Auto Flow Control is disabled (MCR[5] = 0), the rts_n signal is set low by programming MCR[1] (RTS) to a high. When Auto Flow Control is enabled (MCR[5] = 1) and FIFOs are enabled (FCR[0] = 1), the rts_n output is controlled in the same way, but is also gated with the receiver FIFO threshold trigger (rts_n is inactive high when above the threshold). Note that in Loopback mode (MCR[4] = 1), the rts_n output is held inactive-high while the value of this location is internally looped back to an input. | 0x0 |

Table 192: UART2_SBCR_REG (0x50001190)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-------------------------------|--|-------|
| 15:1 | - | - | Reserved | 0x0 |
| 0 | r/w | UART_SHADOW_B REAK_CONTROL | Shadow Break Control Bit. This is a shadow register for the Break bit (LCR[6]), this can be used to remove the burden of having to performing a read modify write on the LCR. This is used to cause a break condition to be transmitted to the receiving device. If set to one the serial output is forced to the spacing (logic 0) state. When not in Loopback Mode, as determined by MCR[4], the sout line is forced low until the Break bit is cleared. If SIR_MODE active (MCR[6] = 1) the sir_out_n line is continuously pulsed. When in Loopback Mode, the break condition is internally looped back to the receiver. | 0x0 |

Table 193: UART2_SDMAM_REG (0x50001194)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------------------|--|-------|
| 15:1 | - | - | Reserved | 0x0 |
| 0 | r/w | UART_SHADOW_D MA_MODE | Shadow DMA Mode. This is a shadow register for the DMA mode bit (FCR[3]). This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the DMA Mode bit gets updated. This determines the DMA signalling mode used for the dma_tx_req_n and dma_rx_req_n output signals. 0 = mode 0 1 = mode 1 | 0x0 |

Table 194: UART2_SFE_REG (0x50001198)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:1 | - | - | Reserved | 0x0 |

Table 194: UART2_SFE_REG (0x50001198)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-------------------------|---|-------|
| 0 | r/w | UART_SHADOW_FIFO_ENABLE | Shadow FIFO Enable. This is a shadow register for the FIFO enable bit (FCR[0]). This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the FIFO enable bit gets updated. This enables/disables the transmit (XMIT) and receive (RCVR) FIFOs. If this bit is set to zero (disabled) after being enabled then both the XMIT and RCVR controller portion of FIFOs are reset. | 0x0 |

Table 195: UART2_SRT_REG (0x5000119C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------------------|--|-------|
| 15:2 | - | - | Reserved | 0x0 |
| 1:0 | r/w | UART_SHADOW_RCVR_TRIGGER | Shadow RCVR Trigger. This is a shadow register for the RCVR trigger bits (FCR[7:6]). This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the RCVR trigger bit gets updated. This is used to select the trigger level in the receiver FIFO at which the Received Data Available Interrupt is generated. It also determines when the dma_rx_req_n signal is asserted when DMA Mode (FCR[3]) = 1. The following trigger levels are supported: 00 = 1 character in the FIFO 01 = FIFO \hat{A} ¼ full 10 = FIFO \hat{A} ½ full 11 = FIFO 2 less than full | 0x0 |

Table 196: UART2_STET_REG (0x500011A0)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------------------------|--|-------|
| 15:2 | - | - | Reserved | 0x0 |
| 1:0 | r/w | UART_SHADOW_TX_EMPTY_TRIGGER | Shadow TX Empty Trigger. This is a shadow register for the TX empty trigger bits (FCR[5:4]). This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the TX empty trigger bit gets updated. This is used to select the empty threshold level at which the THRE Interrupts are generated when the mode is active. The following trigger levels are supported: 00 = FIFO empty 01 = 2 characters in the FIFO 10 = FIFO \hat{A} ¼ full 11 = FIFO \hat{A} ½ full | 0x0 |

Table 197: UART2_HTX_REG (0x500011A4)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:1 | - | - | Reserved | 0x0 |

Table 197: UART2_HTX_REG (0x500011A4)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------------|---|-------|
| 0 | r/w | UART_HALT_TX | This register is use to halt transmissions for testing, so that the transmit FIFO can be filled by the master when FIFOs are implemented and enabled. 0 = Halt TX disabled 1 = Halt TX enabled Note, if FIFOs are implemented and not enabled, the setting of the halt TX register has no effect on operation. | 0x0 |

Table 198: UART2_CPR_REG (0x500011F4)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|------------------------------|-------|
| 15:0 | r | CPR | Component Parameter Register | 0x0 |

Table 199: UART2_UCV_REG (0x500011F8)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------------|------------|
| 15:0 | r | UCV | Component Version | 0x3330382A |

Table 200: UART2_CTR_REG (0x500011FC)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------------------|------------|
| 15:0 | r | CTR | Component Type Register | 0x44570110 |

Table 201: SPI_CTRL_REG (0x50001200)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------------|---|-------|
| 15 | r/w | SPI_EN_CTRL | 0 = SPI_EN pin disabled in slave mode. Pin SPI_EN is don't care. 1 = SPI_EN pin enabled in slave mode. | 0x0 |
| 14 | r/w | SPI_MINT | 0 = Disable SPI_INT_BIT to the Interrupt Controller 1 = Enable SPI_INT_BIT to the Interrupt Controller | 0x0 |
| 13 | r | SPI_INT_BIT | 0 = RX Register or FIFO is empty. 1 = SPI interrupt. Data has been transmitted and received-Must be reset by SW by writing to SPI_CLEAR_INT_REG. | 0x0 |
| 12 | r | SPI_DI | Returns the actual value of pin SPI_DIN (delayed with two internal SPI clock cycles) | 0x0 |
| 11 | r | SPI_TXH | 0 = TX-FIFO is not full, data can be written. 1 = TX-FIFO is full, data can not be written. | 0x0 |
| 10 | r/w | SPI_FORCE_DO | 0 = normal operation 1 = Force SPIDO output level to value of SPI_DO. | 0x0 |
| 9 | r/w | SPI_RST | 0 = normal operation 1 = Reset SPI. Same function as SPI_ON except that internal clock remain active. | 0x0 |
| 8:7 | r/w | SPI_WORD | 00 = 8 bits mode, only SPI_RX_TX_REG0 used 01 = 16 bit mode, only SPI_RX_TX_REG0 used 10 = 32 bits mode, SPI_RX_TX_REG0 & SPI_RX_TX_REG1 used 11 = 9 bits mode. Only valid in master mode. | 0x0 |

Table 201: SPI_CTRL_REG (0x50001200)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|---------|--|-------|
| 6 | r/w | SPI_SMN | Master/slave mode 0 = Master, 1 = Slave(SPI1 only) | 0x0 |
| 5 | r/w | SPI_DO | Pin SPI_DO output level when SPI is idle or when SPI_FORCE_DO=1 | 0x0 |
| 4:3 | r/w | SPI_CLK | Select SPI_CLK clock frequency in master mode:00 = (XTAL) / (CLK_PER_REG *8) 01 = (XTAL) / (CLK_PER_REG *4) 10 = (XTAL) / (CLK_PER_REG *2) 11 = (XTAL) / (CLK_PER_REG *14) | 0x0 |
| 2 | r/w | SPI_POL | Select SPI_CLK polarity. 0 = SPI_CLK is initially low. 1 = SPI_CLK is initially high. | 0x0 |
| 1 | r/w | SPI_PHA | Select SPI_CLK phase. See functional timing diagrams in SPI chapter | 0x0 |
| 0 | r/w | SPI_ON | 0 = SPI Module switched off (power saving). Everything is reset except SPI_CTRL_REG0 and SPI_CTRL_REG1. When this bit is cleared the SPI will remain active in master mode until the shift register and holding register are both empty. 1 = SPI Module switched on. Should only be set after all control bits have their desired values. So two writes are needed! | 0x0 |

Table 202: SPI_RX_TX_REG0 (0x50001202)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-----------|---|-------|
| 15:0 | r0/w | SPI_DATA0 | Write: SPI_TX_REG0 output register 0 (TX-FIFO) Read: SPI_RX_REG0 input register 0 (RX-FIFO) In 8 or 9 bits mode bits 15 to 8 are not used, they contain old data. | 0x0 |

Table 203: SPI_RX_TX_REG1 (0x50001204)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-----------|---|-------|
| 15:0 | r0/w | SPI_DATA1 | Write: SPI_TX_REG1 output register 1 (MSB's of TX-FIFO) Read: SPI_RX_REG1 input register 1 (MSB's of RX-FIFO) In 8 or 9 or 16 bits mode bits this register is not used. | 0x0 |

Table 204: SPI_CLEAR_INT_REG (0x50001206)

| Bit | Mode | Symbol | Description | Reset |
|------|------|---------------|---|-------|
| 15:0 | r0/w | SPI_CLEAR_INT | Writing any value to this register will clear the SPI_CTRL_REG[SPI_INT_BIT] Reading returns 0. | 0x0 |

Table 205: SPI_CTRL_REG1 (0x50001208)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------|---|-------|
| 15:5 | - | - | Reserved | 0x0 |
| 4 | r/w | SPI_9BIT_VAL | Determines the value of the first bit in 9 bits SPI mode. | 0x0 |

Table 205: SPI_CTRL_REG1 (0x50001208)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|---------------|---|-------|
| 3 | r | SPI_BUSY | 0 = The SPI is not busy with a transfer. This means that either no TX-data is available or that the transfers have been suspended due to a full RX-FIFO. The SPIx_CTRL_REG0[SPI_INT_BIT] can be used to distinguish between these situations. 1 = The SPI is busy with a transfer. | 0x0 |
| 2 | r/w | SPI_PRIORITY | 0 = The SPI has low priority, the DMA request signals are reset after the corresponding acknowledge. 1 = The SPI has high priority, DMA request signals remain active until the FIFOs are filled/emptied, so the DMA holds the AHB bus. | 0x0 |
| 1:0 | r/w | SPI_FIFO_MODE | 0: TX-FIFO and RX-FIFO used (Bidirectional mode). 1: RX-FIFO used (Read Only Mode) TX-FIFO single depth, no flow control 2: TX-FIFO used (Write Only Mode), RX-FIFO single depth, no flow control 3: No FIFOs used (backwards compatible mode) | 0x3 |

Table 206: I2C_CON_REG (0x50001300)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------------------|---|-------|
| 15:7 | - | - | Reserved | 0x0 |
| 6 | r/w | I2C_SLAVE_DISABLE | Slave enabled or disabled after reset is applied, which means software does not have to configure the slave. 0=slave is enabled 1=slave is disabled Software should ensure that if this bit is written with '0', then bit 0 should also be written with a '0'. | 0x1 |
| 5 | r/w | I2C_RESTART_EN | Determines whether RESTART conditions may be sent when acting as a master 0= disable 1=enable | 0x1 |
| 4 | r/w | I2C_10BITADDR_MASTER | Controls whether the controller starts its transfers in 7- or 10-bit addressing mode when acting as a master. 0= 7-bit addressing 1= 10-bit addressing | 0x1 |
| 3 | r/w | I2C_10BITADDR_SLAVE | When acting as a slave, this bit controls whether the controller responds to 7- or 10-bit addresses. 0= 7-bit addressing 1= 10-bit addressing | 0x1 |
| 2:1 | r/w | I2C_SPEED | These bits control at which speed the controller operates. 1= standard mode (100 kbit/s) 2= fast mode (400 kbit/s) | 0x2 |
| 0 | r/w | I2C_MASTER_MODE | This bit controls whether the controller master is enabled. 0= master disabled 1= master enabled Software should ensure that if this bit is written with '1' then bit 6 should also be written with a '1'. | 0x1 |

Table 207: I2C_TAR_REG (0x50001304)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|-------------|-------|
| 15:12 | - | - | Reserved | 0x0 |

Table 207: I2C_TAR_REG (0x50001304)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-------------|--|-------|
| 11 | r/w | SPECIAL | This bit indicates whether software performs a General Call or START BYTE command. 0: ignore bit 10 GC_OR_START and use IC_TAR normally 1: perform special I2C command as specified in GC_OR_START bit | 0x0 |
| 10 | r/w | GC_OR_START | If bit 11 (SPECIAL) is set to 1, then this bit indicates whether a General Call or START byte command is to be performed by the controller. 0: General Call Address - after issuing a General Call, only writes may be performed. Attempting to issue a read command results in setting bit 6 (TX_ABRT) of the IC_RAW_INTR_STAT register. The controller remains in General Call mode until the SPECIAL bit value (bit 11) is cleared. 1: START BYTE | 0x0 |
| 9:0 | r/w | IC_TAR | This is the target address for any master transaction. When transmitting a General Call, these bits are ignored. To generate a START BYTE, the CPU needs to write only once into these bits. Note: If the IC_TAR and IC_SAR are the same, loopback exists but the FIFOs are shared between master and slave, so full loopback is not feasible. Only one direction loopback mode is supported (simplex), not duplex. A master cannot transmit to itself; it can transmit to only a slave | 0x55 |

Table 208: I2C_SAR_REG (0x50001308)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|--|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:0 | r/w | IC_SAR | The IC_SAR holds the slave address when the I2C is operating as a slave. For 7-bit addressing, only IC_SAR[6:0] is used. This register can be written only when the I2C interface is disabled, which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect. | 0x55 |

Table 209: I2C_DATA_CMD_REG (0x50001310)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:9 | - | - | Reserved | 0x0 |

Table 209: I2C_DATA_CMD_REG (0x50001310)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|---|-------|
| 8 | r/w | CMD | <p>This bit controls whether a read or a write is performed. This bit does not control the direction when the I2C Ctrl acts as a slave. It controls only the direction when it acts as a master.</p> <p>1 = Read 0 = Write</p> <p>When a command is entered in the TX FIFO, this bit distinguishes the write and read commands. In slave-receiver mode, this bit is a "don't care" because writes to this register are not required. In slave-transmitter mode, a "0" indicates that CPU data is to be transmitted and as DAT or IC_DATA_CMD[7:0]. When programming this bit, you should remember the following: attempting to perform a read operation after a General Call command has been sent results in a TX_ABRT interrupt (bit 6 of the I2C_RAW_INTR_STAT_REG), unless bit 11 (SPECIAL) in the I2C_TAR register has been cleared.</p> <p>If a "1" is written to this bit after receiving a RD_REQ interrupt, then a TX_ABRT interrupt occurs.</p> <p>NOTE: It is possible that while attempting a master I2C read transfer on the controller, a RD_REQ interrupt may have occurred simultaneously due to a remote I2C master addressing the controller. In this type of scenario, it ignores the I2C_DATA_CMD write, generates a TX_ABRT interrupt, and waits to service the RD_REQ interrupt</p> | 0x0 |
| 7:0 | r/w | DAT | <p>This register contains the data to be transmitted or received on the I2C bus. If you are writing to this register and want to perform a read, bits 7:0 (DAT) are ignored by the controller. However, when you read this register, these bits return the value of data received on the controller's interface.</p> | 0x0 |

Table 210: I2C_SS_SCL_HCNT_REG (0x50001314)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------------|---|-------|
| 15:0 | r/w | IC_SS_SCL_HCNT | <p>This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for standard speed. This register can be written only when the I2C interface is disabled which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect.</p> <p>The minimum valid value is 6; hardware prevents values less than this being written, and if attempted results in 6 being set.</p> <p>NOTE: This register must not be programmed to a value higher than 65525, because the controller uses a 16-bit counter to flag an I2C bus idle condition when this counter reaches a value of IC_SS_SCL_HCNT + 10.</p> | 0x48 |

Table 211: I2C_SS_SCL_LCNT_REG (0x50001318)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------------|---|-------|
| 15:0 | r/w | IC_SS_SCL_LCNT | This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low period count for standard speed. This register can be written only when the I2C interface is disabled which corresponds to the I2C_ENABLE register being set to 0. Writes at other times have no effect. The minimum valid value is 8; hardware prevents values less than this being written, and if attempted, results in 8 being set. | 0x4F |

Table 212: I2C_FS_SCL_HCNT_REG (0x5000131C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------------|---|-------|
| 15:0 | r/w | IC_FS_SCL_HCNT | This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for fast speed. It is used in high-speed mode to send the Master Code and START BYTE or General CALL. This register can be written only when the I2C interface is disabled, which corresponds to the I2C_ENABLE register being set to 0. Writes at other times have no effect. The minimum valid value is 6; hardware prevents values less than this being written, and if attempted results in 6 being set. | 0x8 |

Table 213: I2C_FS_SCL_LCNT_REG (0x50001320)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------------|---|-------|
| 15:0 | r/w | IC_FS_SCL_LCNT | This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low-period count for fast speed. It is used in high-speed mode to send the Master Code and START BYTE or General CALL. This register can be written only when the I2C interface is disabled, which corresponds to the I2C_ENABLE register being set to 0. Writes at other times have no effect. The minimum valid value is 8; hardware prevents values less than this being written, and if attempted results in 8 being set. For designs with APB_DATA_WIDTH = 8 the order of programming is important to ensure the correct operation of the controller. The lower byte must be programmed first. Then the upper byte is programmed. | 0x17 |

Table 214: I2C_INTR_STAT_REG (0x5000132C)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|------------|--|-------|
| 15:12 | - | - | Reserved | 0x0 |
| 11 | r | R_GEN_CALL | Set only when a General Call address is received and it is acknowledged. It stays set until it is cleared either by disabling controller or when the CPU reads bit 0 of the I2C_CLR_GEN_CALL register. The controller stores the received data in the Rx buffer. | 0x0 |

Table 214: I2C_INTR_STAT_REG (0x5000132C)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-------------|---|-------|
| 10 | r | R_START_DET | Indicates whether a START or RESTART condition has occurred on the I2C interface regardless of whether controller is operating in slave or master mode. | 0x0 |
| 9 | r | R_STOP_DET | Indicates whether a STOP condition has occurred on the I2C interface regardless of whether controller is operating in slave or master mode. | 0x0 |
| 8 | r | R_ACTIVITY | This bit captures I2C Ctrl activity and stays set until it is cleared. There are four ways to clear it: => Disabling the I2C Ctrl => Reading the IC_CLR_ACTIVITY register => Reading the IC_CLR_INTR register => System reset Once this bit is set, it stays set unless one of the four methods is used to clear it. Even if the controller module is idle, this bit remains set until cleared, indicating that there was activity on the bus. | 0x0 |
| 7 | r | R_RX_DONE | When the controller is acting as a slave-transmitter, this bit is set to 1 if the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission, indicating that the transmission is done. | 0x0 |
| 6 | r | R_TX_ABORT | This bit indicates if the controller, as an I2C transmitter, is unable to complete the intended actions on the contents of the transmit FIFO. This situation can occur both as an I2C master or an I2C slave, and is referred to as a "transmit abort". When this bit is set to 1, the I2C_TX_ABORT_SOURCE register indicates the reason why the transmit abort takes places. NOTE: The controller flushes/resets/empties the TX FIFO whenever this bit is set. The TX FIFO remains in this flushed state until the register I2C_CLR_TX_ABORT is read. Once this read is performed, the TX FIFO is then ready to accept more data bytes from the APB interface. | 0x0 |
| 5 | r | R_RD_REQ | This bit is set to 1 when the controller is acting as a slave and another I2C master is attempting to read data from the controller. The controller holds the I2C bus in a wait state (SCL=0) until this interrupt is serviced, which means that the slave has been addressed by a remote master that is asking for data to be transferred. The processor must respond to this interrupt and then write the requested data to the I2C_DATA_CMD register. This bit is set to 0 just after the processor reads the I2C_CLR_RD_REQ register | 0x0 |
| 4 | r | R_TX_EMPTY | This bit is set to 1 when the transmit buffer is at or below the threshold value set in the I2C_TX_TL register. It is automatically cleared by hardware when the buffer level goes above the threshold. When the IC_ENABLE bit 0 is 0, the TX FIFO is flushed and held in reset. There the TX FIFO looks like it has no data within it, so this bit is set to 1, provided there is activity in the master or slave state machines. When there is no longer activity, then with ic_en=0, this bit is set to 0. | 0x0 |
| 3 | r | R_TX_OVER | Set during transmit if the transmit buffer is filled to 32 and the processor attempts to issue another I2C command by writing to the IC_DATA_CMD register. When the module is disabled, this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared | 0x0 |

Table 214: I2C_INTR_STAT_REG (0x5000132C)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|------------|--|-------|
| 2 | r | R_RX_FULL | Set when the receive buffer reaches or goes above the RX_TL threshold in the I2C_RX_TL register. It is automatically cleared by hardware when buffer level goes below the threshold. If the module is disabled (I2C_ENABLE[0]=0), the RX FIFO is flushed and held in reset; therefore the RX FIFO is not full. So this bit is cleared once the I2C_ENABLE bit 0 is programmed with a 0, regardless of the activity that continues. | 0x0 |
| 1 | r | R_RX_OVER | Set if the receive buffer is completely filled to 32 and an additional byte is received from an external I2C device. The controller acknowledges this, but any data bytes received after the FIFO is full are lost. If the module is disabled (I2C_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared. | 0x0 |
| 0 | r | R_RX_UNDER | Set if the processor attempts to read the receive buffer when it is empty by reading from the IC_DATA_CMD register. If the module is disabled (I2C_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared. | 0x0 |

Table 215: I2C_INTR_MASK_REG (0x50001330)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-------------|--|-------|
| 15:12 | - | - | Reserved | 0x0 |
| 11 | r/w | M_GEN_CALL | These bits mask their corresponding interrupt status bits in the I2C_INTR_STAT register. | 0x1 |
| 10 | r/w | M_START_DET | These bits mask their corresponding interrupt status bits in the I2C_INTR_STAT register. | 0x0 |
| 9 | r/w | M_STOP_DET | These bits mask their corresponding interrupt status bits in the I2C_INTR_STAT register. | 0x0 |
| 8 | r/w | M_ACTIVITY | These bits mask their corresponding interrupt status bits in the I2C_INTR_STAT register. | 0x0 |
| 7 | r/w | M_RX_DONE | These bits mask their corresponding interrupt status bits in the I2C_INTR_STAT register. | 0x1 |
| 6 | r/w | M_TX_ABRT | These bits mask their corresponding interrupt status bits in the I2C_INTR_STAT register. | 0x1 |
| 5 | r/w | M_RD_REQ | These bits mask their corresponding interrupt status bits in the I2C_INTR_STAT register. | 0x1 |
| 4 | r/w | M_TX_EMPTY | These bits mask their corresponding interrupt status bits in the I2C_INTR_STAT register. | 0x1 |
| 3 | r/w | M_TX_OVER | These bits mask their corresponding interrupt status bits in the I2C_INTR_STAT register. | 0x1 |
| 2 | r/w | M_RX_FULL | These bits mask their corresponding interrupt status bits in the I2C_INTR_STAT register. | 0x1 |
| 1 | r/w | M_RX_OVER | These bits mask their corresponding interrupt status bits in the I2C_INTR_STAT register. | 0x1 |
| 0 | r/w | M_RX_UNDER | These bits mask their corresponding interrupt status bits in the I2C_INTR_STAT register. | 0x1 |

