RA6W2

Dual Band Wi-Fi 6 and Bluetooth LE 5.1 Combo IC

The RA6W2 is a highly integrated ultra-low-power Wi-Fi + Bluetooth® Low Energy (Bluetooth LE) combo IC solution. This product includes a highly integrated ultra-low-power Wi-Fi microcontroller unit (MCU) integrating an Arm® Cortex®-M33 system processor with a dual band 802.11a/b/g/n/ax Wi-Fi subsystem, on-chip memory, flexible peripheral interfaces and power management features. The RA6W2 also has a Bluetooth® LE that has a 2.4 GHz transceiver and an Arm® Cortex-M0+® microcontroller with a RAM of 48 kB and a One-Time Programmable (OTP) memory of 32 kB. The radio transceiver, the baseband processor, and the qualified Bluetooth® low energy stack is fully compliant with the Bluetooth® LE 5.1 standard.

The RA6W2 is a synthesis of breakthrough ultra-low-power technologies, which enables extremely low-power operation in the system. The Wi-Fi and Bluetooth® LE shut down every micro element of the chip that is not in use, which creates a power consumption that is near zero when not actively transmitting or receiving data. Such low-power operation can extend the battery life up to a year or more depending on the application. The Wi-Fi also enables ultra-low-power transmission and reception modes when the MCU needs to be awake to exchange information with other devices. Advanced algorithms enable sleep mode until the exact moment when the wake-up is required to transmit or receive data.

Features

- **Arm® Cortex®-M33 Core**
  - Armv8-M architecture
  - Maximum operating frequency: 160 MHz
  - Arm Memory Protection Unit (Arm MPU)
  - SysTick timer
- **Full Networking OS and TCP/IP Stack**
  - Comprehensive networking software stack
  - Provides TCP/IP stack: in the form of network socket APIs
- **Wi-Fi Processor**
  - IEEE 802.11b/g/n/ax, 1×1, 20 MHz channel bandwidth, 2.4 GHz
  - IEEE 802.11a/n/ac/ax, 1×1, 20 MHz channel bandwidth, 5 GHz
  - On-chip PA, LNA, and RF switch
  - Wi-Fi security: WPA/WPA2-Enterprise/Personal, WPA2 SI, WPA3 SAE, and OWE
  - Vendor EAP types: EAP-TTLS/MSCHAPv2, PEAPv0/EAP-MSCHAPv2, PEAPv1, EAP-FAST, and EAP-TLS
  - Operating modes: Station, SoftAP
  - WPS-PIN/PBC for easy Wi-Fi provisioning
  - Connection manager for autonomous and fast Wi-Fi connections
  - Bluetooth® coexistence
- **Wi-Fi Alliance Certifications**
  - Wi-Fi CERTIFIED™ b, g, n
  - WPA™ – Enterprise, Personal
  - WPA2™ – Enterprise, Personal
  - WPA3™ – Enterprise, Personal
  - Wi-Fi Enhanced Open™
  - WMM
  - WMM – Power Save
  - Wi-Fi Protected Setup
- **Clock Source**
  - 40 MHz crystal (± 20 ppm) for master clock
  - 32.768 kHz crystal (± 250 ppm) for RTC clock
  - Integrated 32 kHz RC oscillator
- **Interfaces**
  - eMMC/SD expanded memory
  - SDIO Host/Slave function
  - OQSPI with encrypted XIP for external code/data flash
  - QSPI for additional PSRAM storage
  - UART × 2
  - SPI Master/Slave interface
  - I2C Master/Slave interface
  - I2S for digital audio streaming
  - PDM/Digital Mic for digital audio streaming
  - General PWM Timer 32-bit × 8
  - Multiplexed GPIO × 22
  - ADC (12-bit SAR) for sensor interfaces × 4
- **Advanced Security and Encryption**
  - Secure Crypto Engine
  - Symmetric algorithms: AES, DES/3DES, CHACHA

- Hash/HMAC: SHA1/224/256
- Asymmetric algorithms: RSA, DH, ECC
- TRNG
- Secure boot
- Secure debug (SWD)
- Secure asset storage
- Device lifecycle management
- TLS/DTLS protocol acceleration

- **Memory**
  - SRAM: 704 kB
  - Retention Memory: 64 kB
  - ROM: 256 kB
  - OTP: 2 kB
- **Power Management Unit**
  - On-Chip RTC
  - Wake-up control of fast booting or full booting with minimal initialization time
  - Integrated DC-DC and LDOs
  - Support for ultra-low power Sleep modes
- **Supply**
  - Single operating voltage: 1.8 V to 3.6 V (typical: 3.3 V)
  - Digital I/O Supply Voltage: 1.8 V/3.3 V