Table 216: I2C_RAW_INTR_STAT_REG (0x50001334)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-----------|---|-------|
| 15:12 | - | - | Reserved | 0x0 |
| 11 | r | GEN_CALL | Set only when a General Call address is received and it is acknowledged. It stays set until it is cleared either by disabling controller or when the CPU reads bit 0 of the I2C_CLR_GEN_CALL register. I2C Ctrl stores the received data in the Rx buffer. | 0x0 |
| 10 | r | START_DET | Indicates whether a START or RESTART condition has occurred on the I2C interface regardless of whether controller is operating in slave or master mode. | 0x0 |
| 9 | r | STOP_DET | Indicates whether a STOP condition has occurred on the I2C interface regardless of whether controller is operating in slave or master mode. | 0x0 |
| 8 | r | ACTIVITY | This bit captures I2C Ctrl activity and stays set until it is cleared. There are four ways to clear it: => Disabling the I2C Ctrl => Reading the IC_CLR_ACTIVITY register => Reading the IC_CLR_INTR register => System reset Once this bit is set, it stays set unless one of the four methods is used to clear it. Even if the controller module is idle, this bit remains set until cleared, indicating that there was activity on the bus. | 0x0 |
| 7 | r | RX_DONE | When the controller is acting as a slave-transmitter, this bit is set to 1 if the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission, indicating that the transmission is done. | 0x0 |
| 6 | r | TX_ABRT | This bit indicates if the controller, as an I2C transmitter, is unable to complete the intended actions on the contents of the transmit FIFO. This situation can occur both as an I2C master or an I2C slave, and is referred to as a "transmit abort". When this bit is set to 1, the I2C_TX_ABRT_SOURCE register indicates the reason why the transmit abort takes places. NOTE: The controller flushes/resets/empties the TX FIFO whenever this bit is set. The TX FIFO remains in this flushed state until the register I2C_CLR_TX_ABRT is read. Once this read is performed, the TX FIFO is then ready to accept more data bytes from the APB interface. | 0x0 |
| 5 | r | RD_REQ | This bit is set to 1 when I2C Ctrl is acting as a slave and another I2C master is attempting to read data from the controller. The controller holds the I2C bus in a wait state (SCL=0) until this interrupt is serviced, which means that the slave has been addressed by a remote master that is asking for data to be transferred. The processor must respond to this interrupt and then write the requested data to the I2C_DATA_CMD register. This bit is set to 0 just after the processor reads the I2C_CLR_RD_REQ register | 0x0 |
| 4 | r | TX_EMPTY | This bit is set to 1 when the transmit buffer is at or below the threshold value set in the I2C_TX_TL register. It is automatically cleared by hardware when the buffer level goes above the threshold. When the IC_ENABLE bit 0 is 0, the TX FIFO is flushed and held in reset. There the TX FIFO looks like it has no data within it, so this bit is set to 1, provided there is activity in the master or slave state machines. When there is no longer activity, then with ic_en=0, this bit is set to 0. | 0x0 |

Table 216: I2C_RAW_INTR_STAT_REG (0x50001334)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|----------|--|-------|
| 3 | r | TX_OVER | Set during transmit if the transmit buffer is filled to 32 and the processor attempts to issue another I2C command by writing to the IC_DATA_CMD register. When the module is disabled, this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared | 0x0 |
| 2 | r | RX_FULL | Set when the receive buffer reaches or goes above the RX_TL threshold in the I2C_RX_TL register. It is automatically cleared by hardware when buffer level goes below the threshold. If the module is disabled (I2C_ENABLE[0]=0), the RX FIFO is flushed and held in reset; therefore the RX FIFO is not full. So this bit is cleared once the I2C_ENABLE bit 0 is programmed with a 0, regardless of the activity that continues. | 0x0 |
| 1 | r | RX_OVER | Set if the receive buffer is completely filled to 32 and an additional byte is received from an external I2C device. The controller acknowledges this, but any data bytes received after the FIFO is full are lost. If the module is disabled (I2C_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared. | 0x0 |
| 0 | r | RX_UNDER | Set if the processor attempts to read the receive buffer when it is empty by reading from the IC_DATA_CMD register. If the module is disabled (I2C_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared. | 0x0 |

Table 217: I2C_RX_TL_REG (0x50001338)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|---|-------|
| 15:5 | - | - | Reserved | 0x0 |
| 4:0 | r/w | RX_TL | Receive FIFO Threshold Level Controls the level of entries (or above) that triggers the RX_FULL interrupt (bit 2 in I2C_RAW_INTR_STAT register). The valid range is 0-31, with the additional restriction that hardware does not allow this value to be set to a value larger than the depth of the buffer. If an attempt is made to do that, the actual value set will be the maximum depth of the buffer. A value of 0 sets the threshold for 1 entry, and a value of 31 sets the threshold for 32 entries. | 0x0 |

Table 218: I2C_TX_TL_REG (0x5000133C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|--|-------|
| 15:5 | - | - | Reserved | 0x0 |
| 4:0 | r/w | RX_TL | Transmit FIFO Threshold Level Controls the level of entries (or below) that trigger the TX_EMPTY interrupt (bit 4 in I2C_RAW_INTR_STAT register). The valid range is 0-31, with the additional restriction that it may not be set to value larger than the depth of the buffer. If an attempt is made to do that, the actual value set will be the maximum depth of the buffer. A value of 0 sets the threshold for 0 entries, and a value of 31 sets the threshold for 32 entries.. | 0x0 |

Table 219: I2C_CLR_INTR_REG (0x50001340)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------|--|-------|
| 15:1 | - | - | Reserved | 0x0 |
| 0 | r | CLR_INTR | Read this register to clear the combined interrupt, all individual interrupts, and the I2C_TX_ABRT_SOURCE register. This bit does not clear hardware clearable interrupts but software clearable interrupts. Refer to Bit 9 of the I2C_TX_ABRT_SOURCE register for an exception to clearing I2C_TX_ABRT_SOURCE | 0x0 |

Table 220: I2C_CLR_RX_UNDER_REG (0x50001344)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------|---|-------|
| 15:1 | - | - | Reserved | 0x0 |
| 0 | r | CLR_RX_UNDER | Read this register to clear the RX_UNDER interrupt (bit 0) of the I2C_RAW_INTR_STAT register. | 0x0 |

Table 221: I2C_CLR_RX_OVER_REG (0x50001348)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-------------|--|-------|
| 15:1 | - | - | Reserved | 0x0 |
| 0 | r | CLR_RX_OVER | Read this register to clear the RX_OVER interrupt (bit 1) of the I2C_RAW_INTR_STAT register. | 0x0 |

Table 222: I2C_CLR_TX_OVER_REG (0x5000134C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-------------|--|-------|
| 15:1 | - | - | Reserved | 0x0 |
| 0 | r | CLR_TX_OVER | Read this register to clear the TX_OVER interrupt (bit 3) of the I2C_RAW_INTR_STAT register. | 0x0 |

Table 223: I2C_CLR_RD_REQ_REG (0x50001350)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------|---|-------|
| 15:1 | - | - | Reserved | 0x0 |
| 0 | r | CLR_RD_REQ | Read this register to clear the RD_REQ interrupt (bit 5) of the I2C_RAW_INTR_STAT register. | 0x0 |

Table 224: I2C_CLR_TX_ABRT_REG (0x50001354)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-------------|--|-------|
| 15:1 | - | - | Reserved | 0x0 |
| 0 | r | CLR_TX_ABRT | Read this register to clear the TX_ABRT interrupt (bit 6) of the IC_RAW_INTR_STAT register, and the I2C_TX_ABRT_SOURCE register. This also releases the TX FIFO from the flushed/reset state, allowing more writes to the TX FIFO. Refer to Bit 9 of the I2C_TX_ABRT_SOURCE register for an exception to clearing IC_TX_ABRT_SOURCE. | 0x0 |

Table 225: I2C_CLR_RX_DONE_REG (0x50001358)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-------------|--|-------|
| 15:1 | - | - | Reserved | 0x0 |
| 0 | r | CLR_RX_DONE | Read this register to clear the RX_DONE interrupt (bit 7) of the I2C_RAW_INTR_STAT register. | 0x0 |

Table 226: I2C_CLR_ACTIVITY_REG (0x5000135C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------|--|-------|
| 15:1 | - | - | Reserved | 0x0 |
| 0 | r | CLR_ACTIVITY | Reading this register clears the ACTIVITY interrupt if the I2C is not active anymore. If the I2C module is still active on the bus, the ACTIVITY interrupt bit continues to be set. It is automatically cleared by hardware if the module is disabled and if there is no further activity on the bus. The value read from this register to get status of the ACTIVITY interrupt (bit 8) of the IC_RAW_INTR_STAT register | 0x0 |

Table 227: I2C_CLR_STOP_DET_REG (0x50001360)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------|---|-------|
| 15:1 | - | - | Reserved | 0x0 |
| 0 | r | CLR_ACTIVITY | Reading this register clears the ACTIVITY interrupt if the I2C is not active anymore. If the I2C module is still active on the bus, the ACTIVITY interrupt bit continues to be set. It is automatically cleared by hardware if the module is disabled and if there is no further activity on the bus. The value read from this register to get status of the ACTIVITY interrupt (bit 8) of the IC_RAW_INTR_STAT register. | 0x0 |

Table 228: I2C_CLR_START_DET_REG (0x50001364)

| Bit | Mode | Symbol | Description | Reset |
|------|------|---------------|--|-------|
| 15:1 | - | - | Reserved | 0x0 |
| 0 | r | CLR_START_DET | Read this register to clear the START_DET interrupt (bit 10) of the IC_RAW_INTR_STAT register. | 0x0 |

Table 229: I2C_CLR_GEN_CALL_REG (0x50001368)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------|--|-------|
| 15:1 | - | - | Reserved | 0x0 |
| 0 | r | CLR_GEN_CALL | Read this register to clear the GEN_CALL interrupt (bit 11) of I2C_RAW_INTR_STAT register. | 0x0 |

Table 230: I2C_ENABLE_REG (0x5000136C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:1 | - | - | Reserved | 0x0 |

Table 230: I2C_ENABLE_REG (0x5000136C)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-------------|---|-------|
| 0 | r/w | CTRL_ENABLE | <p>Controls whether the controller is enabled.</p> <p>0: Disables the controller (TX and RX FIFOs are held in an erased state)</p> <p>1: Enables the controller</p> <p>Software can disable the controller while it is active. However, it is important that care be taken to ensure that the controller is disabled properly. When the controller is disabled, the following occurs:</p> <ul style="list-style-type: none"> * The TX FIFO and RX FIFO get flushed. * Status bits in the IC_INTR_STAT register are still active until the controller goes into IDLE state. <p>If the module is transmitting, it stops as well as deletes the contents of the transmit buffer after the current transfer is complete. If the module is receiving, the controller stops the current transfer at the end of the current byte and does not acknowledge the transfer.</p> <p>There is a two ic_clk delay when enabling or disabling the controller</p> | 0x0 |

Table 231: I2C_STATUS_REG (0x50001370)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------|--|-------|
| 15:7 | - | - | Reserved | 0x0 |
| 6 | r | SLV_ACTIVITY | <p>Slave FSM Activity Status. When the Slave Finite State Machine (FSM) is not in the IDLE state, this bit is set.</p> <p>0: Slave FSM is in IDLE state so the Slave part of the controller is not Active</p> <p>1: Slave FSM is not in IDLE state so the Slave part of the controller is Active</p> | 0x0 |
| 5 | r | MST_ACTIVITY | <p>Master FSM Activity Status. When the Master Finite State Machine (FSM) is not in the IDLE state, this bit is set.</p> <p>0: Master FSM is in IDLE state so the Master part of the controller is not Active</p> <p>1: Master FSM is not in IDLE state so the Master part of the controller is Active</p> | 0x0 |
| 4 | r | RFF | <p>Receive FIFO Completely Full. When the receive FIFO is completely full, this bit is set. When the receive FIFO contains one or more empty location, this bit is cleared.</p> <p>0: Receive FIFO is not full</p> <p>1: Receive FIFO is full</p> | 0x0 |
| 3 | r | RFNE | <p>Receive FIFO Not Empty. This bit is set when the receive FIFO contains one or more entries; it is cleared when the receive FIFO is empty.</p> <p>0: Receive FIFO is empty</p> <p>1: Receive FIFO is not empty</p> | 0x0 |
| 2 | r | TFE | <p>Transmit FIFO Completely Empty. When the transmit FIFO is completely empty, this bit is set. When it contains one or more valid entries, this bit is cleared. This bit field does not request an interrupt.</p> <p>0: Transmit FIFO is not empty</p> <p>1: Transmit FIFO is empty</p> | 0x1 |
| 1 | r | TFNF | <p>Transmit FIFO Not Full. Set when the transmit FIFO contains one or more empty locations, and is cleared when the FIFO is full.</p> <p>0: Transmit FIFO is full</p> <p>1: Transmit FIFO is not full</p> | 0x1 |

Table 231: I2C_STATUS_REG (0x50001370)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------------|----------------------|-------|
| 0 | r | I2C_ACTIVITY | I2C Activity Status. | 0x0 |

Table 232: I2C_TXFLR_REG (0x50001374)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|---|-------|
| 15:6 | - | - | Reserved | 0x0 |
| 5:0 | r | TXFLR | Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO. Size is constrained by the TXFLR value | 0x0 |

Table 233: I2C_RXFLR_REG (0x50001378)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|---|-------|
| 15:6 | - | - | Reserved | 0x0 |
| 5:0 | r | RXFLR | Receive FIFO Level. Contains the number of valid data entries in the receive FIFO. Size is constrained by the RXFLR value | 0x0 |

Table 234: I2C_SDA_HOLD_REG (0x5000137C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-------------|---------------|-------|
| 15:0 | r/w | IC_SDA_HOLD | SDA Hold time | 0x1 |

Table 235: I2C_TX_ABRT_SOURCE_REG (0x50001380)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|----------------------|---|-------|
| 15 | r | ABRT_SLVRD_INTX | 1: When the processor side responds to a slave mode request for data to be transmitted to a remote master and user writes a 1 in CMD (bit 8) of 2IC_DATA_CMD register | 0x0 |
| 14 | r | ABRT_SLV_ARBLOST | 1: Slave lost the bus while transmitting data to a remote master. I2C_TX_ABRT_SOURCE[12] is set at the same time. Note: Even though the slave never "owns" the bus, something could go wrong on the bus. This is a fail safe check. For instance, during a data transmission at the low-to-high transition of SCL, if what is on the data bus is not what is supposed to be transmitted, then the controller no longer own the bus. | 0x0 |
| 13 | r | ABRT_SLVFLUSH_TXFIFO | 1: Slave has received a read command and some data exists in the TX FIFO so the slave issues a TX_ABRT interrupt to flush old data in TX FIFO. | 0x0 |
| 12 | r | ARB_LOST | 1: Master has lost arbitration, or if I2C_TX_ABRT_SOURCE[14] is also set, then the slave transmitter has lost arbitration. Note: I2C can be both master and slave at the same time. | 0x0 |
| 11 | r | ABRT_MASTER_DIS | 1: User tries to initiate a Master operation with the Master mode disabled. | 0x0 |
| 10 | r | ABRT_10B_RD_NO RSTRT | 1: The restart is disabled (IC_RESTART_EN bit (I2C_CON[5]) = 0) and the master sends a read command in 10-bit addressing mode. | 0x0 |

Table 235: I2C_TX_ABRT_SOURCE_REG (0x50001380)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|---------------------|---|-------|
| 9 | r | ABRT_SBYTE_NORSTRT | To clear Bit 9, the source of the ABRT_SBYTE_NORSTRT must be fixed first; restart must be enabled (I2C_CON[5]=1), the SPECIAL bit must be cleared (I2C_TAR[11]), or the GC_OR_START bit must be cleared (I2C_TAR[10]). Once the source of the ABRT_SBYTE_NORSTRT is fixed, then this bit can be cleared in the same manner as other bits in this register. If the source of the ABRT_SBYTE_NORSTRT is not fixed before attempting to clear this bit, bit 9 clears for one cycle and then gets re-asserted. 1: The restart is disabled (IC_RESTART_EN bit (I2C_CON[5]) = 0) and the user is trying to send a START Byte. | 0x0 |
| 8 | r | ABRT_HS_NORSTR T | 1: The restart is disabled (IC_RESTART_EN bit (I2C_CON[5]) = 0) and the user is trying to use the master to transfer data in High Speed mode | 0x0 |
| 7 | r | ABRT_SBYTE_ACK DET | 1: Master has sent a START Byte and the START Byte was acknowledged (wrong behavior). | 0x0 |
| 6 | r | ABRT_HS_ACKDET | 1: Master is in High Speed mode and the High Speed Master code was acknowledged (wrong behavior). | 0x0 |
| 5 | r | ABRT_GCALL_REA D | 1: the controller in master mode sent a General Call but the user programmed the byte following the General Call to be a read from the bus (IC_DATA_CMD[9] is set to 1). | 0x0 |
| 4 | r | ABRT_GCALL_NOA CK | 1: the controller in master mode sent a General Call and no slave on the bus acknowledged the General Call. | 0x0 |
| 3 | r | ABRT_TXDATA_NO ACK | 1: This is a master-mode only bit. Master has received an acknowledgement for the address, but when it sent data byte(s) following the address, it did not receive an acknowledgement from the remote slave(s). | 0x0 |
| 2 | r | ABRT_10ADDR2_N OACK | 1: Master is in 10-bit address mode and the second address byte of the 10-bit address was not acknowledged by any slave. | 0x0 |
| 1 | r | ABRT_10ADDR1_N OACK | 1: Master is in 10-bit address mode and the first 10-bit address byte was not acknowledged by any slave. | 0x0 |
| 0 | r | ABRT_7B_ADDR_N OACK | 1: Master is in 7-bit addressing mode and the address sent was not acknowledged by any slave. | 0x0 |

Table 236: I2C_SDA_SETUP_REG (0x50001394)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-----------|--|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | SDA_SETUP | SDA Setup. This register controls the amount of time delay (number of I2C clock periods) between the rising edge of SCL and SDA changing by holding SCL low when I2C block services a read request while operating as a slave-transmitter. The relevant I2C requirement is tSU:DAT (note 4) as detailed in the I2C Bus Specification. This register must be programmed with a value equal to or greater than 2. It is recommended that if the required delay is 1000ns, then for an I2C frequency of 10 MHz, IC_SDA_SETUP should be programmed to a value of 11. Writes to this register succeed only when IC_ENABLE[0] = 0. | 0x64 |

Table 237: I2C_ACK_GENERAL_CALL_REG (0x50001398)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------|---|-------|
| 15:1 | - | - | Reserved | 0x0 |
| 0 | r/w | ACK_GEN_CALL | ACK General Call. When set to 1, I2C Ctrl responds with a ACK (by asserting ic_data_oe) when it receives a General Call. When set to 0, the controller does not generate General Call interrupts. | 0x0 |

Table 238: I2C_ENABLE_STATUS_REG (0x5000139C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-------------------------|--|-------|
| 15:3 | - | - | Reserved | 0x0 |
| 2 | r | SLV_RX_DATA_LOST | Slave Received Data Lost. This bit indicates if a Slave-Receiver operation has been aborted with at least one data byte received from an I2C transfer due to the setting of IC_ENABLE from 1 to 0. When read as 1, the controller is deemed to have been actively engaged in an aborted I2C transfer (with matching address) and the data phase of the I2C transfer has been entered, even though a data byte has been responded with a NACK. NOTE: If the remote I2C master terminates the transfer with a STOP condition before the controller has a chance to NACK a transfer, and IC_ENABLE has been set to 0, then this bit is also set to 1. When read as 0, the controller is deemed to have been disabled without being actively involved in the data phase of a Slave-Receiver transfer. NOTE: The CPU can safely read this bit when IC_EN (bit 0) is read as 0. | 0x0 |
| 1 | r | SLV_DISABLED_WHILE_BUSY | Slave Disabled While Busy (Transmit, Receive). This bit indicates if a potential or active Slave operation has been aborted due to the setting of the IC_ENABLE register from 1 to 0. This bit is set when the CPU writes a 0 to the IC_ENABLE register while: (a) I2C Ctrl is receiving the address byte of the Slave-Transmitter operation from a remote master; OR, (b) address and data bytes of the Slave-Receiver operation from a remote master. When read as 1, the controller is deemed to have forced a NACK during any part of an I2C transfer, irrespective of whether the I2C address matches the slave address set in I2C Ctrl (IC_SAR register) OR if the transfer is completed before IC_ENABLE is set to 0 but has not taken effect. NOTE: If the remote I2C master terminates the transfer with a STOP condition before the the controller has a chance to NACK a transfer, and IC_ENABLE has been set to 0, then this bit will also be set to 1. When read as 0, the controller is deemed to have been disabled when there is master activity, or when the I2C bus is idle. NOTE: The CPU can safely read this bit when IC_EN (bit 0) is read as 0. | 0x0 |

Table 238: I2C_ENABLE_STATUS_REG (0x5000139C)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|--|-------|
| 0 | r | IC_EN | ic_en Status. This bit always reflects the value driven on the output port ic_en. When read as 1, the controller is deemed to be in an enabled state. When read as 0, the controller is deemed completely inactive. NOTE: The CPU can safely read this bit anytime. When this bit is read as 0, the CPU can safely read SLV_RX_DATA_LOST (bit 2) and SLV_DISABLED_WHILE_BUSY (bit 1). | 0x0 |

Table 239: I2C_IC_FS_SPKLEN_REG (0x500013A0)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------|---|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | IC_FS_SPKLEN | This register must be set before any I2C bus transaction can take place to ensure stable operation. This register sets the duration, measured in ic_clk cycles, of the longest spike in the SCL or SDA lines that will be filtered out by the spike suppression logic. This register can be written only when the I2C interface is disabled which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect. The minimum valid value is 2; hardware prevents values less than this being written, and if attempted results in 2 being set. | 0x1 |

Table 240: GPIO_IRQ0_IN_SEL_REG (0x50001400)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:6 | - | - | Reserved | 0x0 |

Table 240: GPIO_IRQ0_IN_SEL_REG (0x50001400)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|---------------|--|-------|
| 5:0 | r/w | KBRD_IRQ0_SEL | input selection that can generate a GPIO interrupt 0: no input selected 1: P0[0] is selected 2: P0[1] is selected 3: P0[2] is selected 4: P0[3] is selected 5: P0[4] is selected 6: P0[5] is selected 7: P0[6] is selected 8: P0[7] is selected 9: P1[0] is selected 10: P1[1] is selected 11: P1[2] is selected 12: P1[3] is selected 13: P1[4] is selected 14: P1[5] is selected 15: P2[0] is selected 16: P2[1] is selected 17: P2[2] is selected 18: P2[3] is selected 19: P2[4] is selected 20: P2[5] is selected 21: P2[6] is selected 22: P2[7] is selected 23: P2[8] is selected 24: P2[9] is selected 25: P3[0] is selected 26: P3[1] is selected 27: P3[2] is selected 28: P3[3] is selected 29: P3[4] is selected 30: P3[5] is selected 31: P3[6] is selected 32: P3[7] is selected all others: no input selected | 0x0 |

Table 241: GPIO_IRQ1_IN_SEL_REG (0x50001402)

| Bit | Mode | Symbol | Description | Reset |
|------|------|---------------|-------------------|-------|
| 15:6 | - | - | Reserved | 0x0 |
| 5:0 | r/w | KBRD_IRQ1_SEL | see KBRD_IRQ0_SEL | 0x0 |

Table 242: GPIO_IRQ2_IN_SEL_REG (0x50001404)

| Bit | Mode | Symbol | Description | Reset |
|------|------|---------------|-------------------|-------|
| 15:6 | - | - | Reserved | 0x0 |
| 5:0 | r/w | KBRD_IRQ2_SEL | see KBRD_IRQ0_SEL | 0x0 |

Table 243: GPIO_IRQ3_IN_SEL_REG (0x50001406)

| Bit | Mode | Symbol | Description | Reset |
|------|------|---------------|-------------------|-------|
| 15:6 | - | - | Reserved | 0x0 |
| 5:0 | r/w | KBRD_IRQ3_SEL | see KBRD_IRQ0_SEL | 0x0 |

Table 244: GPIO_IRQ4_IN_SEL_REG (0x50001408)

| Bit | Mode | Symbol | Description | Reset |
|------|------|---------------|-------------------|-------|
| 15:6 | - | - | Reserved | 0x0 |
| 5:0 | r/w | KBRD_IRQ4_SEL | see KBRD_IRQ0_SEL | 0x0 |

Table 245: GPIO_DEBOUNCE_REG (0x5000140C)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-----------------|---|-------|
| 15:14 | - | - | Reserved | 0x0 |
| 13 | r/w | DEB_ENABLE_KBRD | enables the debounce counter for the KBRD interface | 0x0 |
| 12 | r/w | DEB_ENABLE4 | enables the debounce counter for GPIO IRQ4 | 0x0 |
| 11 | r/w | DEB_ENABLE3 | enables the debounce counter for GPIO IRQ3 | 0x0 |
| 10 | r/w | DEB_ENABLE2 | enables the debounce counter for GPIO IRQ2 | 0x0 |
| 9 | r/w | DEB_ENABLE1 | enables the debounce counter for GPIO IRQ1 | 0x0 |
| 8 | r/w | DEB_ENABLE0 | enables the debounce counter for GPIO IRQ0 | 0x0 |
| 7:6 | - | - | Reserved | 0x0 |
| 5:0 | r/w | DEB_VALUE | Keyboard debounce time if enabled. Generate KEYB_INT after specified time. Debounce time: $N * 1 \text{ ms}$. $N = 0..63$ | 0x0 |

Table 246: GPIO_RESET_IRQ_REG (0x5000140E)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-----------------|--|-------|
| 15:6 | - | - | Reserved | 0x0 |
| 5 | r0/w | RESET_KBRD_IRQ | writing a 1 to this bit will reset the KBRD IRQ. Reading returns 0. | 0x0 |
| 4 | r0/w | RESET_GPIO4_IRQ | writing a 1 to this bit will reset the GPIO4 IRQ. Reading returns 0. | 0x0 |
| 3 | r0/w | RESET_GPIO3_IRQ | writing a 1 to this bit will reset the GPIO3 IRQ. Reading returns 0. | 0x0 |
| 2 | r0/w | RESET_GPIO2_IRQ | writing a 1 to this bit will reset the GPIO2 IRQ. Reading returns 0. | 0x0 |
| 1 | r0/w | RESET_GPIO1_IRQ | writing a 1 to this bit will reset the GPIO1 IRQ. Reading returns 0. | 0x0 |
| 0 | r0/w | RESET_GPIO0_IRQ | writing a 1 to this bit will reset the GPIO0 IRQ. Reading returns 0. | 0x0 |

Table 247: GPIO_INT_LEVEL_CTRL_REG (0x50001410)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------------|---|-------|
| 15:14 | - | - | Reserved | 0x0 |
| 12 | r/w | EDGE_LEVELN4 | see EDGE_LEVELN0, but for GPIO IRQ4 | 0x0 |
| 11 | r/w | EDGE_LEVELN3 | see EDGE_LEVELN0, but for GPIO IRQ3 | 0x0 |
| 10 | r/w | EDGE_LEVELN2 | see EDGE_LEVELN0, but for GPIO IRQ2 | 0x0 |
| 9 | r/w | EDGE_LEVELN1 | see EDGE_LEVELN0, but for GPIO IRQ1 | 0x0 |
| 8 | r/w | EDGE_LEVELN0 | 0: do not wait for key release after interrupt was reset for GPIO IRQ0, so a new interrupt can be initiated immediately 1: wait for key release after interrupt was reset for IRQ0 | 0x0 |
| 7:6 | - | - | Reserved | 0x0 |

Table 247: GPIO_INT_LEVEL_CTRL_REG (0x50001410)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------------|---|-------|
| 4 | r/w | INPUT_LEVEL4 | see INPUT_LEVEL0, but for GPIO IRQ4 | 0x0 |
| 3 | r/w | INPUT_LEVEL3 | see INPUT_LEVEL0, but for GPIO IRQ3 | 0x0 |
| 2 | r/w | INPUT_LEVEL2 | see INPUT_LEVEL0, but for GPIO IRQ2 | 0x0 |
| 1 | r/w | INPUT_LEVEL1 | see INPUT_LEVEL0, but for GPIO IRQ1 | 0x0 |
| 0 | r/w | INPUT_LEVEL0 | 0 = selected input will generate GPIO IRQ0 if that input is high. 1 = selected input will generate GPIO IRQ0 if that input is low. | 0x0 |

Table 248: KBRD_IRQ_IN_SEL0_REG (0x50001412)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-------------|---|-------|
| 15 | r/w | KBRD_REL | 0 = No interrupt on key release 1 = Interrupt also on key release (also debouncing if enabled) | 0x0 |
| 14 | r/w | KBRD_LEVEL | 0 = enabled input will generate KBRD IRQ if that input is high. 1 = enabled input will generate KBRD IRQ if that input is low. | 0x0 |
| 13:8 | r/w | KEY_REPEAT | While key is pressed, automatically generate repeating KEYB_INT after specified time unequal to 0. Repeat time: $N \times 1$ ms. $N = 1..63$, $N=0$ disables the timer. | 0x0 |
| 7 | r/w | KBRD_P07_EN | enable P0[7] for the keyboard interrupt | 0x0 |
| 6 | r/w | KBRD_P06_EN | enable P0[6] for the keyboard interrupt | 0x0 |
| 5 | r/w | KBRD_P05_EN | enable P0[5] for the keyboard interrupt | 0x0 |
| 4 | r/w | KBRD_P04_EN | enable P0[4] for the keyboard interrupt | 0x0 |
| 3 | r/w | KBRD_P03_EN | enable P0[3] for the keyboard interrupt | 0x0 |
| 2 | r/w | KBRD_P02_EN | enable P0[2] for the keyboard interrupt | 0x0 |
| 1 | r/w | KBRD_P01_EN | enable P0[1] for the keyboard interrupt | 0x0 |
| 0 | r/w | KBRD_P00_EN | enable P0[0] for the keyboard interrupt | 0x0 |

Table 249: KBRD_IRQ_IN_SEL1_REG (0x50001414)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-------------|---|-------|
| 15 | r/w | KBRD_P15_EN | enable P1[5] for the keyboard interrupt | 0x0 |
| 14 | r/w | KBRD_P14_EN | enable P1[4] for the keyboard interrupt | 0x0 |
| 13 | r/w | KBRD_P13_EN | enable P1[3] for the keyboard interrupt | 0x0 |
| 12 | r/w | KBRD_P12_EN | enable P1[2] for the keyboard interrupt | 0x0 |
| 11 | r/w | KBRD_P11_EN | enable P1[1] for the keyboard interrupt | 0x0 |
| 10 | r/w | KBRD_P10_EN | enable P1[0] for the keyboard interrupt | 0x0 |
| 9 | r/w | KBRD_P29_EN | enable P2[9] for the keyboard interrupt | 0x0 |
| 8 | r/w | KBRD_P28_EN | enable P2[8] for the keyboard interrupt | 0x0 |
| 7 | r/w | KBRD_P27_EN | enable P2[7] for the keyboard interrupt | 0x0 |
| 6 | r/w | KBRD_P26_EN | enable P2[6] for the keyboard interrupt | 0x0 |
| 5 | r/w | KBRD_P25_EN | enable P2[5] for the keyboard interrupt | 0x0 |
| 4 | r/w | KBRD_P24_EN | enable P2[4] for the keyboard interrupt | 0x0 |
| 3 | r/w | KBRD_P23_EN | enable P2[3] for the keyboard interrupt | 0x0 |
| 2 | r/w | KBRD_P22_EN | enable P2[2] for the keyboard interrupt | 0x0 |

Table 249: KBRD_IRQ_IN_SEL1_REG (0x50001414)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-------------|---|-------|
| 1 | r/w | KBRD_P21_EN | enable P2[1] for the keyboard interrupt | 0x0 |
| 0 | r/w | KBRD_P20_EN | enable P2[0] for the keyboard interrupt | 0x0 |

Table 250: KBRD_IRQ_IN_SEL2_REG (0x50001416)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-------------|---|-------|
| 7 | r/w | KBRD_P37_EN | enable P3[7] for the keyboard interrupt | 0x0 |
| 6 | r/w | KBRD_P36_EN | enable P3[6] for the keyboard interrupt | 0x0 |
| 5 | r/w | KBRD_P35_EN | enable P3[5] for the keyboard interrupt | 0x0 |
| 4 | r/w | KBRD_P34_EN | enable P3[4] for the keyboard interrupt | 0x0 |
| 3 | r/w | KBRD_P33_EN | enable P3[3] for the keyboard interrupt | 0x0 |
| 2 | r/w | KBRD_P32_EN | enable P3[2] for the keyboard interrupt | 0x0 |
| 1 | r/w | KBRD_P31_EN | enable P3[1] for the keyboard interrupt | 0x0 |
| 0 | r/w | KBRD_P30_EN | enable P3[0] for the keyboard interrupt | 0x0 |

Table 251: GP_ADC_CTRL_REG (0x50001500)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-----------------|--|-------|
| 15 | r/w | GP_ADC_LDO_ZERO | Forces LDO-output to 0V. | 0x0 |
| 14 | r/w | GP_ADC_LDO_EN | Turns on LDO. | 0x0 |
| 13 | r/w | GP_ADC_CHOP | Takes two samples with opposite GP_ADC_SIGN to cancel the internal offset voltage of the ADC; Highly recommended for DC-measurements. | 0x0 |
| 12 | r/w | GP_ADC_MUTE | Takes sample at mid-scale (to determine the internal offset and/or noise of the ADC with regards to VDD_REF which is also sampled by the ADC). | 0x0 |
| 11 | r/w | GP_ADC_SE | 0 = Differential mode 1 = Single ended mode | 0x0 |
| 10 | r/w | GP_ADC_SIGN | 0 = Default 1 = Conversion with opposite sign at input and output to cancel out the internal offset of the ADC and low-frequency | 0x0 |
| 9:6 | r/w | GP_ADC_SEL | ADC input selection which must be set before the GP_ADC_START bit is enabled. If GP_ADC_SE = 1 (single ended mode): 0000 = P0[0] 0001 = P0[1] 0010 = P0[2] 0011 = P0[3] 0100 = AVS 0101 = VDD_REF 0110 = VDD_RTT 0111 = VBAT3V 1000 = VDCDC 1001 = VBAT1V All other combinations are reserved. If GP_ADC_SE = 0 (differential mode): 0000 = P0[0] vs P0[1] All other combinations are P0[2] vs P0[3]. | 0x0 |
| 5 | r/w | GP_ADC_MINT | 0 = Disable (mask) GP_ADC_INT. 1 = Enable GP_ADC_INT to ICU. | 0x0 |

Table 251: GP_ADC_CTRL_REG (0x50001500)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|----------------|---|-------|
| 4 | r | GP_ADC_INT | 1 = AD conversion ready and has generated an interrupt. Must be cleared by writing any value to GP_ADC_CLEAR_INT_REG. | 0x0 |
| 3 | r/w | GP_ADC_CLK_SEL | 0 = Internal high-speed ADC clock used. 1 = Digital clock used. | 0x0 |
| 2 | rsvd | GP_ADC_TEST | Reserved, keep 0. | 0x0 |
| 1 | r/w | GP_ADC_START | 0 = ADC conversion ready. 1 = If a 1 is written, the ADC starts a conversion. After the conversion this bit will be set to 0 and the GP_ADC_INT bit will be set. | 0x0 |
| 0 | r/w | GP_ADC_EN | 0 = ADC is disabled and in reset. 1 = ADC is enabled and sampling of input is started. | 0x0 |

Table 252: GP_ADC_CTRL2_REG (0x50001502)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-----------------|---|-------|
| 15:4 | - | - | Reserved | 0x0 |
| 3 | r/w | GP_ADC_I20U | Adds 20uA constant load current at the ADC LDO to minimize ripple on the reference voltage of the ADC. | 0x0 |
| 2 | r/w | GP_ADC_IDYN | Enables dynamic load current at the ADC LDO to minimize ripple on the reference voltage of the ADC. | 0x0 |
| 1 | r/w | GP_ADC_ATTN3X | 0 = Input voltages up to 1.2V allowed. 1 = Input voltages up to 3.6V allowed by enabling 3x attenuator. | 0x0 |
| 0 | r/w | GP_ADC_DELAY_EN | Enables delay function for several signals. This is not auto-cleared. Toggle this bit before every sampling to enable successive conversions. | 0x0 |

Table 253: GP_ADC_OFFP_REG (0x50001504)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-------------|---|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:0 | r/w | GP_ADC_OFFP | Offset adjust of 'positive' array of ADC-network (effective if "GP_ADC_SE=0", or "GP_ADC_SE=1 AND GP_ADC_SIGN=0") | 0x200 |

Table 254: GP_ADC_OFFN_REG (0x50001506)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-------------|---|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:0 | r/w | GP_ADC_OFFN | Offset adjust of 'negative' array of ADC-network (effective if "GP_ADC_SE=0", or "GP_ADC_SE=1 AND GP_ADC_SIGN=1") | 0x200 |

Table 255: GP_ADC_CLEAR_INT_REG (0x50001508)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------------|---|-------|
| 15:0 | r0/w | GP_ADC_CLR_INT | Writing any value to this register will clear the ADC_INT interrupt. Reading returns 0. | 0x0 |

Table 256: GP_ADC_RESULT_REG (0x5000150A)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|------------|---|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:0 | r | GP_ADC_VAL | Returns the 10 bits linear value of the last AD conversion. | 0x0 |

Table 257: GP_ADC_DELAY_REG (0x5000150C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------|--|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | DEL_LDO_EN | Defines the delay before the LDO enable (GP_ADC_LDO_EN). Reset value is 0 μ s since the LDO enable should be the first thing to be programmed in the sequence of bringing the GP ADC up. | 0x0 |

Table 258: GP_ADC_DELAY2_REG (0x5000150E)

| Bit | Mode | Symbol | Description | Reset |
|------|------|---------------|---|-------|
| 15:8 | r/w | DEL_ADC_START | Defines the delay for the GP_ADC_START bit. Reset value is 17 μ s which is the recommended value to wait before starting the GP ADC. This is the third and last step of bringing up the GP ADC. | 0x88 |
| 7:0 | r/w | DEL_ADC_EN | Defines the delay for the GP_ADC_EN bit. Reset value is 16 μ s which is the recommended value to wait after enabling the LDO. This is the second step in bringing up the GP ADC. | 0x80 |

Table 259: CLK_REF_SEL_REG (0x50001600)

| Bit | Mode | Symbol | Description | Reset |
|------|------|---------------|--|-------|
| 15:3 | - | - | Reserved | 0x0 |
| 2 | r/w | REF_CAL_START | Writing a '1' starts a calibration. This bit is cleared when calibration is finished, and CLK_REF_VAL is ready. | 0x0 |
| 1:0 | r/w | REF_CLK_SEL | Select clock input for calibration: 0x0 : RC32KHz oscillator 0x1 : RC16MHz oscillator 0x2 : XTAL32KHz oscillator 0x3 : RCX32KHz oscillator | 0x0 |

Table 260: CLK_REF_CNT_REG (0x50001602)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-------------|--|-------|
| 15:0 | r/w | REF_CNT_VAL | Indicates the calibration time, with a decrement counter to 1. | 0x0 |

Table 261: CLK_REF_VAL_L_REG (0x50001604)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------|--|-------|
| 15:0 | r | XTAL_CNT_VAL | Returns the lower 16 bits of XTAL16 clock cycles during the calibration time, defined with REF_CNT_VAL | 0x0 |

Table 262: CLK_REF_VAL_H_REG (0x50001606)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------|--|-------|
| 15:0 | r | XTAL_CNT_VAL | Returns the upper 16 bits of XTAL16 clock cycles during the calibration time, defined with REF_CNT_VAL | 0x0 |

Table 263: P0_DATA_REG (0x50003000)

| Bit | Mode | Symbol | Description | Reset |
|------|------|---------|---|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | P0_DATA | Set P0 output register when written; Returns the value of P0 port when read | 0x0 |

Table 264: P0_SET_DATA_REG (0x50003002)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|---|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | P0_SET | Writing a 1 to P0[y] sets P0[y] to 1. Writing 0 is discarded; Reading returns 0 | 0x0 |

Table 265: P0_RESET_DATA_REG (0x50003004)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------|---|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | P0_RESET | Writing a 1 to P0[y] sets P0[y] to 0. Writing 0 is discarded; Reading returns 0 | 0x0 |

Table 266: P00_MODE_REG (0x50003006)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|---|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In ADC mode, these bits are don't care | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |

Table 266: P00_MODE_REG (0x50003006)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|--|-------|
| 4:0 | r/w | PID | Function of port 0 = Port function, PUPD as set above 1 = UART1_RX 2 = UART1_TX 3 = UART2_RX 4 = UART2_TX 5 = SPI_DI 6 = SPI_DO 7 = SPI_CLK 8 = SPI_EN 9 = I2C_SCL 10 = I2C_SDA 11 = UART1_IRDA_RX 12 = UART1_IRDA_TX 13 = UART2_IRDA_RX 14 = UART2_IRDA_TX 15 = ADC (only for P0[3:0]) 16 = PWM0 17 = PWM1 18 = BLE_DIAG (only for P0[7:0]) 19 = UART1_CTSN 20 = UART1_RTSN 21 = UART2_CTSN 22 = UART2_RTSN 23 = PWM2 24 = PWM3 25 = PWM4 Note: when a certain input function (like SPI_DI) is selected on more than 1 port pin, the port with the lowest index has the highest priority and P0 has higher priority than P1. | 0x0 |

Table 267: P01_MODE_REG (0x50003008)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|---|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In ADC mode, these bits are don't care | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |

Table 267: P01_MODE_REG (0x50003008)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|--|-------|
| 4:0 | r/w | PID | Function of port 0 = Port function, PUPD as set above 1 = UART1_RX 2 = UART1_TX 3 = UART2_RX 4 = UART2_TX 5 = SPI_DI 6 = SPI_DO 7 = SPI_CLK 8 = SPI_EN 9 = I2C_SCL 10 = I2C_SDA 11 = UART1_IRDA_RX 12 = UART1_IRDA_TX 13 = UART2_IRDA_RX 14 = UART2_IRDA_TX 15 = ADC (only for P0[3:0]) 16 = PWM0 17 = PWM1 18 = BLE_DIAG (only for P0[7:0]) 19 = UART1_CTSN 20 = UART1_RTSN 21 = UART2_CTSN 22 = UART2_RTSN 23 = PWM2 24 = PWM3 25 = PWM4 Note: when a certain input function (like SPI_DI) is selected on more than 1 port pin, the port with the lowest index has the highest priority and P0 has higher priority than P1. | 0x0 |

Table 268: P02_MODE_REG (0x5000300A)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|---|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In ADC mode, these bits are don't care | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |

Table 268: P02_MODE_REG (0x5000300A)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|--|-------|
| 4:0 | r/w | PID | Function of port 0 = Port function, PUPD as set above 1 = UART1_RX 2 = UART1_TX 3 = UART2_RX 4 = UART2_TX 5 = SPI_DI 6 = SPI_DO 7 = SPI_CLK 8 = SPI_EN 9 = I2C_SCL 10 = I2C_SDA 11 = UART1_IRDA_RX 12 = UART1_IRDA_TX 13 = UART2_IRDA_RX 14 = UART2_IRDA_TX 15 = ADC (only for P0[3:0]) 16 = PWM0 17 = PWM1 18 = BLE_DIAG (only for P0[7:0]) 19 = UART1_CTSN 20 = UART1_RTSN 21 = UART2_CTSN 22 = UART2_RTSN 23 = PWM2 24 = PWM3 25 = PWM4 Note: when a certain input function (like SPI_DI) is selected on more than 1 port pin, the port with the lowest index has the highest priority and P0 has higher priority than P1. | 0x0 |

Table 269: P03_MODE_REG (0x5000300C)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|---|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In ADC mode, these bits are don't care | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |

Table 269: P03_MODE_REG (0x5000300C)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|--|-------|
| 4:0 | r/w | PID | Function of port 0 = Port function, PUPD as set above 1 = UART1_RX 2 = UART1_TX 3 = UART2_RX 4 = UART2_TX 5 = SPI_DI 6 = SPI_DO 7 = SPI_CLK 8 = SPI_EN 9 = I2C_SCL 10 = I2C_SDA 11 = UART1_IRDA_RX 12 = UART1_IRDA_TX 13 = UART2_IRDA_RX 14 = UART2_IRDA_TX 15 = ADC (only for P0[3:0]) 16 = PWM0 17 = PWM1 18 = BLE_DIAG (only for P0[7:0]) 19 = UART1_CTSN 20 = UART1_RTSN 21 = UART2_CTSN 22 = UART2_RTSN 23 = PWM2 24 = PWM3 25 = PWM4 Note: when a certain input function (like SPI_DI) is selected on more than 1 port pin, the port with the lowest index has the highest priority and P0 has higher priority than P1. | 0x0 |

Table 270: P04_MODE_REG (0x5000300E)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|---|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In ADC mode, these bits are don't care | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |

Table 270: P04_MODE_REG (0x5000300E)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|--|-------|
| 4:0 | r/w | PID | Function of port 0 = Port function, PUPD as set above 1 = UART1_RX 2 = UART1_TX 3 = UART2_RX 4 = UART2_TX 5 = SPI_DI 6 = SPI_DO 7 = SPI_CLK 8 = SPI_EN 9 = I2C_SCL 10 = I2C_SDA 11 = UART1_IRDA_RX 12 = UART1_IRDA_TX 13 = UART2_IRDA_RX 14 = UART2_IRDA_TX 15 = ADC (only for P0[3:0]) 16 = PWM0 17 = PWM1 18 = BLE_DIAG (only for P0[7:0]) 19 = UART1_CTSN 20 = UART1_RTSN 21 = UART2_CTSN 22 = UART2_RTSN 23 = PWM2 24 = PWM3 25 = PWM4 Note: when a certain input function (like SPI_DI) is selected on more than 1 port pin, the port with the lowest index has the highest priority and P0 has higher priority than P1. | 0x0 |

Table 271: P05_MODE_REG (0x50003010)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|---|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In ADC mode, these bits are don't care | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |

Table 271: P05_MODE_REG (0x50003010)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|--|-------|
| 4:0 | r/w | PID | Function of port 0 = Port function, PUPD as set above 1 = UART1_RX 2 = UART1_TX 3 = UART2_RX 4 = UART2_TX 5 = SPI_DI 6 = SPI_DO 7 = SPI_CLK 8 = SPI_EN 9 = I2C_SCL 10 = I2C_SDA 11 = UART1_IRDA_RX 12 = UART1_IRDA_TX 13 = UART2_IRDA_RX 14 = UART2_IRDA_TX 15 = ADC (only for P0[3:0]) 16 = PWM0 17 = PWM1 18 = BLE_DIAG (only for P0[7:0]) 19 = UART1_CTSN 20 = UART1_RTSN 21 = UART2_CTSN 22 = UART2_RTSN 23 = PWM2 24 = PWM3 25 = PWM4 Note: when a certain input function (like SPI_DI) is selected on more than 1 port pin, the port with the lowest index has the highest priority and P0 has higher priority than P1. | 0x0 |