  Blackout and brownout detector
- **Bluetooth**
  - Compatible with Bluetooth® v5.1, ETSI EN 300 328 and EN 300 440 Class 2 (Europe), FCC CFR47 Part 15 (US) and ARIB STD-T66 (Japan) core
  - Supports up to 3 connections
- **Processing and Memories**
  - 16 MHz 32-bit Arm Cortex-M0+ with SWD interface
  - 48 kB RAM
  - 144 kB ROM
  - 32 kB OTP
- **Current Consumption**
  - 2 mA RX at VBAT = 3 V
  - 4 mA TX at VBAT = 3 V and 0 dBm

1.8 µA at sleep with all RAM retained

- **Radio**
  - Programmable RF transmit power
  - -93 dBm receiver sensitivity

Applications

- **Home Appliances**
  - Smart air-conditioners
  - Smart dishwashers
  - Printers
  - Washing machines
  - Refrigerators
  - Vacuum cleaners
  - Dryers
  - Coffee makers
  - Air purifiers
- **Home Automation**
  - Smoke detectors
  - Smart thermostats
  - Smart cameras
  - Consumer and DIY video
  - Digital voice assistant
  - Smart plugs
  - Smart door locks
  - Smart video doorbells
  - Smart building
  - Energy management devices
  - Smart smoke detectors
  - Garage door openers
  - Educational toys
  - Wi-fi enabled toys
  - Baby monitors
  - Internet music players
  - Live streaming devices
  - Generic IoT sensors
  - Light control
  - Smart light bulbs
  - Garage door openers
  - Motion sensors
  - Air quality sensors

- Smart blinds **Wearables**
  - Smart watches
  - Smart rings
  - Smart bracelets
  - Smart bands
  - Elderly tracking devices
  - Pet trackers
  - Child location monitors
- **Industrial**
  - Smart watches
  - Electronic shelf labels
  - Building security access control
  - Connected solar panels

- Smart EV chargers
- Agricultural sensors and monitors
- Environmental sensors: humidity, CO/CO2, methane, fine particles etc.
- Point of sales (POS) solutions
- Retail location trackers
- Robotics
- Factory automation devices
- Industrial wireless sensors

Contents

Figures

Tables

1. Block Diagrams

The RA6W2 provides a high level of integration for a battery powered wireless system, with integrated dual band IEEE 802.11 b/g/n/ax and Bluetooth V5.1. The RA6W2 is designed to address the needs of battery powered devices that require minimal power consumption and reliable operation.

Figure 1 shows all the physical blocks in RA6W2.