Table 272: P06_MODE_REG (0x50003012)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|---|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In ADC mode, these bits are don't care | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |

Table 272: P06_MODE_REG (0x50003012)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|--|-------|
| 4:0 | r/w | PID | Function of port 0 = Port function, PUPD as set above 1 = UART1_RX 2 = UART1_TX 3 = UART2_RX 4 = UART2_TX 5 = SPI_DI 6 = SPI_DO 7 = SPI_CLK 8 = SPI_EN 9 = I2C_SCL 10 = I2C_SDA 11 = UART1_IRDA_RX 12 = UART1_IRDA_TX 13 = UART2_IRDA_RX 14 = UART2_IRDA_TX 15 = ADC (only for P0[3:0]) 16 = PWM0 17 = PWM1 18 = BLE_DIAG (only for P0[7:0]) 19 = UART1_CTSN 20 = UART1_RTSN 21 = UART2_CTSN 22 = UART2_RTSN 23 = PWM2 24 = PWM3 25 = PWM4 Note: when a certain input function (like SPI_DI) is selected on more than 1 port pin, the port with the lowest index has the highest priority and P0 has higher priority than P1. | 0x0 |

Table 273: P07_MODE_REG (0x50003014)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|---|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In ADC mode, these bits are don't care | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |

Table 273: P07_MODE_REG (0x50003014)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|--|-------|
| 4:0 | r/w | PID | Function of port 0 = Port function, PUPD as set above 1 = UART1_RX 2 = UART1_TX 3 = UART2_RX 4 = UART2_TX 5 = SPI_DI 6 = SPI_DO 7 = SPI_CLK 8 = SPI_EN 9 = I2C_SCL 10 = I2C_SDA 11 = UART1_IRDA_RX 12 = UART1_IRDA_TX 13 = UART2_IRDA_RX 14 = UART2_IRDA_TX 15 = ADC (only for P0[3:0]) 16 = PWM0 17 = PWM1 18 = BLE_DIAG (only for P0[7:0]) 19 = UART1_CTSN 20 = UART1_RTSN 21 = UART2_CTSN 22 = UART2_RTSN 23 = PWM2 24 = PWM3 25 = PWM4 Note: when a certain input function (like SPI_DI) is selected on more than 1 port pin, the port with the lowest index has the highest priority and P0 has higher priority than P1. | 0x0 |

Table 274: P1_DATA_REG (0x50003020)

| Bit | Mode | Symbol | Description | Reset |
|------|------|---------|---|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | P1_DATA | Set P1 output register when written; Returns the value of P1 port when read | 0x0 |

Table 275: P1_SET_DATA_REG (0x50003022)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|---|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | P1_SET | Writing a 1 to P1[y] sets P1[y] to 1. Writing 0 is discarded; Reading returns 0 | 0x0 |

Table 276: P1_RESET_DATA_REG (0x50003024)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------|---|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | P1_RESET | Writing a 1 to P1[y] sets P1[y] to 0. Writing 0 is discarded; Reading returns 0 | 0x0 |

Table 277: P10_MODE_REG (0x50003026)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|--|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In analog mode, these bits are don't care P14_MODE_REG and P15_MODE_REG reset value is 1 (i.e. pulled up) | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0x0 |

Table 278: P11_MODE_REG (0x50003028)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|--|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In analog mode, these bits are don't care P14_MODE_REG and P15_MODE_REG reset value is 1 (i.e. pulled up) | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0x0 |

Table 279: P12_MODE_REG (0x5000302A)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|--|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In analog mode, these bits are don't care P14_MODE_REG and P15_MODE_REG reset value is 1 (i.e. pulled up) | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0x0 |

Table 280: P13_MODE_REG (0x5000302C)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|--|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In analog mode, these bits are don't care P14_MODE_REG and P15_MODE_REG reset value is 1 (i.e. pulled up) | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |

Table 280: P13_MODE_REG (0x5000302C)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|-----------------------|-------|
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0x0 |

Table 281: P14_MODE_REG (0x5000302E)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|--|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In analog mode, these bits are don't care P14_MODE_REG and P15_MODE_REG reset value is 1 (i.e. pulled up) | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0x0 |

Table 282: P15_MODE_REG (0x50003030)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|--|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In analog mode, these bits are don't care P14_MODE_REG and P15_MODE_REG reset value is 1 (i.e. pulled up) | 0x1 |
| 7:5 | - | - | Reserved | 0x0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0x0 |

Table 283: P2_DATA_REG (0x50003040)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|---------|---|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:0 | r/w | P2_DATA | Set P2 output register when written; Returns the value of P2 port when read | 0x0 |

Table 284: P2_SET_DATA_REG (0x50003042)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|--|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:0 | r/w | P2_SET | Writing a 1 to P2[y] sets P2[y] to 1. Writing 0 is discarded; Reading returns 0 | 0x0 |

Table 285: P2_RESET_DATA_REG (0x50003044)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|----------|--|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:0 | r/w | P2_RESET | Writing a 1 to P2[y] sets P2[y] to 0. Writing 0 is discarded; Reading returns 0 | 0x0 |

Table 286: P20_MODE_REG (0x50003046)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|--|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In analog mode, these bits are don't care | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0x0 |

Table 287: P21_MODE_REG (0x50003048)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|--|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In analog mode, these bits are don't care | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0x0 |

Table 288: P22_MODE_REG (0x5000304A)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|--|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In analog mode, these bits are don't care | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0x0 |

Table 289: P23_MODE_REG (0x5000304C)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|--|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In analog mode, these bits are don't care | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0x0 |

Table 290: P24_MODE_REG (0x5000304E)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|-------------|-------|
| 15:10 | - | - | Reserved | 0x0 |

Table 290: P24_MODE_REG (0x5000304E)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|--|-------|
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In analog mode, these bits are don't care | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0x0 |

Table 291: P25_MODE_REG (0x50003050)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|--|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In analog mode, these bits are don't care | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0x0 |

Table 292: P26_MODE_REG (0x50003052)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|--|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In analog mode, these bits are don't care | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0x0 |

Table 293: P27_MODE_REG (0x50003054)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|--|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In analog mode, these bits are don't care | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0x0 |

Table 294: P28_MODE_REG (0x50003056)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|-------------|-------|
| 15:10 | - | - | Reserved | 0x0 |

Table 294: P28_MODE_REG (0x50003056)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|--|-------|
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In analog mode, these bits are don't care | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0x0 |

Table 295: P29_MODE_REG (0x50003058)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|--|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In analog mode, these bits are don't care | 0x2 |
| 7:5 | - | - | Reserved | 0x0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0x0 |

Table 296: P01_PADPWR_CTRL_REG (0x50003070)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-------------|---|-------|
| 15:14 | - | - | Reserved | 0x0 |
| 13:8 | r/w | P1_OUT_CTRL | 1 = P1_x port output is powered by the 1 V rail 0 = P1_x port output is powered by the 3 V rail bit 8 controls the power of P1[0], bit 13 controls the power of P1[5] (Note 21) | 0x0 |
| 7:0 | r/w | P0_OUT_CTRL | 1 = P0_x port output is powered by the 1 V rail 0 = P0_x port output is powered by the 3 V rail bit 0 controls the power of P0[0], bit 7 controls the power of P0[7] (Note 22) | 0x0 |

Note 21: In Buck mode the output must be powered by the 3 V rail. In Boost mode the outputs can be powered by the 1 V rail or by the 3 V rail. In Boost mode the 3 V rail can only supply a limited current, e.g. for switching a high-impedance input of an external device. See table 'Digital input/output characteristics'.

Note 22: In Buck mode the output must be powered by the 3 V rail. In Boost mode the outputs can be powered by the 1 V rail or by the 3 V rail. In Boost mode the 3 V rail can only supply a limited current, e.g. for switching a high-impedance input of an external device. See table 'Digital input/output characteristics'.

Table 297: P2_PADPWR_CTRL_REG (0x50003072)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-------------|---|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:0 | r/w | P2_OUT_CTRL | 1 = P2_x port output is powered by the 1 V rail 0 = P2_x port output is powered by the 3 V rail bit 0 controls the power of P2[0], bit 9 controls the power of P2[9], (Note 23) | 0x0 |

Note 23: In Buck mode the output must be powered by the 3 V rail. In Boost mode the outputs can be powered by the 1 V rail or by the 3 V rail. In Boost mode the 3 V rail can only supply a limited current, e.g. for switching a high-impedance input of an external device. See table 'Digital input/output characteristics'.

input/output characteristics'.

Table 298: P3_PADPWR_CTRL_REG (0x50003074)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-------------|---|-------|
| 15:8 | - | - | Reserved | 0 |
| 7:0 | r/w | P3_OUT_CTRL | 1 = P3_x port output is powered by the 1 V rail 0 = P3_x port output is powered by the 3 V rail bit 0 controls the power of P3[0], bit 7 controls the power of P3[7], (Note 24) | 0 |

Note 24: In Buck mode the output must be powered by the 3 V rail. In Boost mode the outputs can be powered by the 1 V rail or by the 3 V rail. In Boost mode the 3 V rail can only supply a limited current, e.g. for switching a high-impedance input of an external device. See table 'Digital input/output characteristics'.

Table 299: P3_DATA_REG (0x50003080)

| Bit | Mode | Symbol | Description | Reset |
|------|------|---------|---|-------|
| 15:8 | - | - | Reserved | 0 |
| 7:0 | r/w | P3_DATA | Set P3 output register when written; Returns the value of P3 port when read | 0 |

Table 300: P3_SET_DATA_REG (0x50003082)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|--|-------|
| 15:8 | - | - | Reserved | 0 |
| 7:0 | r0/w | P3_SET | Writing a 1 to P3[y] sets P3[y] to 1. Writing 0 is discarded; Reading returns 0 | 0 |

Table 301: P3_RESET_DATA_REG (0x50003084)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------|--|-------|
| 15:8 | - | - | Reserved | 0 |
| 7:0 | r0/w | P3_RESET | Writing a 1 to P0[y] sets P0[y] to 0. Writing 0 is discarded; Reading returns 0 | 0 |

Table 302: P30_MODE_REG (0x50003086)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|---|-------|
| 15:10 | - | - | Reserved | 0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In ADC mode, these bits are don't care | 2 |
| 7:5 | - | - | Reserved | 0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0 |

Table 303: P31_MODE_REG (0x50003088)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|-------------|-------|
| 15:10 | - | - | Reserved | 0 |

Table 303: P31_MODE_REG (0x50003088)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|---|-------|
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In ADC mode, these bits are don't care | 2 |
| 7:5 | - | - | Reserved | 0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0 |

Table 304: P32_MODE_REG (0x5000308A)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|---|-------|
| 15:10 | - | - | Reserved | 0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In ADC mode, these bits are don't care | 2 |
| 7:5 | - | - | Reserved | 0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0 |

Table 305: P33_MODE_REG (0x5000308C)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|---|-------|
| 15:10 | - | - | Reserved | 0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In ADC mode, these bits are don't care | 2 |
| 7:5 | - | - | Reserved | 0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0 |

Table 306: P34_MODE_REG (0x5000308E)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|---|-------|
| 15:10 | - | - | Reserved | 0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In ADC mode, these bits are don't care | 2 |
| 7:5 | - | - | Reserved | 0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0 |

Table 307: P35_MODE_REG (0x50003090)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|-------------|-------|
| 15:10 | - | - | Reserved | 0 |

Table 307: P35_MODE_REG (0x50003090)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|---|-------|
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In ADC mode, these bits are don't care | 2 |
| 7:5 | - | - | Reserved | 0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0 |

Table 308: P36_MODE_REG (0x50003092)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|---|-------|
| 15:10 | - | - | Reserved | 0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In ADC mode, these bits are don't care | 2 |
| 7:5 | - | - | Reserved | 0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0 |

Table 309: P37_MODE_REG (0x50003094)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|---|-------|
| 15:10 | - | - | Reserved | 0 |
| 9:8 | r/w | PUPD | 00 = Input, no resistors selected 01 = Input, pull-up selected 10 = Input, Pull-down selected 11 = Output, no resistors selected In ADC mode, these bits are don't care | 2 |
| 7:5 | - | - | Reserved | 0 |
| 4:0 | r/w | PID | See P0x_MODE_REG[PID] | 0 |

Table 310: WATCHDOG_REG (0x50003100)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------|---|-------|
| 15:9 | r0/w | WDOG_WEN | 0000.000 = Write enable for Watchdog timer else Write disable. This filter prevents unintentional presetting the watchdog with a SW run-away. | 0x0 |
| 8 | r/w | WDOG_VAL_NEG | 0 = Watchdog timer value is positive. 1 = Watchdog timer value is negative. | 0x0 |
| 7:0 | r/w | WDOG_VAL | <u>Write</u> : Watchdog timer reload value. Note that all bits 15-9 must be 0 to reload this register. <u>Read</u> : Actual Watchdog timer value. Decrement by 1 every 10.24 ms. Bit 8 indicates a negative counter value. 2, 1, 0, 1FF ₁₆ , 1FE ₁₆ etc. An NMI or WDOG (SYS) reset is generated under the following conditions: If WATCHDOG_CTRL_REG[NMI_RST] = 0 then If WDOG_VAL = 0 -> NMI (Non Maskable Interrupt) if WDOG_VAL = 1F0 ₁₆ -> WDOG reset -> reload FF ₁₆ If WATCHDOG_CTRL_REG[NMI_RST] = 1 then if WDOG_VAL <= 0 -> WDOG reset -> reload FF ₁₆ | 0xFF |

Table 311: WATCHDOG_CTRL_REG (0x50003102)

| Bit | Mode | Symbol | Description | Reset |
|------|------|---------|---|-------|
| 15:2 | - | - | Reserved | 0x0 |
| 1 | - | - | Reserved | 0x0 |
| 0 | r/w | NMI_RST | <p>0 = Watchdog timer generates NMI at value 0, and WDOG (SYS) reset at <math>\leq -16</math>. Timer can be frozen /resumed using SET_FREEZE_REG[FRZ_WDOG]/ RESET_FREEZE_REG[FRZ_WDOG].</p> <p>1 = Watchdog timer generates a WDOG (SYS) reset at value 0 and can not be frozen by Software. Note that this bit can only be set to 1 by SW and only be reset with a WDOG (SYS) reset or SW reset.</p> <p>The watchdog is always frozen when the Cortex-M0 is halted in DEBUG State.</p> | 0x0 |

Table 312: CHIP_ID1_REG (0x50003200)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|----------|--|-------|
| 7:0 | r | CHIP_ID1 | First character of device type "580" in ASCII. | 0x35 |

Table 313: CHIP_ID2_REG (0x50003201)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|----------|---|-------|
| 7:0 | r | CHIP_ID2 | Second character of device type "580" in ASCII. | 0x38 |

Table 314: CHIP_ID3_REG (0x50003202)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|----------|--|-------|
| 7:0 | r | CHIP_ID3 | Third character of device type "580" in ASCII. | 0x30 |

Table 315: CHIP_SWC_REG (0x50003203)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|----------|--|-------|
| 7:4 | - | - | Reserved | 0x0 |
| 3:0 | r | CHIP_SWC | <p>SoftWare Compatibility code. Integer (default = 0) which is incremented if a silicon change has impact on the CPU Firmware. Can be used by software developers to write silicon revision dependent code.</p> | 0x0 |

Table 316: CHIP_REVISION_REG (0x50003204)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-------------|--|-------|
| 7:0 | r | REVISION_ID | Chip version, corresponds with type number in ASCII. 0x41 = 'A', 0x42 = 'B' | 0x41 |

Table 317: SET_FREEZE_REG (0x50003300)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:4 | - | - | Reserved | 0x0 |

Table 317: SET_FREEZE_REG (0x50003300)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-------------|--|-------|
| 3 | r/w | FRZ_WDOG | If '1', the watchdog timer is frozen, '0' is discarded. WATCHDOG_CTRL_REG[NMI_RST] must be '0' to allow the freeze function. | 0x0 |
| 2 | r/w | FRZ_BLETIM | If '1', the BLE master clock is frozen, '0' is discarded. | 0x0 |
| 1 | r/w | FRZ_SWTIM | If '1', the SW Timer (TIMER0) is frozen, '0' is discarded. | 0x0 |
| 0 | r/w | FRZ_WKUPTIM | If '1', the Wake Up Timer is frozen, '0' is discarded. | 0x0 |

Table 318: RESET_FREEZE_REG (0x50003302)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-------------|---|-------|
| 15:4 | - | - | Reserved | 0x0 |
| 3 | r/w | FRZ_WDOG | If '1', the watchdog timer continues, '0' is discarded. | 0x0 |
| 2 | r/w | FRZ_BLETIM | If '1', the the BLE master clock continues, '0' is discarded. | 0x0 |
| 1 | r/w | FRZ_SWTIM | If '1', the SW Timer (TIMER0) continues, '0' is discarded. | 0x0 |
| 0 | r/w | FRZ_WKUPTIM | If '1', the Wake Up Timer continues, '0' is discarded. | 0x0 |

Table 319: DEBUG_REG (0x50003304)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------------|--|-------|
| 15:1 | - | - | Reserved | 0x0 |
| 0 | r/w | DEBUGS_FREEZE_EN | Default '1', freezing of the on-chip timers is enabled when the Cortex-M0 is halted in DEBUG State. If '0', freezing of the on-chip timers is depending on FREEZE_REG when the Cortex-M0 is halted in DEBUG State <u>except</u> the watchdog timer. The watchdog timer is always frozen when the Cortex-M0 is halted in DEBUG State. | 0x1 |

Table 320: GP_STATUS_REG (0x50003306)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-----------|---|-------|
| 15:1 | - | - | Reserved | 0x0 |
| 0 | r/w | CAL_PHASE | If '1', it designates that the chip is in Calibration Phase i.e. the OTP has been initially programmed but no Calibration has occurred. | 0x0 |

Table 321: GP_CONTROL_REG (0x50003308)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:6 | - | - | Reserved | 0 |

Table 321: GP_CONTROL_REG (0x50003308)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|----------------|--|-------|
| 5:1 | r/w | EM_MAP | Select the mapping of the Exchange memory pages. 0: EM size 0 kB, SysRAM size 42 kB 1: EM size 2 kB, SysRAM size 48 kB 2: EM size 3 kB, SysRAM size 47 kB 3: EM size 4 kB, SysRAM size 46 kB 4: EM size 5 kB, SysRAM size 45 kB 5: EM size 6 kB, SysRAM size 44 kB 6: EM size 7 kB, SysRAM size 43 kB 7: EM size 8 kB, SysRAM size 42 kB 8: Reserved 9: EM size 4 kB, SysRAM size 40 kB 10: EM size 5 kB, SysRAM size 40 kB 11: EM size 6 kB, SysRAM size 40 kB 12: EM size 7 kB, SysRAM size 40 kB 13: EM size 8 kB, SysRAM size 40 kB 14: EM size 9 kB, SysRAM size 40 kB 15: EM size 10 kB, SysRAM size 40 kB 16: Reserved 17: EM size 6 kB, SysRAM size 38 kB 18: EM size 7 kB, SysRAM size 38 kB 19: EM size 8 kB, SysRAM size 38 kB 20: EM size 9 kB, SysRAM size 38 kB 21: EM size 10 kB, SysRAM size 38 kB 22: EM size 11 kB, SysRAM size 38 kB 23: EM size 12 kB, SysRAM size 38 kB other: Reserved. | 0x1 |
| 0 | r/w | BLE_WAKEUP_REQ | If '1', the BLE wakes up. Must be kept high at least for 1 low power clock period. If the BLE is in deep sleep state, then by setting this bit it will cause the wakeup LP IRQ to be asserted with a delay of 3 to 4 low power cycles. | 0x0 |

Table 322: TIMER0_CTRL_REG (0x50003400)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------|---|-------|
| 15:4 | - | - | Reserved | 0x0 |
| 3 | r/w | PWM_MODE | 0 = PWM signals are '1' during high time. 1 = PWM signals send out the (fast) clock divided by 2 during high time. So it will be in the range of 1 to 8 MHz. | 0x0 |
| 2 | r/w | TIM0_CLK_DIV | 1 = Timer0 uses selected clock frequency as is. 0 = Timer0 uses selected clock frequency divided by 10. Note that this applies only to the ON-counter. | 0x0 |
| 1 | r/w | TIM0_CLK_SEL | 1 = Timer0 uses 16, 8, 4 or 2 MHz (fast) clock frequency. 0 = Timer0 uses 32 kHz (slow) clock frequency. | 0x0 |
| 0 | r/w | TIM0_CTRL | 0 = Timer0 is off and in reset state. 1 = Timer0 is running. | 0x0 |

Table 323: TIMER0_ON_REG (0x50003402)

| Bit | Mode | Symbol | Description | Reset |
|------|------|---------|--|-------|
| 15:0 | r0/w | TIM0_ON | Timer0 On reload value: If read the actual counter value ON_CNTER is returned | 0x0 |

Table 324: TIMER0_RELOAD_M_REG (0x50003404)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|--|-------|
| 15:0 | r0/w | TIM0_M | Timer0 'high' reload value if read the actual counter value T0_CNTER is returned | 0x0 |

Table 325: TIMER0_RELOAD_N_REG (0x50003406)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|---|-------|
| 15:0 | r0/w | TIM0_N | Timer0 'low' reload value: if read the actual counter value T0_CNTER is returned | 0x0 |

Table 326: PWM2_DUTY_CYCLE (0x50003408)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------|--------------------|-------|
| 13:0 | r/w | DUTY_CYCLE | duty cycle for PWM | 0x0 |

Table 327: PWM3_DUTY_CYCLE (0x5000340A)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------|--------------------|-------|
| 13:0 | r/w | DUTY_CYCLE | duty cycle for PWM | 0x0 |

Table 328: PWM4_DUTY_CYCLE (0x5000340C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------|--------------------|-------|
| 13:0 | r/w | DUTY_CYCLE | duty cycle for PWM | 0x0 |

Table 329: TRIPLE_PWM_FREQUENCY (0x5000340E)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|--------------------|-------|
| 13:0 | r/w | FREQ | Freq for PWM 2 3 4 | 0x0 |

Table 330: TRIPLE_PWM_CTRL_REG (0x50003410)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-----------------------|------------------------------|-------|
| 2 | r/w | HW_PAUSE_EN | '1' = HW can pause PWM 2,3,4 | 0x1 |
| 1 | r/w | SW_PAUSE_EN | '1' = PWM 2 3 4 is paused | 0x0 |
| 0 | r/w | TRIPLE_PWM_ENA BLE | '1' = PWM 2 3 4 is enabled | 0x0 |

28 CODEC Registers

Table 331: Register Map CODEC

| Address | Port | Description |
|------------|---------------------|--|
| 0x00001200 | ENV_REG | Test environment register |
| 0x00001202 | TEST_CTRL_REG | Test control register |
| 0x00001210 | BANDGAP_REG | Bandgap register |
| 0x00001212 | BAT_CTRL_REG | Power control register |
| 0x00001220 | GPIO_DATA_REG | GPIO Data input /out register |
| 0x00001222 | GPIO_SET_DATA_REG | GPIO Set port pins register |
| 0x00001224 | GPIO_RESET_DATA_REG | GPIO Reset port pins register |
| 0x00001226 | GPIO_DIR_REG | GPIO Direction register |
| 0x0000122A | GPIO_PUPD_REG | GPIO Pull-up pull-down register |
| 0x00001250 | DMA0_A_STARTL_REG | Start address Low A of DMA channel 0 |
| 0x00001254 | DMA0_A_IDX_REG | DMA Receive index A of channel 0 |
| 0x00001256 | DMA0_B_STARTL_REG | Start address Low B of DMA channel 0 |
| 0x0000125A | DMA0_B_IDX_REG | DMA Receive B index channel 0 |
| 0x0000125E | DMA0_LEN_REG | DMA receive length register channel 0 |
| 0x00001260 | DMA0_CTRL_REG | Control register for the DMA channel 0 |
| 0x00001280 | CODEC_MIC_REG | CODEC microphone control register |
| 0x00001282 | CODEC_LSR_REG | CODEC loudspeaker control register |
| 0x00001284 | CODEC_VREF_REG | CODEC Vref control register |
| 0x00001286 | CODEC_IN_OUT_REG | CODEC input/output register |
| 0x00001288 | CODEC_ADDA_REG | CODEC ad/da control register |
| 0x0000128A | CODEC_OFFSET1_REG | CODEC offset error and compensation register |
| 0x0000128C | CODEC_TEST_CTRL_REG | CODEC test control register codec |
| 0x0000128E | CODEC_OFFSET2_REG | CODEC offset compensation register |
| 0x000012E2 | CLK_CTRL_REG | PLL control register |
| 0x000012E4 | PLL_DIV_REG | PLL control register |
| 0x000012E6 | PER_DIV_REG | Peripheral divider register |
| 0x000012E8 | CODEC_DIV_REG | CODEC divider register |
| 0x000012FA | VERSION_REG0 | Version register 0 |
| 0x000012FB | VERSION_REG1 | Version register 1 |
| 0x000012FC | VERSION_REG2 | Version register 2 |
| 0x000012FD | VERSION_REG3 | Version register 3 |
| 0x000012FE | VERSION_REG4 | Version register 4 |
| 0x000012FF | VERSION_REG5 | Version register 5 |

Table 332: ENV_REG (0x00001200)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------|---|-------|
| 15:4 | - | - | Reserved | 0x0 |
| 3:0 | r | ENV_DATA | Value of pins GPIO[3-2] on rising edge of RSTn. Used for test purposes Bit fields [1-0] are reserved | 0x0 |

Table 333: TEST_CTRL_REG (0x00001202)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-----------|---|-------|
| 15:0 | r/w | TEST_CTRL | For test purpose only. Must be all 0 for normal operation | 0x0 |

Table 334: BANDGAP_REG (0x00001210)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-------------|---|-------|
| 15:12 | - | - | Reserved | 0x0 |
| 11:10 | r | BG_COMP | Bandgap Trim comparator output (see also chapter "Bandgap reference") (valid when BG_COMP_PD = '0') 11: Bandgap voltage is to high (relative to VDD) 01: Bandgap voltage is OK (relative to VDD) 00: Bandgap voltage is to low (relative to VDD) | 0x0 |
| 9 | r/w | BG_COMP_PD | 0: Enable bandgap trim comparators 1: Bandgap trim comparators in power-down. | 0x0 |
| 8:6 | r/w | BANDGAP_I | Internal reference current adjust. Since the internal reference current can not be measured by the customer, these bits shall not be changed. Setting these bits will not affect the bandgap voltage. 1.00: + 40 % 1.01: + 30 % 1.10: + 20 % 1.11: + 10 % 0.00: + 0 % <- prescribed value if no current tuning done 0.01: - 10 % 0.10: - 20 % 0.11: - 30 % | 0x0 |
| 5:4 | r/w | BANDGAP_VIT | 1x: Voltage and current -5.6 % <- for test purposes only x1: Voltage and current +5.6 % <- for test purposes only | 0x0 |
| 3:0 | r/w | BANDGAP_VI | Bandgap reference voltage and current adjust to trim internal bandgap. The correct value of the bandgap can be checked with BG_COMP if BG_COMP_PD=0. These bits must be set before the current bits 8-6 because it affects the reference currents. 0000: -5.6 % 0001: -4.9 % 0010: -4.2 % 0011: -3.5 % 0100: -2.8 % 0101: -2.1 % 0110: -1.4 % 0111: -0.7 % 1000: mid level 1001: +0.7 % 1010: +1.4 % 1011: +2.1 % 1100: +2.8 % 1101: +3.5 % 1110: +4.2 % | 0x8 |

Table 335: BAT_CTRL_REG (0x00001212)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|-------------|-------|
| 15:2 | - | - | Reserved | 0x0 |

Table 335: BAT_CTRL_REG (0x00001212)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|---|-------|
| 1:0 | r/w | PADS_A | Reference voltage adjust for PAD input stage. This register must be set to a higher value before a higher VDDIO is applied and set to a lower value after a lower VDDIO is applied. Reversing this order may damage the PADS. 0x: For VDDIO: 1.75 ... 2.0 V 1x: For VDDIO: 2.01 .. 3.45 V | 0x3 |

Table 336: GPIO_DATA_REG (0x00001220)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|-----------|---|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | GPIO_DATA | Write, set pin GPIOx (if output enabled with GPIO_DIR_REG[x]) Read, returns the value of pin value GPIOx Bit fields [7-4] and [1-0] are reserved. | 0x0 |

Table 337: GPIO_SET_DATA_REG (0x00001222)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|----------|--|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:0 | r/w | GPIO_SET | Writing a '1' sets GPIO_DATA_REG[x] to '1' if output enabled with GPIO_DIR_REG[x] Writing 0 is discarded, Reading returns 0 Bit fields [7-4] and [1-0] are reserved. | 0x0 |

Table 338: GPIO_RESET_DATA_REG (0x00001224)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|------------|--|-------|
| 15:10 | - | - | Reserved | 0x0 |
| 9:0 | r/w | GPIO_RESET | Writing a '1' sets GPIO_DATA_REG[x] to '0' if output enabled with GPIO_DIR_REG[x] Writing 0 is discarded, Reading returns 0 Bit fields [7-4] and [1-0] are reserved. | 0x0 |

Table 339: GPIO_DIR_REG (0x00001226)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------|--|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | GPIO_DIR | 0: Pin GPIOx is input 1: Pin GPIOx is output Bit fields [7-4] and [1-0] are reserved. Keep reset value | 0x0 |

Table 340: GPIO_PUPD_REG (0x0000122A)

| Bit | Mode | Symbol | Description | Reset |
|------|------|-----------|--|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | GPIO_PUPD | 0: Pull-up resistor disabled on pin GPIOx if input 1: Pull-up resistor enabled on pin GPIOx if input Bit fields [7-4] and [1-0] are reserved. Keep 1 | 0xFF |

Table 341: DMA0_A_STARTL_REG (0x00001250)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|---------------|--|-------|
| 15:13 | - | - | Reserved | 0x0 |
| 12:0 | r/w | DMA0_A_STARTL | Start source/destination address A. The source/destination direction is set by bit DIR in DMA_CTRL_REGx. | 0x0 |

Table 342: DMA0_A_IDX_REG (0x00001254)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------|---|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | DMA0_A_IDX | This register, added to DMA_A_START_REGx, is the source/destination address of the next DMA cycle. Incremented with 0 or 2 after each DMA cycle. The source/destination direction is set by bit DIR in CTRL_REGx. | 0x0 |

Table 343: DMA0_B_STARTL_REG (0x00001256)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|---------------|--|-------|
| 15:13 | - | - | Reserved | 0x0 |
| 12:0 | r/w | DMA0_B_STARTL | Start source/destination address B. The source/destination direction is set by bit DIR in CTRL_REGx. | 0x0 |

Table 344: DMA0_B_IDX_REG (0x0000125A)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------|---|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | DMA0_B_IDX | This register, added to DMA_B_START_REGx, is the source/destination address of the next DMA cycle. Incremented with 0 or 2 after each DMA cycle. The source/destination direction is set by bit DIR in CTRL_REGx. | 0x0 |

Table 345: DMA0_LEN_REG (0x0000125E)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------|--|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7:0 | r/w | DMA0_LEN | Length in bytes (DMA_LEN must be $\leq 0xFF - AINC$) If this register equals DMA0_A_IDX_REG then if DMA_CIRCULAR = 0, then the DMA stops. DMA0_A_IDX_REG, DMA0_B_IDX_REG must be reset by SW if DMA_CIRCULAR = 1, then DMA_A_IDX_REG and DMA0_B_IDX_REG are both reset and DMA continues. | 0x0 |

Table 346: DMA0_CTRL_REG (0x00001260)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|------------|---|-------|
| 15 | - | - | Reserved | 0x0 |
| 14 | r/w | SYNC_SEL | 0: 8/16 kHz trigger for DMA by CODEC (Note: CODEC must be switched on) 1: Reserved | 0x0 |
| 13:12 | - | - | Reserved | 0x0 |
| 11 | r/w | DREQ_LEVEL | 0: normal operation 1: Stop DMA if DMA_ON is set to '0'. | 0x0 |

Table 346: DMA0_CTRL_REG (0x00001260)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|-----------|---|-------|
| 10 | r/w | CIRCULAR | 0: Normal mode. The DMA channel stops if register DMA_A_IDX_REGx equals register DMA0_LEN_REG. This mode is for SPI operation. 1: Circular mode. If DMA0_A_IDX_REG equals register DMA0_LEN_REG, the DMA0_A_IDX_REG and DMA0_B_IDX_REG are both reset and the DMA starts from the start. This mode is used to transfer CODEC samples | 0x0 |
| 9:8 | r/w | AINC | Increment count DMA0_A_IDX_REG register. 00: do not increment 01: Reserved 10: Increment address with one word | 0x0 |
| 7:6 | r/w | BINC | Increment count DMA0_B_IDX_REG register. 00: do not increment 01: Reserved 10: Increment address with one word | 0x0 |
| 5 | r/w | DREQ_MODE | 0: DMA channel starts immediately. 1: DMA channel triggered by 8/16 kHz Codec clock | 0x0 |
| 4 | - | - | Reserved | 0x0 |
| 3 | r/w | IND | Reserved, must be set to 1 | 0x0 |
| 2 | r/w | DIR | Transfer direction: 0: A to B 1: B to A | 0x0 |
| 1 | - | - | Reserved | 0x0 |
| 0 | r/w | DMA_ON | 0: DMA channel is off, clocks are disabled 1: DMA channel is on will be automatically cleared in normal mode after a complete transfer if DMA0_IDX_REG equals DMA0_LEN_REG. If DMA operation must be stopped before a transfer is completed, bit 11 must be set to '1' and DMA_ON must be set to '0' | 0x0 |

Table 347: CODEC_MIC_REG (0x00001280)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|---------------|---|-------|
| 15:13 | - | - | Reserved | 0x0 |
| 12 | - | - | Reserved | 0x0 |
| 11 | r/w | MICN_AGND_OUT | 0: MICn not connected to internal AGND 1: MICn connected to internal AGND | 0x0 |
| 10 | - | - | Reserved | 0x0 |
| 9 | r/w | MIC_OFFCOM_ON | Offset compensation control (for test purpose only) 0: off (normal operation) 1: on. | 0x0 |
| 8 | r/w | MIC_OFFCOM_SG | Offset compensation values (for test purpose only) 0: added to MICp. 1: added to MICn | 0x0 |
| 7:4 | r/w | MIC_GAIN | Microphone amplifier gain in steps of 2 dB 0000: 0 dB, 1111: +30 dB (Refer to table "Microphone Amplifier" for the voltage level at 0 dBm0) | 0x0 |
| 3 | r/w | MIC_MUTE | 0: normal operation 1: mute MIC inputs | 0x0 |
| 2 | r/w | MIC_PD | 1: Microphone amplifier power down 0: on. | 0x1 |

Table 347: CODEC_MIC_REG (0x00001280)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|----------|---|-------|
| 1:0 | r/w | MIC_MODE | Microphone input configuration M[1], M[0] 00: Normal Differential or single ended AC coupled. 01: MICp single ended. MICn disabled; High impedance 10: MICn single ended. MICp disabled; High impedance 11: MICp/n disabled; High impedance. And microphone amplifier is muted. Disabled pins are internally connected to analog ground. | 0x0 |

Table 348: CODEC_LSR_REG (0x00001282)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|------------|--|-------|
| 15:14 | - | - | Reserved | 0x0 |
| 13:11 | r/w | LSRATT | Loudspeaker amplifier analog gain in steps of 1 dB 000: 2 dB, 111: -12 dB (Refer to table "LSRp/LSRn output" for voltage level at 0 dBm0) | 0x0 |
| 10 | r/w | LSR_GND_PD | Obsolete, any value written to this bit will have no effect. | 0x1 |
| 9 | r/w | LSREN_SE | 1: Differential to single ended conversion enabled. 0: disabled | 0x0 |
| 8:6 | - | - | Reserved | 0x4 |
| 5 | r/w | LSRN_PD | 1: LSRn amplifier power down 0: on. | 0x1 |
| 4:3 | r/w | LSRN_MODE | 00: LSRn output has AGND reference voltage (If LSRN_PD = 1 then LSRn has high impedance) 01: Reserved 10: Differential mode DAn to LSRn amplifier 11: Single ended mode to LSRn amplifier (LSREN_SE must be set to 1) | 0x0 |
| 2 | r/w | LSRP_PD | 1: LSRp amplifier power down 0: on. | 0x1 |
| 1:0 | r/w | LSRP_MODE | 00: LSRp output has AGND reference voltage (If LSRP_PD = 1 then LSR+ has high impedance) 01: reserved 10: Differential mode DAp amplifier to LSRp amplifier 11: Single ended mode to LSRp amplifier (LSREN_SE must be set to 1) | 0x0 |

Table 349: CODEC_VREF_REG (0x00001284)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------|--|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7 | r/w | BIAS_PD | 1: Bias current for all CODEC amplifiers power down 0: on. | 0x1 |
| 6 | r/w | VREF_BG_PD | 1: Bandgap current mirror power down 0: on. Must be active if VREFp is used. | 0x0 |
| 5 | - | - | Reserved | 0x0 |
| 4 | r/w | VBUF_INIT | 0: 1 M Ω (+/-30%) series resistor to VBUF pin.(recommended setting) 1: 100 k Ω (+/-30%) series resistor to VBUF pin | 0x1 |
| 3:1 | - | - | Reserved | 0x0 |
| 0 | r/w | VREF_PD | 1: VREFp amplifier power down 0: on | 0x1 |

Table 350: CODEC_IN_OUT_REG (0x00001286)

| Bit | Mode | Symbol | Description | Reset |
|------|------|------------|--|-------|
| 15:0 | r/w | CODEC_DATA | Read: 16 bits CODEC input register Write: 16 bits CODEC output register if selected | 0x0 |

Table 351: CODEC_ADDA_REG (0x00001288)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------------|--|-------|
| 15 | r/w | SIDE_TONE_ON | Digital Side tone 0: off 1: on | 0x0 |
| 14:12 | r/w | SIDE_TONE | Digital Side tone between MIC to LSR 000: 0 dB 001: -6 dB 010: -12 dB 011: -18 dB (default) 100: -24 dB 101: -30 dB 110: -36 dB 111: -42 dB | 0x3 |
| 11 | - | - | Reserved | 0x0 |
| 10 | r/w | ADDA_COFF | 0: ADDA clocks on. Normal operation. 1: ADDA clocks off. This bit or all AD_PD, DA_PD, MIC_PD, LSRP_PD, LSRN_PD cause the digital part of the Audio CODEC to be switched off. For optimal power saving in the Audio CODEC analog part, all applicable CODEC_xxx_PD bits must also be set to '1' if a block is not used. | 0x0 |
| 9:7 | - | - | Reserved | 0x0 |
| 6 | r/w | REFINT_PD | 1: Internal reference opamps power down 0: on. Must be active if CODEC AD or DA is used. | 0x1 |
| 5:4 | - | - | Reserved | 0x0 |
| 3 | r/w | AD_DITH_OFF | 0: CODEC AD dithering on (normal), 1: off | 0x0 |
| 2 | r/w | DA_DITH_OFF | 0: CODEC DA dithering on (normal), 1: off | 0x0 |
| 1 | r/w | DA_PD | 1: CODEC DA power down, 0: on | 0x1 |
| 0 | r/w | AD_PD | 1: CODEC AD power down, 0: on | 0x1 |

Table 352: CODEC_OFFSET1_REG (0x0000128A)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|---|-------|
| 15:0 | r | COR1 | Offset error value (2's complement) (Refer to chapter "CODEC offset compensation") Each MIC_GAIN setting has a different offset error value. | 0x0 |

Table 353: CODEC_TEST_CTRL_REG (0x0000128C)

| Bit | Mode | Symbol | Description | Reset |
|-------|------|--------|-------------|-------|
| 15:14 | - | - | Reserved | 0x0 |

Table 353: CODEC_TEST_CTRL_REG (0x0000128C)

| Bit | Mode | Symbol | Description | Reset |
|------|------|----------|---|-------|
| 13 | r | COR_STAT | 0: CODEC_OFFSET_REG2 may be loaded with a new value. 1: The CODEC_OFFSET_REG2 is being copied to the internal offset compensation register, caused by a write operation to the CODEC_MIC_REG. During this synchronization process, the new MIC_GAIN and corresponding CODEC_OFFSET_REG2 value become effective at the same time. | 0x0 |
| 12 | r/w | COR_ON | 0: Digital offset compensation disabled. 1: Digital offset compensation enabled. Must be set to '1' before the CODEC_OFFSET_REG2 and CODEC_MIC_REG[MIC_GAIN] are initially written. | 0x0 |
| 11:0 | r/w | TCR | Test control bit must be 0 for normal operation | 0x0 |

Table 354: CODEC_OFFSET2_REG (0x0000128E)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|---|-------|
| 15:0 | r/w | COR2 | Offset compensation value 2 (2's complement). For an applied MIC_GAIN setting, the offset error value must be copied to this register. | 0x0 |

Table 355: CLK_CTRL_REG (0x000012E2)

| Bit | Mode | Symbol | Description | Reset | | | | | | | | | | | | | | | |
|--------|------------|--------------|--|---------|------|-------|------|---------|--------|------------|------|-----------|---|--------|------------|-------|---------|---|-----|
| 15:7 | - | - | Reserved | 0x0 | | | | | | | | | | | | | | | |
| 6 | r/w | HF_SEL | 0: PLL operates from 28-48 MHz 1: PLL operates from 48-56 MHz. This bit must be set at the same time as VCO_ON is set. | 0x0 | | | | | | | | | | | | | | | |
| 5 | r/w | LPF_SEL | 0: PLL slow loop filter; used if Fupd <= 1 MHz 1: PLL fast loop filter; used if Fupd > 1 MHz Examples: <table border="1" style="font-size: small; margin: 5px 0;"> <thead> <tr> <th>Fclk</th> <th>Fvco</th> <th>VD/XD</th> <th>Fupd</th> <th>LPF_SEL</th> </tr> </thead> <tbody> <tr> <td>10.368</td> <td>41.472 MHz</td> <td>12/3</td> <td>3.456 MHz</td> <td>1</td> </tr> <tr> <td>12.288</td> <td>36.824 MHz</td> <td>48/16</td> <td>768 kHz</td> <td>0</td> </tr> </tbody> </table> | Fclk | Fvco | VD/XD | Fupd | LPF_SEL | 10.368 | 41.472 MHz | 12/3 | 3.456 MHz | 1 | 12.288 | 36.824 MHz | 48/16 | 768 kHz | 0 | 0x0 |
| Fclk | Fvco | VD/XD | Fupd | LPF_SEL | | | | | | | | | | | | | | | |
| 10.368 | 41.472 MHz | 12/3 | 3.456 MHz | 1 | | | | | | | | | | | | | | | |
| 12.288 | 36.824 MHz | 48/16 | 768 kHz | 0 | | | | | | | | | | | | | | | |
| 4 | r/w | VCO_ON | 0: PLL VCO off 1: PLL VCO on, settling time to be timed with software (Refer to chapter PLL Clock switching procedure" | 0x0 | | | | | | | | | | | | | | | |
| 3 | r/w | PLL_BYPASSN | 0: PLL in bypass mode, Xtal is main system clock 1: PLL clock is main system clock | 0x0 | | | | | | | | | | | | | | | |
| 2 | - | - | Reserved | 0x0 | | | | | | | | | | | | | | | |
| 1 | r/w | CP_ON | 0: PLL VCO Charge pump off 1: PLL VCO Charge pump on (Normal operation) | 0x0 | | | | | | | | | | | | | | | |
| 0 | r/w | TESTMODE_SEL | If PLL test mode enabled: 0: VCO test mode 1: PLL divider test mode | 0x0 | | | | | | | | | | | | | | | |

Table 356: PLL_DIV_REG (0x000012E4)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|-------------|-------|
| 15 | - | - | Reserved | 0x0 |

Table 356: PLL_DIV_REG (0x000012E4)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------|---|-------|
| 14 | r/w | DIV3 | PLL VCO divider DIV3 DIV4 VD[3-0] Divide by 0 0 0000 undefined 1 0 xxxx 8 1 1 xxxx 12 0 0 0001 2x8 = 16 0 0 0010 3x8 = 24 0 0 1111 16x8 = 128 0 1 0001 2x12 = 24 0 1 0010 3x12 = 36 0 1 1011 12x12 = 144 0 1 1111 16x12 = 192 | 0x0 |
| 13 | r/w | DIV4 | 0 | 0x0 |
| 12:9 | r/w | VD | 0 | 0x0 |
| 8 | r/w | DIV1 | PLL xtal divider DIV1 DIV2 XD[6-0] Divide by 0 0 000 0000 undefined 1 0 xxx xxxx 4 1 1 xxx xxxx 3 0 x 000 0111 7 0 x 000 1010 10 0 x 000 1011 11 0 x 000 1101 13 0 x 000 1110 14 0 x 100 0001 65 0 x 111 1111 127 | 0x0 |
| 7 | r/w | DIV2 | 0 | 0x0 |
| 6:0 | r/w | XD | 0000000 | 0x0 |

Table 357: PER_DIV_REG (0x000012E6)

| Bit | Mode | Symbol | Description | Reset |
|------|------|---------|--|-------|
| 15:6 | - | - | Reserved | 0x0 |
| 5:0 | r/w | PER_DIV | Peripheral clock divider factor. This value determines the base frequency for PER_CLK if PLL is on. The minimum value of this register is 2. | 0x4 |

Table 358: CODEC_DIV_REG (0x000012E8)

| Bit | Mode | Symbol | Description | Reset |
|------|------|--------------|--|-------|
| 15:8 | - | - | Reserved | 0x0 |
| 7 | - | - | Reserved | 0x0 |
| 6 | r/w | CLOCK_SOURCE | 0: CODEC_DIV_REG clock input is PLL/XTAL 1: CODEC_DIV_REG clock input is PER_CLK clock | 0x0 |
| 5:0 | r/w | CODEC_DIV | Codec clock divider factor. This value determines the CODEC_CLK base frequency of 1.152 MHz (8 kHz) resp 2.304 MHz (16 kHz) . The input frequency of this divider depends on the value of CLOCK_SOURCE and the PLL/XTAL and PER_CLK. | 0x20 |

Table 359: VERSION_REG0 (0x000012FA)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|--|-------|
| 7:0 | r | VDEV1 | First character of device type in ASCII. "439" | 0x34 |

Table 360: VERSION_REG1 (0x000012FB)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|---|-------|
| 7:0 | r | VDEV2 | Second character of device type in ASCII. "439" | 0x33 |

Table 361: VERSION_REG2 (0x000012FC)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|--|-------|
| 7:0 | r | VDEV3 | Third character of device type in ASCII. "439" | 0x39 |

Table 362: VERSION_REG3 (0x000012FD)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|-------------|-------|
| 7:0 | r | VCONF | Reserved | 0x0 |

Table 363: VERSION_REG4 (0x000012FE)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|---|-------|
| 7:4 | r | VREV | Chip version, corresponds with type number 0x1: 'A', 0x2: 'B', etc. | 0x1 |
| 3:0 | r | VMETAL | Metal change revision 0x0: No metal change in mask set of "chip version" 0x1: first metal change. 0x2: Second metal change, etc. | 0x0 |

Table 364: VERSION_REG5 (0x000012FF)

| Bit | Mode | Symbol | Description | Reset |
|-----|------|--------|-------------|-------|
| 7:0 | r | VMODE | Reserved | 0x0 |

29 Specifications

All MIN/MAX specification limits are guaranteed by design, production testing and/or statistical characterization. Typical values are based on characterization results at default measurement conditions and are informative only.

Default measurement conditions (unless otherwise specified): $V_{BAT}(VBAT3V) = 3.0\text{ V}$ (buck mode), $T_A = 25\text{ }^\circ\text{C}$. All radio measurements are performed with standard RF measurement equipment providing a source/load impedance of $50\ \Omega$.

The specifications in the following tables are valid for the reference circuit shown in Figure 78 (Buck mode).

Note: The DA14582 does not support DC-DC converter Boost mode.

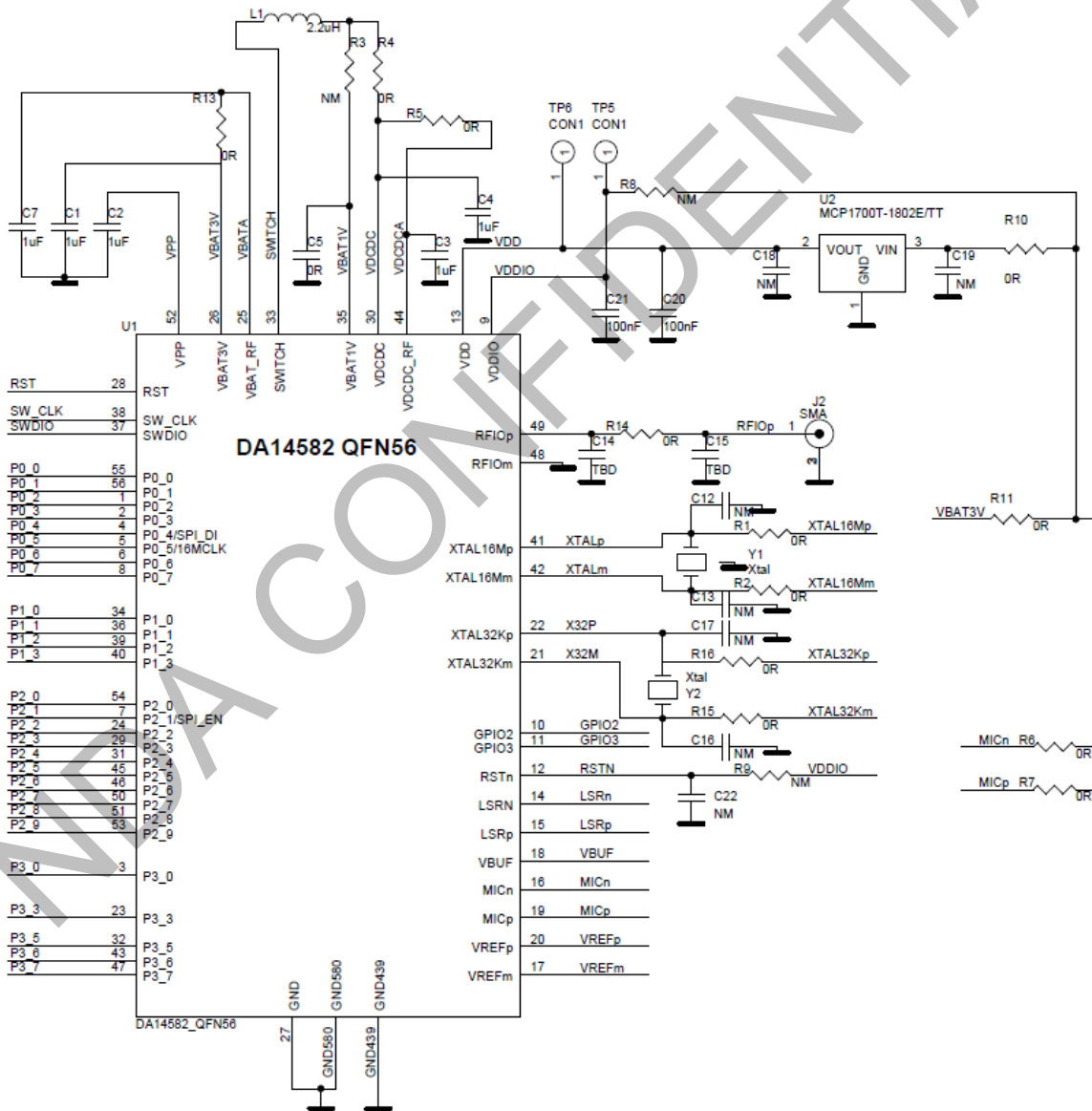


Figure 78: Lithium Coin Cell Powered System Diagram (Buck Mode)

Table 365: Absolute Maximum Ratings

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|----------------------------------|--|--|------|-----|------------------------------------|------|
| V _{PIN(LIM)} (default) | limiting voltage on a pin | Voltage between pin and GND (Note 25) | -0.1 | | min{3.6, V _{BAT_RF} +0.2} | V |
| T _{STG} | storage temperature | | -50 | | 150 | °C |
| t _{R(SUP)} | supply rise time | Power supply rise time | | | 100 | ms |
| V _{BAT(LIM)} (VBAT1V) | limiting battery supply voltage | Supply voltage on VBAT1V in a boost converter application (VBAT3V is second output of boost-converter in this case) (Note 25) | -0.1 | | 3.6 | V |
| V _{BAT(LIM)} (VBAT3V) | limiting battery supply voltage | Supply voltage on VBAT3V and VBAT_RF in a buck-converter application, pin VBAT1V is connected to ground (Note 25) | -0.1 | | 3.6 | V |
| V _{PIN(LIM)} (1V2) | limiting voltage on a pin | XTAL32Km, XTAL16Mp, XTAL16Mm (Note 25) | -0.2 | | min(1.2, V _{BAT_RF} +0.2) | V |
| V _{PIN(LIM)} (VDCDC_RF) | limiting voltage on the VDCDC_RF pin | Supply voltage on VDCDC_RF (Note 25) | -0.2 | | min(3.3, V _{BAT_RF} +0.2) | V |
| V _{PIN(LIM)} (XTAL32Kp) | limiting voltage on a pin | XTAL32Kp | -0.2 | | min(1.5, V _{BAT_RF} +0.2) | V |
| V _{ESD(HBM)} (WLCSP34) | electrostatic discharge voltage (Human Body Model) | | | | 2000 | V |
| V _{ESD(HBM)} (QFN40) | electrostatic discharge voltage (Human Body Model) | | | | 4000 | V |
| V _{ESD(HBM)} (QFN48) | electrostatic discharge voltage (Human Body Model) | | | | 4000 | V |
| V _{ESD(MM)} (WLCSP34) | electrostatic discharge voltage (Machine Model) | | | | 200 | V |
| V _{ESD(MM)} (QFN40) | electrostatic discharge voltage (Machine Model) | | | | 200 | V |
| V _{ESD(MM)} (QFN48) | electrostatic discharge voltage (Machine Model) | | | | 200 | V |
| V _{ESD(CDM)} (WLCSP34) | electrostatic discharge voltage (Charged Device Model) | | | | 500 | V |
| V _{ESD(CDM)} (QFN40) | electrostatic discharge voltage (Charged Device Model) | | | | 1000 | V |
| V _{ESD(CDM)} (QFN48) | electrostatic discharge voltage (Charged Device Model) | | | | 1000 | V |

Note 25: The device should not be exposed for prolonged periods of time to voltages between the Recommended Operating Conditions and the Absolute Maximum Ratings range.

Table 366: Recommended Operating Conditions

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|-----------------------------|------------------------|---|-------------------|-----|--------------------------------------|------|
| V _{PP} | programming voltage | Supply voltage on pin VPP during OTP programming; T _J ≤ 50 °C | 6.6 | 6.7 | 6.8 | V |
| V _{BAT} (VBAT1V) | battery supply voltage | Supply voltage on VBAT1V in a boost converter application (VBAT3V is second output of boost-converter in this case) | 0.9 | | 3.3 | V |
| V _{BAT} (VBAT3V) | battery supply voltage | Supply voltage on VBAT3V and VBAT_RF in a buck-converter application, pin VBAT1V is connected to ground | 2.35 (Note 26) | | 3.3 | V |
| V _{PIN} (default) | voltage on a pin | Voltage between pin and GND | 0 | | min(3.3, V _{BAT_RF+} - 0.2) | V |
| V _{PIN} (1V2) | voltage on a pin | XTAL32Km, XTAL16Mp, XTAL16Mm | 0 | | 1.2 | V |
| V _{PIN} (VDCDC_RF) | voltage on a pin | Supply voltage on VDCDC_RF | 0 | | 3.3 | V |
| T _A | ambient temperature | | -20 | | 60 | °C |

Note 26: Cold boot should not be performed if voltage is less than 2.5 V because of possible corruption during OTP data mirroring. Trim values programmed in the OTP as well as the application image, should be copied into RAM while VBAT3V ≥ 2.5 V.

Table 367: DC Characteristics

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|---------------------------------------|------------------------|---|-----|------|-----|------|
| I _{BAT} (DP_SLP)_BOOST_1kB | battery supply current | Boost configuration in deep-sleep with 1 kB retention RAM active, running from RC32K oscillator at lowest frequency | | 0.48 | | μA |
| I _{BAT} (DP_SLP)_BOOST_2kB | battery supply current | Boost configuration in deep-sleep with 2 kB retention RAM active, running from XTAL32K oscillator | | 0.55 | | μA |
| I _{BAT} (DP_SLP)_BOOST_8kB | battery supply current | Typical boost-application in deep-sleep with 8 kB retention RAM active, running from XTAL32K oscillator | | 0.7 | 2 | μA |
| I _{BAT} (EXT_SLP)_BOOST_43KB | battery supply current | Typical boost-application in extended-sleep mode with 42 kB (SysRAM) and 1 kB (RetRAM) retained | | 1.37 | | μA |

Table 367: DC Characteristics

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|----------------------------------|------------------------|--|-----|------|-----|---------|
| $I_{BAT}(EXT_SLP)_BOOST_50kB$ | battery supply current | Typical boost-application in extended-sleep mode with 42 kB (SysRAM) and 8 kB (RetRAM) retained | | 1.5 | 3 | μA |
| $I_{BAT}(DP_SLP)_BUCK_1kB$ | battery supply current | Buck configuration in deep-sleep with 1 kB retention RAM active, running from RC32K oscillator at lowest frequency | | 0.4 | | μA |
| $I_{BAT}(DP_SLP)_BUCK_2kB$ | battery supply current | Buck configuration in deep-sleep with 2 kB retention RAM active, running from XTAL32K oscillator | | 0.45 | | μA |
| $I_{BAT}(DP_SLP)_BUCK_8kB$ | battery supply current | Typical buck-application in deep-sleep with 8 kB retention RAM active, running from XTAL32K oscillator | | 0.6 | 2 | μA |
| $I_{BAT}(EXT_SLP)_BUCK_43kB$ | battery supply current | Typical buck-application in extended-sleep mode with 42 kB (SysRAM) and 1 kB (RetRAM) retained | | 1.2 | | μA |
| $I_{BAT}(EXT_SLP)_BUCK_50kB$ | battery supply current | Typical buck-application in extended-sleep mode with 42 kB (SysRAM) and 8 kB (RetRAM) retained | | 1.4 | 3 | μA |
| $I_{BAT}(ACT_RX)_BOOST$ | battery supply current | Typical application with boost converter and receiver active | | 13.4 | 16 | mA |
| $I_{BAT}(ACT_TX)_BOOST$ | battery supply current | Typical application with boost converter and transmitter active | | 12.4 | 15 | mA |
| $I_{BAT}(ACT_RX)_BUCK$ | battery supply current | Typical application with buck converter and receiver active | | 5.1 | 6 | mA |
| $I_{BAT}(ACT_TX)_BUCK$ | battery supply current | Typical application with buck converter and transmitter active | | 4.8 | 6 | mA |

Table 368: Timing Characteristics

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|------------------|--------------|--|-----|------------------|-----|------|
| $t_{STA}(BOOST)$ | startup time | Boost-mode; time from deep-sleep to software start. Typical application, running from retention RAM on 16 MHz RC oscillator | | 1.2 (Note 27) | | ms |

Table 368: Timing Characteristics

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|-----------------|--------------|---|-----|----------------|-----|------|
| $t_{STA(BUCK)}$ | startup time | Buck-mode; time from deep-sleep to software start. Typical application, running from retention RAM on 16 MHz RC oscillator | | 1 (Note 27) | | ms |

Note 27: Worst-case value under Normal Operating Conditions.

Table 369: 16 MHz Crystal Oscillator: Recommended Operating Conditions

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|---------------------------|------------------------------|---|-----|-----|------|----------|
| $f_{XTAL(16M)}$ | crystal oscillator frequency | | | 16 | | MHz |
| ESR(16M) | equivalent series resistance | | | | 100 | Ω |
| $C_L(16M)$ | load capacitance | No external capacitors are required | 10 | | 12 | pF |
| $C_0(16M)$ | shunt capacitance | | | | 5 | pF |
| $\Delta f_{XTAL(16M)}$ | crystal frequency tolerance | After optional trimming; including aging and temperature drift (Note 28) | -20 | | 20 | ppm |
| $\Delta f_{XTAL(16M)UNT}$ | crystal frequency tolerance | Untrimmed; including aging and temperature drift (Note 29) | -40 | | 40 | ppm |
| $P_{DRV(MAX)}(16M)$ | maximum drive power | (Note 30) | 100 | | | μW |
| $V_{CLK(EXT)}(16M)$ | external clock voltage | Only in case of an external reference clock on XTAL16Mp (XTAL16Mm floating or connected to mid-level 0.6 V) | 1 | 1.2 | | V |
| $\varphi_N(EXTERNAL)16M$ | phase noise | $f_C = 50$ kHz in case of an external reference clock | | | -130 | dBc/Hz |

Note 28: Using the internal varicaps a wide range of crystals can be trimmed to the required tolerance.

Note 29: Maximum allowed frequency tolerance for compensation by the internal varicap trimming mechanism.

Note 30: Select a crystal which can handle a drive-level equal or more than this specification

Table 370: 16 MHz Crystal Oscillator: Timing Characteristics

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|----------------------|---------------------------------|------------|-----|-----|-----|------|
| $t_{STA(XTAL)}(16M)$ | crystal oscillator startup time | | 0.5 | 2 | 3 | ms |

Table 371: 32 kHz Crystal Oscillator: Recommended Operating Conditions

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|----------------------------|---|---|------|--------|-----|------|
| V _{CLK(EXT)(32K)} | external clock voltage | peak-peak voltage of external clock at XTAL32Kp, pin XTAL32Km floating. note: XTAL32Kp is internally AC coupled | 0.1 | 0.2 | 1.5 | V |
| f _{XTAL(32k)} | crystal oscillator frequency | frequency range for an external clock (for a crystal, use either 32.000 kHz or 32.768 kHz) | 10 | 32.768 | 100 | kHz |
| ESR(32k) | equivalent series resistance | | | | 100 | kΩ |
| C _{L(32k)} | load capacitance | no external capacitors are required for a 6 pF or 7 pF crystal | 6 | 7 | 9 | pF |
| C _{0(32k)} | shunt capacitance | | | 1 | 2 | pF |
| Δf _{XTAL(32k)} | crystal frequency tolerance (including aging) | Timing accuracy is dominated by crystal accuracy. A much smaller value is preferred | -250 | | 250 | ppm |
| P _{DRV(MAX)(32k)} | maximum drive power | (Note 31) | 0.1 | | | μW |

Note 31: Select a crystal that can handle a drive-level of at least this specification.

Table 372: 32 kHz Crystal Oscillator: Timing Characteristics

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|-----------------------------|---------------------------------|--|-----|-----|-----|------|
| t _{STA(XTAL)(32k)} | crystal oscillator startup time | Typical application, time until 1000 clocks are detected (Note 32) | | 0.4 | | s |

Note 32: This parameter is very much dependent on crystal parameters

Table 373: DC-DC Converter: Recommended Operating Conditions

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|--------------------------|----------------------------|---|-----|-----|-----|------|
| L | effective inductance | | 1.5 | 2.2 | 3 | μH |
| C _{OUT(VDCDC)} | effective load capacitance | VDCDC and VDDCRF combined (Note 33) | 0.5 | 1 | 10 | μF |
| C _{OUT(VBAT3V)} | effective load capacitance | VBATRF and VBAT3V combined are the second output of the boost-converter (Note 33) | 0.5 | 1 | 10 | μF |

Note 33: A low value will result in lowest power consumption, keep this value at 1 uF or 2 uF.

Table 374: DC-DC Converter: DC Characteristics

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|----------------------|----------------|------------------|-----|------|-----|------|
| V _{O(BUCK)} | output voltage | default settings | | 1.41 | | V |

Table 374: DC-DC Converter: DC Characteristics

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|---|-------------------------------|---|------|-------|-----|------|
| V _O (BOOST) | output voltage | default settings, VDCDC | | 1.41 | | V |
| $\eta_{\text{CONV_MAX}}(\text{BUCK})$ | maximum conversion efficiency | | | 86 | | % |
| $\eta_{\text{CONV_MAX}}(\text{BOOST})$ | maximum conversion efficiency | | | 80 | | % |
| $\Delta V_{\text{O}}/\Delta V_{\text{I}}(\text{BUCK})$ | line regulation | 2.35 V ≤ VBAT3V ≤ 3.3 V | -2 | 0.7 | 2 | %/V |
| $\Delta V_{\text{O}}/\Delta V_{\text{I}}(\text{BOOST})$ | line regulation | 0.9 V ≤ VBAT1V ≤ 1.2 V (Note 34) | -2 | 1 | 4 | %/V |
| $\Delta V_{\text{O}}/\Delta I_{\text{L}}(\text{BUCK})$ | load regulation | VBAT3V = 2.5 V | -0.2 | -0.02 | 0.2 | %/mA |
| $\Delta V_{\text{O}}/\Delta I_{\text{L}}(\text{BOOST})$ | load regulation | VBAT1V = 1.2 V | -0.2 | -0.07 | 0.2 | %/mA |
| V _{RPL} (BUCK) | ripple voltage | buck mode; RMS ripple voltage | | 5 | | mV |
| V _{RPL} (BOOST) | ripple voltage | VBAT1V ≤ 1.2 V, boost mode; RMS ripple voltage (Note 34) | | 8 | | mV |

Note 34: When VBAT1V > VDCDC_nominal, VDCDC will follow VBAT1V.

Table 375: Digital Input/Output: DC Characteristics

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|--------------------------|---------------------------|--|------|-----|------|------|
| V _{IH} | HIGH level input voltage | | 0.84 | | | V |
| V _{IL} | LOW level input voltage | | | | 0.36 | V |
| V _{IH} (RST) | HIGH level input voltage | RST pin | 0.84 | | | V |
| V _{IL} (RST) | LOW level input voltage | RST pin | | | 0.36 | V |
| V _{OH} (VBAT1V) | HIGH level output voltage | I _{out} = -250 μA, VBAT3V = 2.35 V, VBAT1V = 0.9 V | 0.72 | | | V |
| V _{OH} (VBAT3V) | HIGH level output voltage | I _{out} = -4.8 mA, VBAT3V = 2.35 V, VBAT1V = 0 V (Note 35) | 1.88 | | | V |
| V _{OL} (VBAT1V) | LOW level output voltage | I _{out} = 250 μA, VBAT3V = 2.35 V, VBAT1V = 0.9 V | | | 0.18 | V |
| V _{OL} (VBAT3V) | LOW level output voltage | I _{out} = 4.8 mA, VBAT3V = 2.35 V, VBAT1V = 0 V (Note 36) | | | 0.47 | V |
| I _{IH} | HIGH level input current | V _{in} = VBAT3V = 2.5 V | -1 | | 1 | μA |
| I _{IL} | LOW level input current | V _{in} = VSS = 0 V | -1 | | 1 | μA |
| I _{IH} (PD) | HIGH level input current | V _{in} = VBAT3V = 2.5 V | 50 | | 150 | μA |
| I _{IL} (PU) | LOW level input current | V _{in} = VSS = 0 V | -150 | | -50 | μA |
| I _{IH} (RST) | HIGH level input current | RST pin, V(RST) = 1.2 V | 25 | | 75 | μA |

Note 35: In Boost mode the output source current is limited to I_{out} = -250 μA.

Note 36: In Boost mode the output sink current is limited to I_{out} = 250 μA.

Table 376: General Purpose ADC: Recommended Operating Conditions

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|------------------------|-----------------------------|------------|-----|-----|-----|------|
| N _{BIT} (ADC) | number of bits (resolution) | | | 10 | | bit |

Table 377: General Purpose ADC: DC Characteristics

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|---------------------|-----------------------------------|--|------|-------|------|------|
| V _{I(ZS)} | zero-scale input voltage | single-ended, calibrated at zero input | -2.5 | 0 | 2.5 | mV |
| V _{I(FS)} | full-scale input voltage | single-ended, calibrated at zero input | 1150 | 1180 | 1250 | mV |
| V _{I(FSN)} | negative full-scale input voltage | differential, calibrated at zero input | | -1180 | | mV |
| V _{I(FSP)} | positive full-scale input voltage | differential, calibrated at zero input | | 1180 | | mV |
| INL | integral non-linearity | | -2 | | 2 | LSB |
| DNL | differential non-linearity | | -2 | | 2 | LSB |

Table 378: General Purpose ADC: Timing Characteristics

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|-------------------------|-----------------|--|-----|------|-----|------|
| t _{CONV} (ADC) | conversion time | Excluding initial settling time of the LDO and the 3x-attenuation (if used): LDO settling time is 20 μs (max), 3x-attenuation settling time = 1 μs (max) Using internal ADC-clock (~200 MHz) | | 0.25 | 0.4 | μs |

Table 379: Radio: DC Characteristics

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|-------------------------|------------------------|---|-----|-----|-----|------|
| I _{BAT} (RF)RX | battery supply current | receive mode; radio receiver and synthesizer active; DCDC converter assumed ideal; T _A = 25 °C (Note 37) | | 3.7 | 4.3 | mA |
| I _{BAT} (RF)TX | battery supply current | transmit mode; radio transmitter and synthesizer active; DCDC converter assumed ideal; T _A = 25 °C (Note 37) | | 3.4 | 4 | mA |

Note 37: The DCDC-converter efficiency is assumed to be 100 % to enable benchmarking of the radio currents at battery supply domain (VBAT3V = 3 V).

Table 380: Radio: AC Characteristics

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|--------------------------|---|---|-----|-------|-----|------|
| P _{SENS(CLEAN)} | sensitivity level | DC-DC converter enabled; Dirty Transmitter disabled; T _A = 25 °C; PER = 30.8 % (Note 38) | | -93 | | dBm |
| P _{SENS} | sensitivity level | Normal Operating Conditions; DC-DC converter enabled; T _A = 25 °C; PER = 30.8 % (Note 38) | | -92.5 | | dBm |
| P _{SENS(EOC)} | sensitivity level | Extreme Operating Conditions; DC-DC converter enabled; PER = 30.8 %; -40 °C ≤ T _A ≤ +85 °C (Note 39) | | | -87 | dBm |
| P _{I(max)} | input power level | DC-DC converter disabled; T _A = 25 °C; PER ≤ 30.8 % (Note 38) | 10 | | | dBm |
| P _{INT(IMD)} | intermodulation distortion interferer power level | worst case interferer level @ f ₁ , f ₂ with 2f ₁ -f ₂ = f ₀ , f ₁ -f ₂ = n MHz and n = 3,4,5; P _{WANTED} = -64 dBm @ f ₀ ; PER = 30.8 %; T _A = 25 °C (Note 40) | -35 | -31 | | dBm |
| CIR(0) | carrier to interferer ratio | n = 0; interferer @ f ₁ = f ₀ + n*1 MHz; T _A = 25 °C (Note 41) | | 7 | 21 | dB |
| CIR(1) | carrier to interferer ratio | n = ±1; interferer @ f ₁ = f ₀ + n*1 MHz; T _A = 25 °C (Note 41) | | -3 | 15 | dB |
| CIR(P2) | carrier to interferer ratio | n = +2 (image frequency); interferer @ f ₁ = f ₀ + n*1 MHz; T _A = 25 °C (Note 41) | | -20 | -9 | dB |
| CIR(M2) | carrier to interferer ratio | n = -2; interferer @ f ₁ = f ₀ + n*1 MHz; T _A = 25 °C (Note 41) | | -30 | -17 | dB |
| CIR(P3) | carrier to interferer ratio | n = +3 (image frequency + 1 MHz); interferer @ f ₁ = f ₀ + n*1 MHz; T _A = 25 °C (Note 41) | | -30 | -15 | dB |
| CIR(M3) | carrier to interferer ratio | n = -3; interferer @ f ₁ = f ₀ + n*1 MHz; T _A = 25 °C (Note 41) | | -35 | -27 | dB |

Table 380: Radio: AC Characteristics

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|------------------------------|------------------------------|--|------|-------|-------|--------|
| CIR(4) | carrier to interferer ratio | $ n \geq 4$ (any other BLE channel); interferer @ $f_1 = f_0 + n \cdot 1$ MHz; $T_A = 25$ °C (Note 41) | | -37 | -27 | dB |
| P _{BL(I)} | blocker power level | $30 \text{ MHz} \leq f_{BL} \leq 2000$ MHz; P _{WANTED} = -67 dBm; $T_A = 25$ °C (Note 42) | -5 | | | dBm |
| P _{BL(II)} | blocker power level | $2003 \text{ MHz} \leq f_{BL} \leq 2399$ MHz; P _{WANTED} = -67 dBm; $T_A = 25$ °C (Note 42) | -15 | | | dBm |
| P _{BL(III)} | blocker power level | $2484 \text{ MHz} \leq f_{BL} \leq 2997$ MHz; P _{WANTED} = -67 dBm; $T_A = 25$ °C (Note 42) | -15 | | | dBm |
| P _{BL(IV)} | blocker power level | $3000 \text{ MHz} \leq f_{BL} \leq 12.75$ GHz; P _{WANTED} = -67 dBm; $T_A = 25$ °C (Note 42) | -5 | | | dBm |
| P _{RSSI(min)} | RSSI power level | absolute power level for RXRSSI[7:0] = 0; $T_A = 25$ °C (Note 43) | -115 | -112 | -109 | dBm |
| P _{RSSI(max)} | RSSI power level | upper limit of monotonous range; $T_A = 25$ °C | -26 | -19 | | dBm |
| L _{ACC(RSSI)BO OST} | level accuracy | tolerance of 5 % to 95 % confidence interval of P _{RF} : when RXRSSI[7:0] = X, $50 < X < 175$; burst mode 1500 packets; $T_A = 25$ °C; DC-DC converter in BOOST mode | | 0 | 3 | dB |
| L _{ACC(RSSI)BU CK} | level accuracy | tolerance of 5 % to 95 % confidence interval of P _{RF} : when RXRSSI[7:0] = X, $50 < X < 175$; burst mode 1500 packets; $T_A = 25$ °C; DC-DC converter in BUCK mode | | 0 | 2 | dB |
| L _{RES(RSSI)} | level resolution | gradient of monotonous range for RXRSSI[7:0] = X, $50 < X < 175$; burst mode 1500 packets; $T_A = 25$ °C | 0.46 | 0.474 | 0.485 | dB/LSB |
| ACP(2M) | adjacent channel power level | $f_{\text{OFFSET}} = 2$ MHz; $T_A = 25$ °C (Note 44) | | -53 | -50 | dBm |

Table 380: Radio: AC Characteristics

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|-------------------|---|---|-----|-----|-----|------|
| ACP(2M)(EOC) | adjacent channel power level | $f_{\text{OFFSET}} = 2 \text{ MHz}$; $-40 \text{ }^\circ\text{C} \leq T_A \leq +85 \text{ }^\circ\text{C}$ (Note 44) | | -53 | -47 | dBm |
| ACP(3M) | adjacent channel power level | $f_{\text{OFFSET}} \geq 3 \text{ MHz}$; $T_A = 25 \text{ }^\circ\text{C}$ (Note 44) | | -57 | -55 | dBm |
| ACP(3M)(EOC) | adjacent channel power level | $f_{\text{OFFSET}} \geq 3 \text{ MHz}$; $-40 \text{ }^\circ\text{C} \leq T_A \leq +85 \text{ }^\circ\text{C}$ (Note 44) | | -57 | -47 | dBm |
| P_O | output power level | $V_{\text{DD}} = 3 \text{ V}$; maximum gain; $T_A = 25 \text{ }^\circ\text{C}$ | -2 | -1 | 0 | dBm |
| $P_O(\text{HD}2)$ | output power level (second harmonic) | $V_{\text{DD}} = 3 \text{ V}$; maximum gain; $T_A = 25 \text{ }^\circ\text{C}$ | | -54 | -40 | dBm |
| $P_O(\text{HD}3)$ | output power level (third harmonic) | $V_{\text{DD}} = 3 \text{ V}$; maximum gain; $T_A = 25 \text{ }^\circ\text{C}$ | | -56 | -40 | dBm |
| $P_O(\text{HD}4)$ | output power level (fourth harmonic) | $V_{\text{DD}} = 3 \text{ V}$; maximum gain; $T_A = 25 \text{ }^\circ\text{C}$ | | -70 | -40 | dBm |
| $P_O(\text{HD}5)$ | output power level (fifth harmonic) | $V_{\text{DD}} = 3 \text{ V}$; maximum gain; $T_A = 25 \text{ }^\circ\text{C}$ | | -70 | -40 | dBm |
| $P_O(\text{NFM})$ | output power level in 'Near Field Mode' | $V_{\text{DD}} = 3 \text{ V}$; maximum gain; $T_A = 25 \text{ }^\circ\text{C}$ (Note 45) | -25 | -20 | -15 | dBm |

Note 38: Measured according to Bluetooth® Low Energy Test Specification RF-PHY.TS/4.0.1, section 6.4.1.

Note 39: Measured according to Bluetooth® Low Energy Test Specification RF-PHY.TS/4.0.1, section 6.4.2.

Note 40: Measured according to Bluetooth® Core Technical Specification document, version 4.2, volume 6, section 4.4. Published value is for $n = \text{IXIT} = 4$. $\text{IXIT} = 5$ gives the same results, $\text{IXIT} = 3$ gives results that are 5 dB lower.

Note 41: Measured according to Bluetooth® Core Technical Specification document, version 4.2, volume 6, section 4.2.

Note 42: Measured according to Bluetooth® Core Technical Specification document, version 4.2, volume 6, section 4.3. Due to limitations of the measurement equipment, levels of -5 dBm should be interpreted as $> -5 \text{ dBm}$.

Note 43: $\text{PRF} = \text{PRSSI}(\text{min}) + \text{LRES}(\text{RSSI}) \times \text{RXRSSI}[7:0] \pm \text{LACC}(\text{RSSI})$. Thanks to constant gain biasing of RF part in the receiver, the RSSI can be used to estimate absolute power levels, rather than mere level changes. Even across the full temperature range the variation is limited.

Note 44: Measured according to Bluetooth® Low Energy Test Specification RF-PHY.TS/4.0.1, section 6.2.3.

Note 45: To activate the "Near Field Mode", program address 0x50002418 with the value 0x0030.

Table 381: Stable Low Frequency RCX Oscillator: Timing Characteristics

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|---|--------------------------------|--|------|-----|------|---------------------------|
| $f_{\text{RC}}(\text{RCX})$ | RCX oscillator frequency | default setting, buck mode only | 5 | 10 | 40 | kHz |
| $\Delta f_{\text{RC}}(\text{RCX})$ | RCX oscillator frequency drift | buck mode only (Note 46) | -500 | | 500 | ppm |
| $\Delta T_A / \Delta t(\text{RCX})100\text{ms}$ | ambient temperature gradient | buck mode only; connection interval 100 ms | | | 0.66 | $^\circ\text{C}/\text{s}$ |
| $\Delta T_A / \Delta t(\text{RCX})4\text{s}$ | ambient temperature gradient | buck mode only; connection interval 4 s | | | 0.33 | $^\circ\text{C}/\text{s}$ |

Note 46: Maximum recommended connection interval (including slave latency) for the RCX usage is 2 s.

30 CODEC Specifications

All MIN/MAX specification limits are guaranteed by design, production testing and/or statistical characterization. Typical values are based on characterization results at default measurement conditions and are informative only.

Default measurement conditions (unless otherwise specified): $V_{DD} = 1.8\text{ V}$, $T_A = 25\text{ }^\circ\text{C}$. AVD = VDD (pin 13), AVS = GND plane (pin 57).

Table 382: Absolute Maximum Ratings (Note 47)

| Parameter | Description | Conditions | Min | Max | Unit |
|-----------|---|------------|-----|-----|------|
| Vddio_max | Max Digital I/O supply voltage (VDDIO-VSS) | | | 3.6 | V |
| Vdd_max | Max Core supply voltage (VDD-VSS / AVD-AVS) | | | 2.0 | V |

Note 47: The device should not be exposed for prolonged periods of time to voltages between the Recommended Operating Conditions and the Absolute Maximum Ratings range.

Table 383: Recommended Operating Conditions

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|-----------|---|------------|-----------|-----|------------|------|
| Vddio | Digital I/O supply voltage VDDIO-VSS | | 1.75 | 2.5 | 3.45 | V |
| Vdd | Supply voltages VDD-VSS, AVD-AVS | (Note 48) | 1.75 | 1.8 | 1.98 | V |
| Vdig | Voltage on digital pins. | | | | VDDIO +0.3 | V |
| Vana | Voltage on analog pins | | | | 2.0 | V |
| Vpin_neg | Minimum voltage on any pin | | VSSIO-0.3 | | | V |
| Vprot_mic | Voltage on pins MIC (with protection) | | 0.4 | | 1.4 | V |
| Iprot_mic | Current through protection diode MICp to AGND | | | | 2.4 | mA |

Note 48: Full operating mode; the differences between AVD and VDD may never be more than 300 mV; during a short period of time e.g. during power up more than 300 mV difference is allowed. Analog performance is only guaranteed from 1.75 V to 1.98 V.

30.1 CODEC DC CHARACTERISTICS

VDD = AVD = 1.8 V. All signals are related to VSS = GND plane (pin 57). $T_A = -20\text{ }^\circ\text{C} - +60\text{ }^\circ\text{C}$.

Table 384: Supply Currents

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|---------------|---------------------------------|--|-----|------|-----|---------------|
| Ivddio_static | Static supply current at VDDIO. | RSTn = 0, CLK = 0, VDDIO = 3.45 V. | | 20 | | μA |
| Ivdd_static | Static supply current at VDD | RSTn = 0, CLK = 0. | | 150 | | μA |
| Iavd_static | Static supply current at AVD | Any analog block which needs the band-gap reference voltage or current, causes this static current to flow. With BANDGAP_REG = 0x0208 and CODEC_VREF_REG = 0x00D1 and all PD bits set to '1' reduces this current to 15 μA . | | 100 | | μA |
| Ivdd_voice | Digital supply current at VDD | SPI 16 bits Rx Tx, CODEC on, CLK = 16 MHz, PLL off. (Including Ivdd_static) | | 0.75 | | mA |
| Ivdd_pll | Digital PLL supply current | CLK = 16 MHz Fpll = 46.06 MHz | | 0.15 | | mA |
| Iavd_pll | Analog PLL supply current | | | 0.6 | | mA |

Table 384: Supply Currents

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|------------|------------------------------|--|-----|-----|-----|------|
| lavd_voice | Analog supply current at AVD | Reference amplifiers on, microphone amp on, LSRp/n on (no load), CODEC on. | | 4.5 | | mA |

Note 49: Total maximum chip current = $I_{vdd_voice} + I_{avd_pll*on} + I_{vdd_pll*on} + I_{avd_pa} + I_{vdd_voice} + I_{vddio_static} - I_{vdd_static}$ (is already included in I_{vdd_voice}).

Additional currents due to external components **must be added** to the total chip current: E.g current through loudspeakers and microphone, current through (external) pull-up/pull-down resistors.

Table 385: CODEC Digital Inputs

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|-------------|--|--------------------------------------|-----------|-----|-----------|------|
| Vil_dig | Logic 0 input level all digital pads | VDDIO = 1.75 V to 3.0 V (Note 50) | | | 0.3*VDDIO | V |
| Vil_dig_33 | Logic 0 input level all digital pads | VDDIO = 3 V to 3.45 V (Note 50) | | | 0.9 | V |
| Vih_dig | Logic 1 input level all digital pads | VDDIO = 1.75 V to 3.45 V (Note 50) | 0.7*VDDIO | | | V |
| Ileak_hi | Input current of all inputs with (programmable) pull-down resistors disabled | Vin = VDDIO | | | 10 | μA |
| Ileak_lo | Input current of all inputs with (programmable) pull-up resistors disabled. | Vin = VSS | | | 10 | μA |
| Ipull_up_lo | Input current with internal pull-up enabled. | Vin = VSS VDDIO = 1.75 V to 3.3 V | 20 | | 100 | μA |

Note 50: To meet the specified Vil/Vih levels, BAT_CTRL_REG[PADS_A] bits must be set to the corresponding applied VDDIO voltage.

Table 386: CODEC Digital Outputs

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|-----------|----------------------|---|-------------|-----|-----------|------|
| Vol_100u | Logic 0 output level | Iout = 100 μA | | | 0.1 | V |
| Vol_175 | Logic 0 output level | Iout = 2 mA VDDIO = 1.75 V to 1.98 V | | | 0.2*VDDIO | V |
| Vol_25 | Logic 0 output level | Iout = 4 mA VDDIO = 2.25 V to 2.75 V | | | 0.2*VDDIO | V |
| Vol_33 | Logic 0 output level | Iout = 8 mA VDDIO = 3.0 V to 3.45 V | | | 0.2*VDDIO | V |
| Voh_100u | Logic 1 output level | Iout = 100 μA. VDDIO = 3.45 V | VDDIO - 0.1 | | | V |
| Voh_175 | Logic 1 output level | Iout = 2 mA VDDIO = 1.75 V to 1.98 V | 0.8*VDDIO | | | V |
| Voh_25 | Logic 1 output level | Iout = 4 mA VDDIO = 2.25 V to 2.75 V | 0.8*VDDIO | | | V |
| Voh_33 | Logic 1 output level | Iout = 8 mA VDDIO = 3.0 V to 3.45 V | 0.8*VDDIO | | | V |

30.2 ANALOG FRONTEND SPECIFICATIONS

Table 387: Microphone Amplifier

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|---------------|---|--|------|------|------|------------|
| Vmic_0dB_unt | Untrimmed differential RMS input voltage between MICp and MICn (0 dBm0 reference level) | 0 dBm0 on COUT (Note 51) MIC_GAIN[3:0] = 0, @ 1020 Hz; Tolerance: • 13% when untrimmed (BANDGAP_REG=8) • 6% when trimmed (Note 52) | 114 | 131 | 149 | mV |
| Vmic_cm_level | MICp and MICn common mode voltage | | | 0.9 | | V |
| Amic_gain_0 | Reference level at Vmic_0dB_unt | MIC_GAIN[3:0] = 0000 | | 0 | | dB |
| Amic_gain_1 | Microphone gain relative to Amic_gain_0 | MIC_GAIN[3:0] = 0001 | | 1.8 | | dB |
| Amic_gain_2 | | MIC_GAIN[3:0] = 0010 | | 4.1 | | dB |
| Amic_gain_3 | | MIC_GAIN[3:0] = 0011 | | 6.0 | | dB |
| Amic_gain_4 | | MIC_GAIN[3:0] = 0100 | | 7.8 | | dB |
| Amic_gain_5 | | MIC_GAIN[3:0] = 0101 | | 10.1 | | dB |
| Amic_gain_6 | | MIC_GAIN[3:0] = 0110 | | 12.0 | | dB |
| Amic_gain_7 | | MIC_GAIN[3:0] = 0111 | | 14.5 | | dB |
| Amic_gain_8 | | MIC_GAIN[3:0] = 1000 | | 16.1 | | dB |
| Amic_gain_9 | | MIC_GAIN[3:0] = 1001 | | 18.1 | | dB |
| Amic_gain_A | | MIC_GAIN[3:0] = 1010 | | 20.6 | | dB |
| Amic_gain_B | | MIC_GAIN[3:0] = 1011 | | 22.0 | | dB |
| Amic_gain_C | | MIC_GAIN[3:0] = 1100 | | 24.0 | | dB |
| Amic_gain_D | | MIC_GAIN[3:0] = 1101 | | 26.4 | | dB |
| Amic_gain_E | | MIC_GAIN[3:0] = 1110 | | 27.9 | | dB |
| Amic_gain_F | | MIC_GAIN[3:0] = 1111 | | 29.8 | | dB |
| Rin_mic | Resistance of activated microphone amplifier inputs (MICp, MICn and MICh) to internal buffered AGND | | 75 | 150 | | k Ω |
| Vmic_offset | Input referred DC-offset | MIC_GAIN[3:0] = 1111 3 sigma deviation limits | -2.6 | | +2.6 | mV |

Note 51: 0 dBm0 on COUT = -3.14 dB of max PCM value. COUT is CODEC output in test mode.

Note 52: Trimming possibility is foreseen. At system production the bandgap reference voltage can be controlled within 2% accuracy and data can be stored in Flash. Either AVD or VREF can be trimmed within 2% accuracy. If AVD is trimmed VREF will be within 2% accuracy related to either AVD. Or vice versa VREF can be trimmed. For Vref trimming measure Δ (VREFp VREFm) and update BANDGAP_REG[3..0]

Table 388: Microphone Supply Voltages

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|-------------|------------------------------------|--|------|-----|------|----------|
| Vref_unt | VREFp-VREFm untrimmed | $I_{LOAD} = 0$ mA BANDGAP_REG = 8 (Note 52) | 1.41 | 1.5 | 1.59 | V |
| Rout_vrefp | VREFp output resistance | | | 10 | | Ω |
| Nvrefp_idle | Peak noise on VREFp/AVS | CCITT weighted | | | -120 | dBV |
| PSRRvrefp | Power supply rejection Vref output | See Figure 82 AVD2 to VREFp/m $f = 100$ Hz to 4 kHz BANDGAP_REG[5:4] = 3 | 40 | | | dB |

Note 53: Vrefm is a clean ground input and is the 0 V reference.

Table 389: VREFp Load Circuits

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|-------------|------------------------------------|------------|-----|-----|-----|------|
| Cload_vrefp | VREFp (parasitic) load capacitance | | | | 20 | pF |
| Iout_vrefp | VREFp output current | | | | 1 | mA |

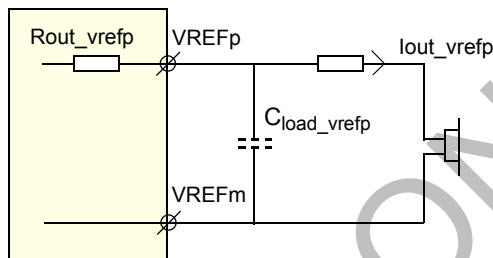


Figure 79: VREFp Load Circuit

Table 390: LSRp/LSRn Outputs

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|--------------|--|--|-----|-----|-----|----------|
| Vlsr_0dB_unt | Untrimmed differential RMS output voltage between LSRp and LSRn in audio mode (0 dBm0 reference level) | 0 dBm0 on CIN (Note 54), LSRATT[2:0] = 001, @ 1020 Hz Load circuit A (see Figure 80, Table 391) with $RL1 = \infty \Omega$, Cp1 or load circuit B (see Figure 81) with RL2, Cp2 and Cs2 <u>Tolerance:</u> <ul style="list-style-type: none"> 13% when untrimmed (BANDGAP_REG = 8) 6% when trimmed (Note 52) | 621 | 714 | 807 | mV |
| Rout_lsr | Resistance of activated loudspeaker amplifier outputs LSRp and LSRn | | | 1 | | Ω |

Table 390: LSRp/LSRn Outputs

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|----------------|---|---|-----|------|-------|------|
| Alsr_df_gain_1 | Reference level at Vlsr_0dB_unt | LSRATT[2:0] =001 | | 0 | | dB |
| Alsr_df_gain_0 | Loudspeaker gain relative to Alsr_df_gain_1 | LSRATT[2:0] =000 | | 2.3 | | dB |
| Alsr_df_gain_2 | | LSRATT[2:0] =010 | | -2.2 | | dB |
| Alsr_df_gain_3 | | LSRATT[2:0] =011 | | -4.0 | | dB |
| Alsr_df_gain_4 | | LSRATT[2:0] =100 | | -5.7 | | dB |
| Alsr_df_gain_5 | | LSRATT[2:0] =101 | | -8.0 | | dB |
| Alsr_df_gain_6 | | LSRATT[2:0] =110 | | -9.9 | | dB |
| Alsr_df_gain_7 | | LSRATT[2:0] =111 | | | -12.1 | |
| Vlsr_dc | DC offset between LSRp and LSRn | LSRATT[2:0] = 3 $R_{L1} = 28 \Omega$ 3 sigma deviation limits | -20 | | 20 | mV |

Note 54: 0 dBm0 on CIN = -3.14 dB of max PCM value.

Table 391: LSR+/LSR- Load Circuits

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|-------------|---------------------------|--|-----|-----|-----|---------------|
| Cp1_RI1_inf | Load capacitance | see Figure 80, $R_{L1} = \infty$ | | | 30 | pF |
| Cp1_RI1_1k | Load capacitance | see Figure 80, $R_{L1} \leq 1 \text{ k}\Omega$ | | | 100 | pF |
| RI1 | Load resistance | | 28 | | | Ω |
| Cp2 | Parallel load capacitance | see Figure 81 | | | 30 | pF |
| Cs2 | Serial load capacitance | | | | 30 | μF |
| RI2 | Load resistance | | 600 | | | Ω |

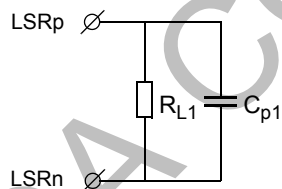


Figure 80: Load Circuit A Dynamic Loudspeaker

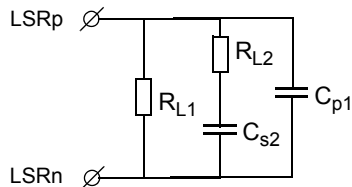


Figure 81: Load Circuit B Piezo Loudspeaker

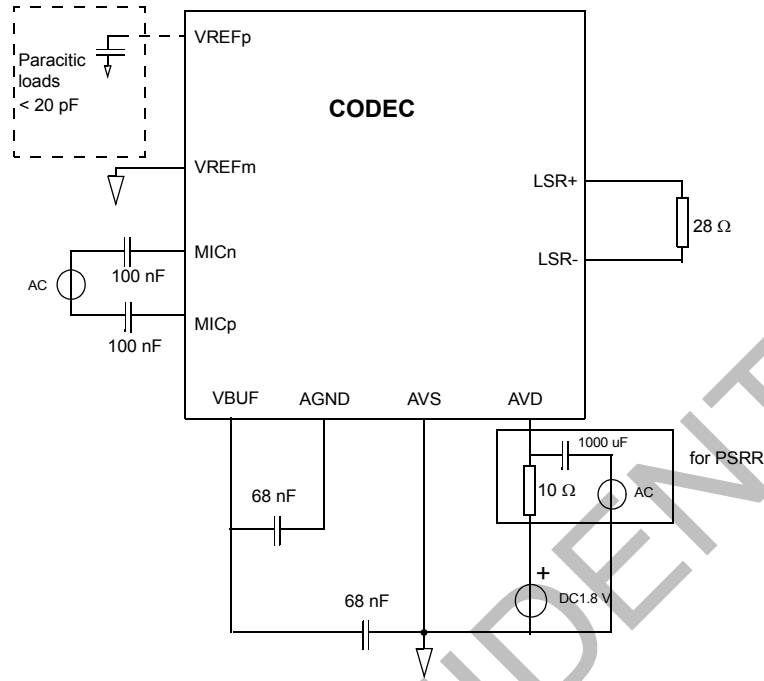


Figure 82: CODEC Measurement Diagram

NDA CONFIDENTIAL

30.3 CODEC AC CHARACTERISTICS

Table 392: CODEC AC Characteristics

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|-----------|---|--|-----|-----|-----|-------|
| SDad_0dB | Signal to total distortion ratio Analog/Digital | see Figure 82, $R_{ab} = \infty$ differential input signal between MICp and MICn with $f = 1020$ Hz ADPCM transcoder active MIC_GAIN[3:0] = 0 0 dBm0 on COUT | 40 | 60 | | dB |
| SDad_40dB | | -40 dBm0 on COUT | 35 | | | dB |
| SDad_45dB | | -45 dBm0 on COUT | 30 | | | dB |
| SDda_0dB | Signal to total distortion ratio Digital/Analog | see Figure 82 differential input signal between MICp and MICn with $f = 1020$ Hz ADPCM transcoder active LSRATT[2-0] = 3 0 dBm0 on CIN | 40 | 65 | | dB |
| SDda_40dB | | -40 dBm0 on CIN | 35 | | | dB |
| SDda_45dB | | -40 dBm0 on CIN | 30 | | | dB |
| Nad_idle | Idle channel noise Analog/Digital | see Figure 82, $R_{ab} = 0$ Ohm Relative to 0 dBm0 MIC_GAIN[3:0] = 0x0B | | -80 | | dBm0p |
| Nda_idle | Idle channel noise Digital/Analog | see Figure 82 Relative to 0 dBm0 LSRATT[2-0] = 3 | | -83 | -77 | dBm0p |
| PSRRad_0 | Power supply rejection ratio Analog/Digital | See Figure 82, AVD2 to COUT, $f = 100$ Hz to 4 kHz MIC_GAIN[3:0] = 0 | 40 | | | dB |
| PSRRad_F | | MIC_GAIN[3:0] = 0x0F | 30 | | | dB |
| PSRRda | Power supply rejection ratio Digital/Analog | See Figure 82, AVD2 to LSRp/n, $f = 100$ Hz to 4 kHz LSRATT[2-0] = 3 | 40 | | | dB |

Table 393: CODEC Frequency Responses

| Parameter | Description | Conditions | Result |
|-----------|------------------------------------|---|--|
| Fad_freq | Frequency response Analog/ Digital | see Figure 82, $R_{ab} = \infty$ relative to 1020 Hz | See frequency response diagrams Figure 83 on page 258, Figure 84 on page 258, Figure 85 on page 259 |
| Fda_freq | Frequency response Digital/Analog | see Figure 82, $R_{ab} = 1$ k Ω relative to 1020 Hz | See frequency response diagrams Figure 86 on page 259, Figure 87 on page 260, Figure 88 on page 260 |

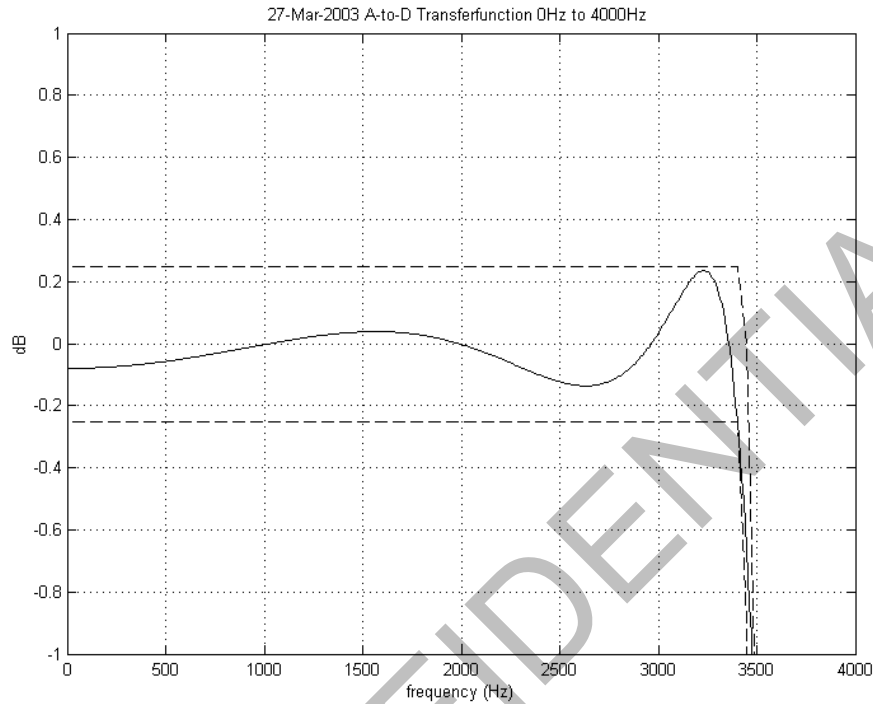


Figure 83: CODEC Frequency Response, Analog to Digital, Pass Band

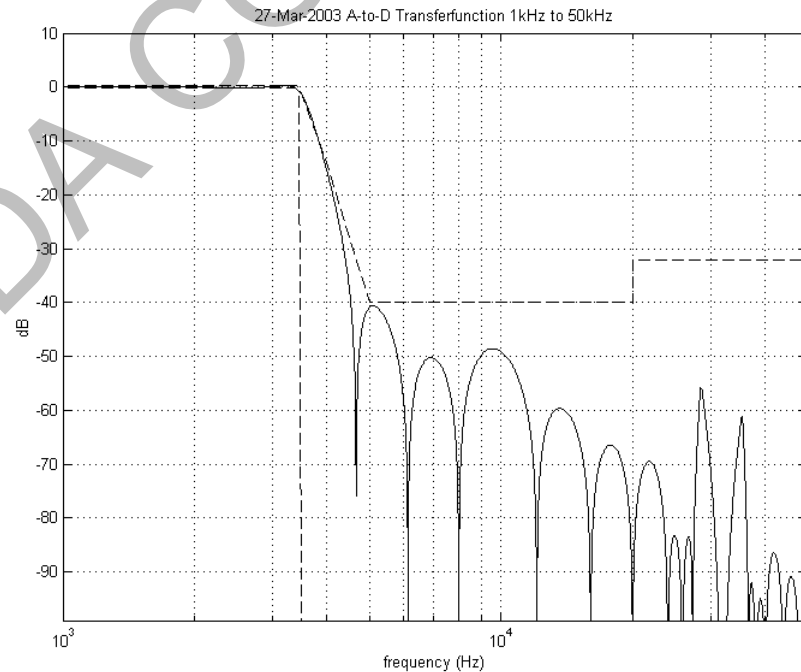


Figure 84: CODEC Frequency Response, Analog to Digital, Stop Band

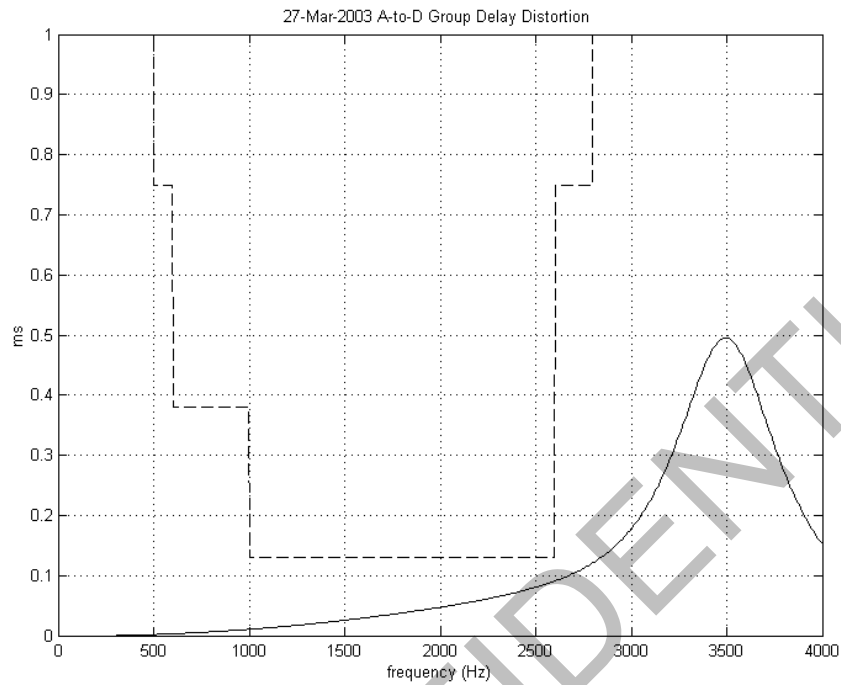


Figure 85: CODEC Analog to Digital, Group Delay

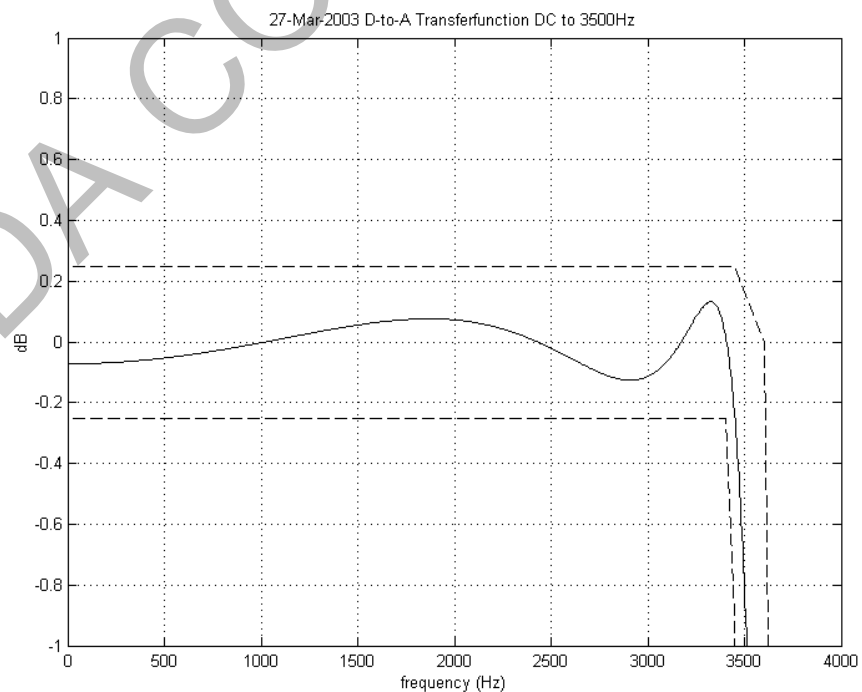


Figure 86: Frequency Response, Digital to Analog, Pass Band

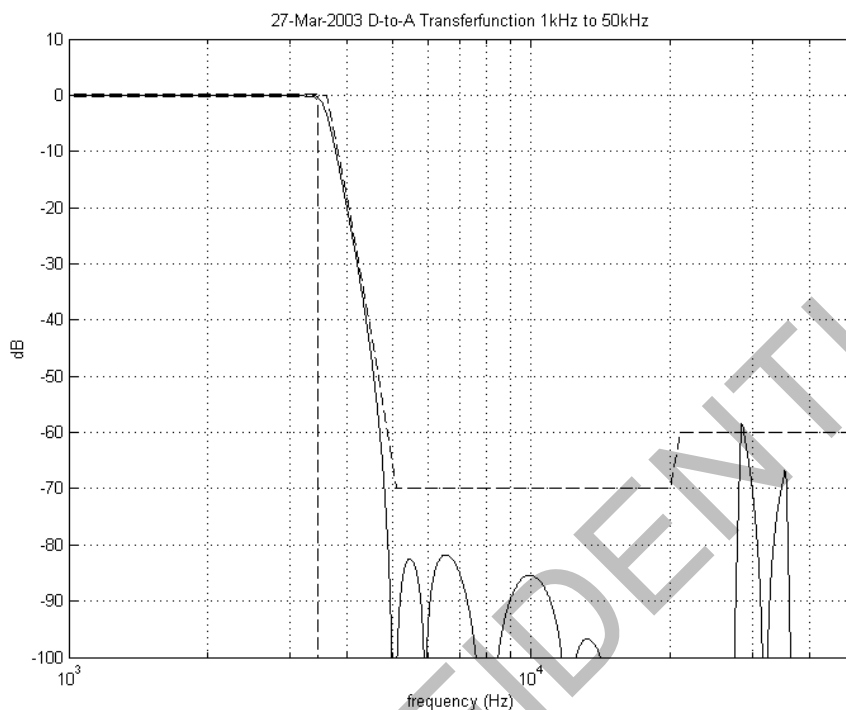


Figure 87: CODEC Frequency Response, Digital to Analog, Stop Band

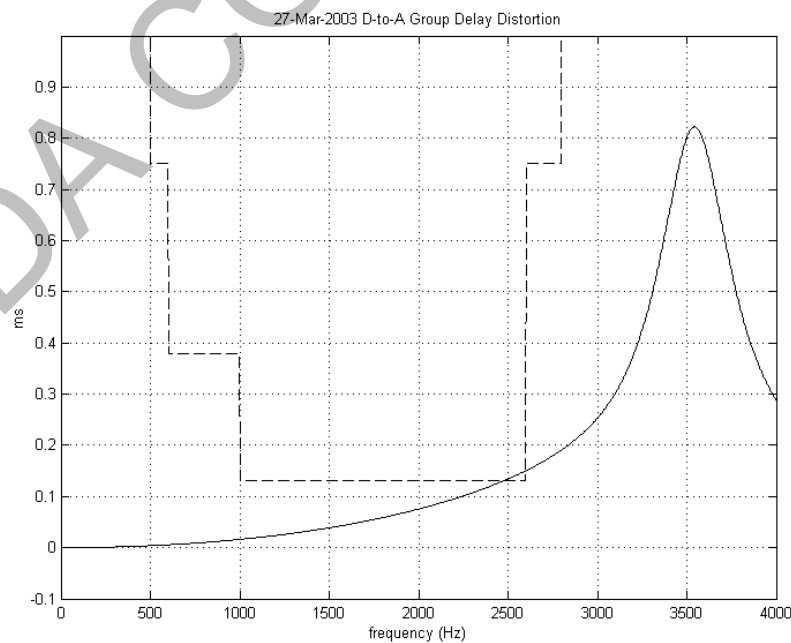


Figure 88: CODEC Digital to Analog, Group Delay Distortion

30.4 CODEC TIMING CHARACTERISTICS

Table 394: PLL Characteristics

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|----------------|-----------------------|---|-----|-------|-----|------|
| Tlock_pll_lpf0 | PLL Lock time | CLK = 16 MHz Fpll = 46.08 MHz, LPF_EN = 0 | | | 0.5 | ms |
| Fpll_vco | PLL maximum frequency | | | 46.08 | | MHz |

Table 395: Digital Inputs and RSTn Pin

| Parameter | Description | Conditions | Min | Typ | Max | Unit |
|-----------|--|------------|-----|-----|-----|------|
| Tlow_rst | Minimum low time RSTn pin during active CLK. | (Note 55) | 1 | | | μs |
| Trf | Maximum rise and fall time of all digital inputs including RSTn pin. | | | | 25 | ns |

Note 55: spikes down to 5 ns may reset the device.

31 Package Information

31.1 MOISTURE SENSITIVITY LEVEL (MSL)

The MSL is an indicator for the maximum allowable time period (floor life time) in which a moisture sensitive plastic device, once removed from the dry bag, can be exposed to an environment with a maximum temperature of 30 °C and a maximum relative humidity of 60 % RH. before the solder reflow process.

QFN packages are qualified for MSL 3.

| MSL Level | Floor Life Time |
|-----------|------------------------------|
| MSL 4 | 72 hours |
| MSL 3 | 168 hours |
| MSL 2A | 4 weeks |
| MSL 2 | 1 year |
| MSL 1 | Unlimited at 30 °C / 85 % RH |

31.2 SOLDERING INFORMATION

Refer to the JEDEC standard J-STD-020 for relevant soldering information.

This document can be downloaded from <http://www.jedec.org>

31.3 PACKAGE OUTLINES

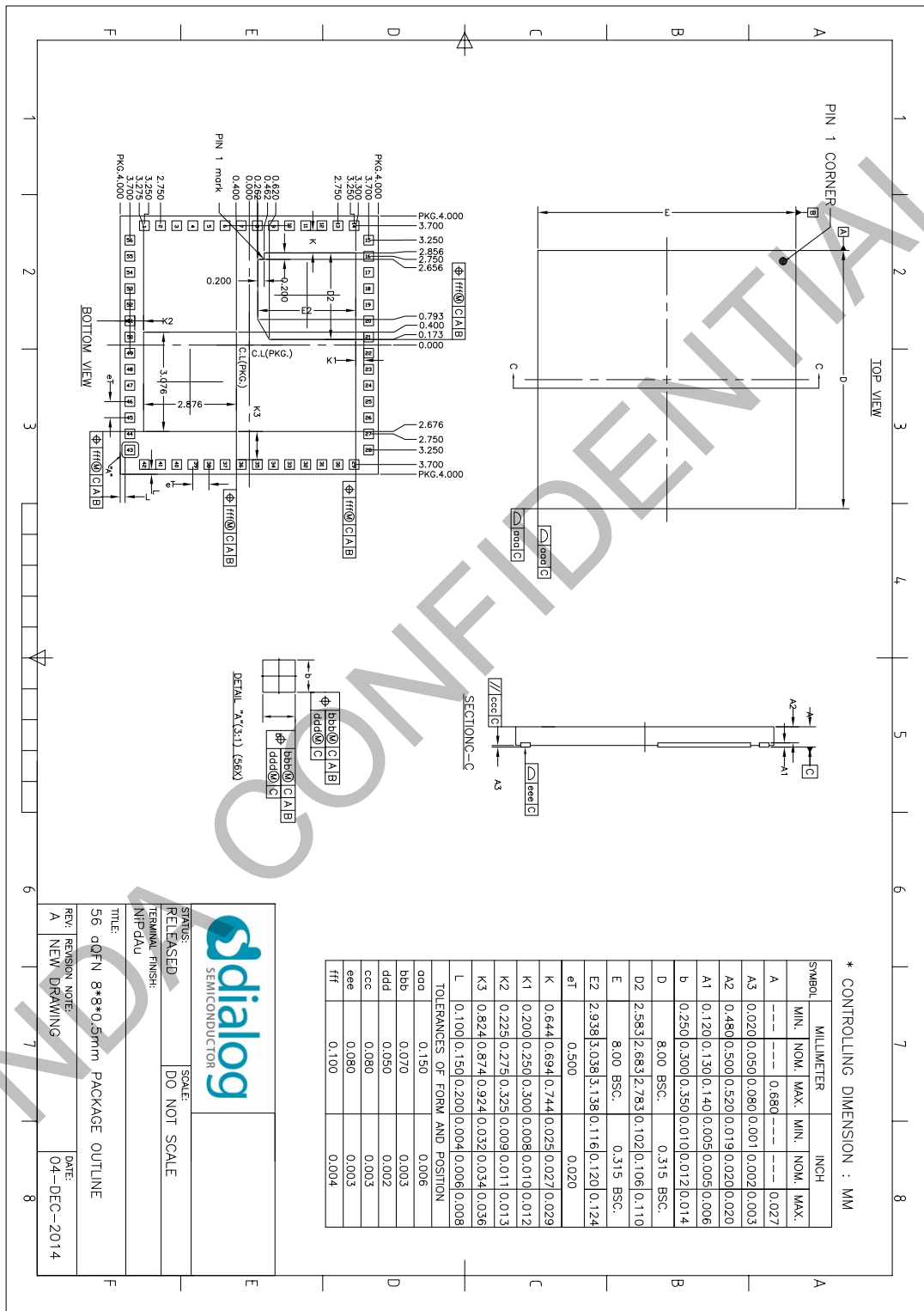


Figure 89: QFN56 Package Outline Drawing

Status Definitions

| Version | Datasheet Status | Product Status | Definition |
|---------|------------------|----------------|--|
| 1.<n> | Target | Development | This datasheet contains the design specifications for product development. Specifications may change in any manner without notice. |
| 2.<n> | Preliminary | Qualification | This datasheet contains the specifications and preliminary characterization data for products in pre-production. Specifications may be changed at any time without notice in order to improve the design. |
| 3.<n> | Final | Production | This datasheet contains the final specifications for products in volume production. The specifications may be changed at any time in order to improve the design, manufacturing and supply. Major specification changes are communicated via Customer Product Notifications. Datasheet changes are communicated via www.dialog-semiconductor.com . |
| 4.<n> | Obsolete | Archived | This datasheet contains the specifications for discontinued products. The information is provided for reference only. |

RoHS Compliance

Dialog Semiconductor's suppliers certify that its products are in compliance with the requirements of Directive 2011/65/EU of the European Parliament on the restriction of the use of certain hazardous substances in electrical and electronic equipment. RoHS certificates from our suppliers are available on request.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01 Jan 2024)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.