RA6W2: Dual Band Wi-Fi 6/Bluetooth LE 5.1 Combo IC

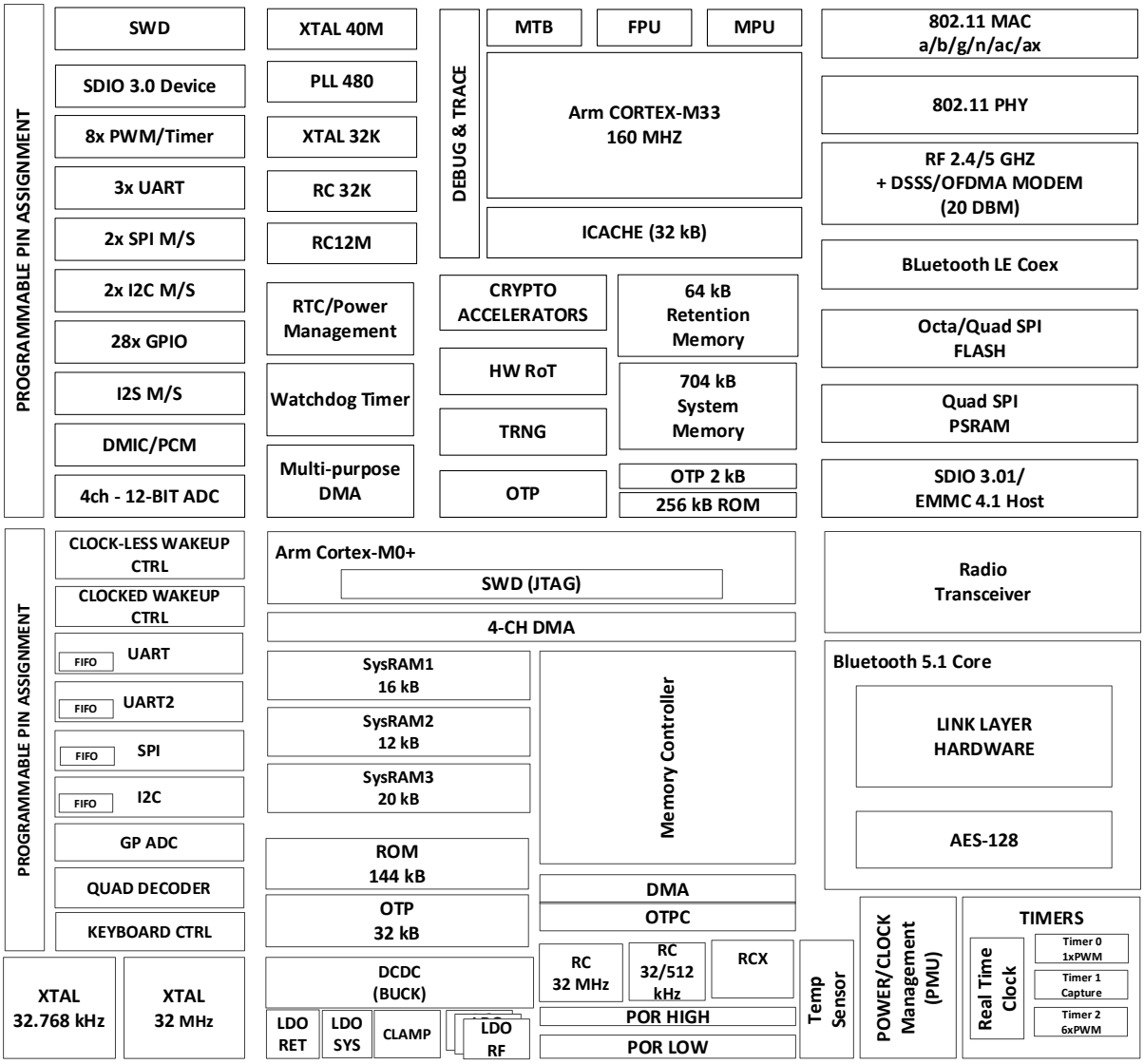


Figure 1. RA6W2 block diagram

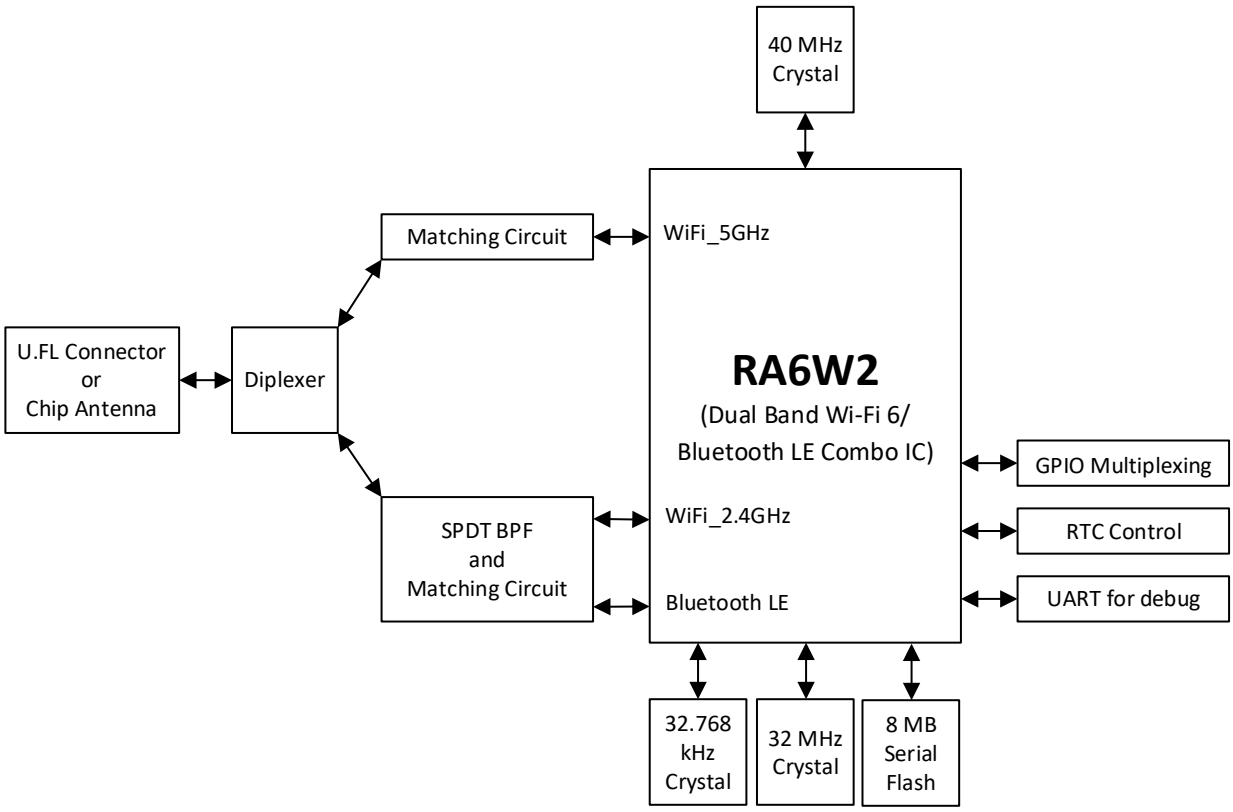


Figure 2. RA6W2 application diagram

2. Part Numbering

Figure 3 shows the product part number information, including memory capacity and package type. Table 1 shows a list of products.

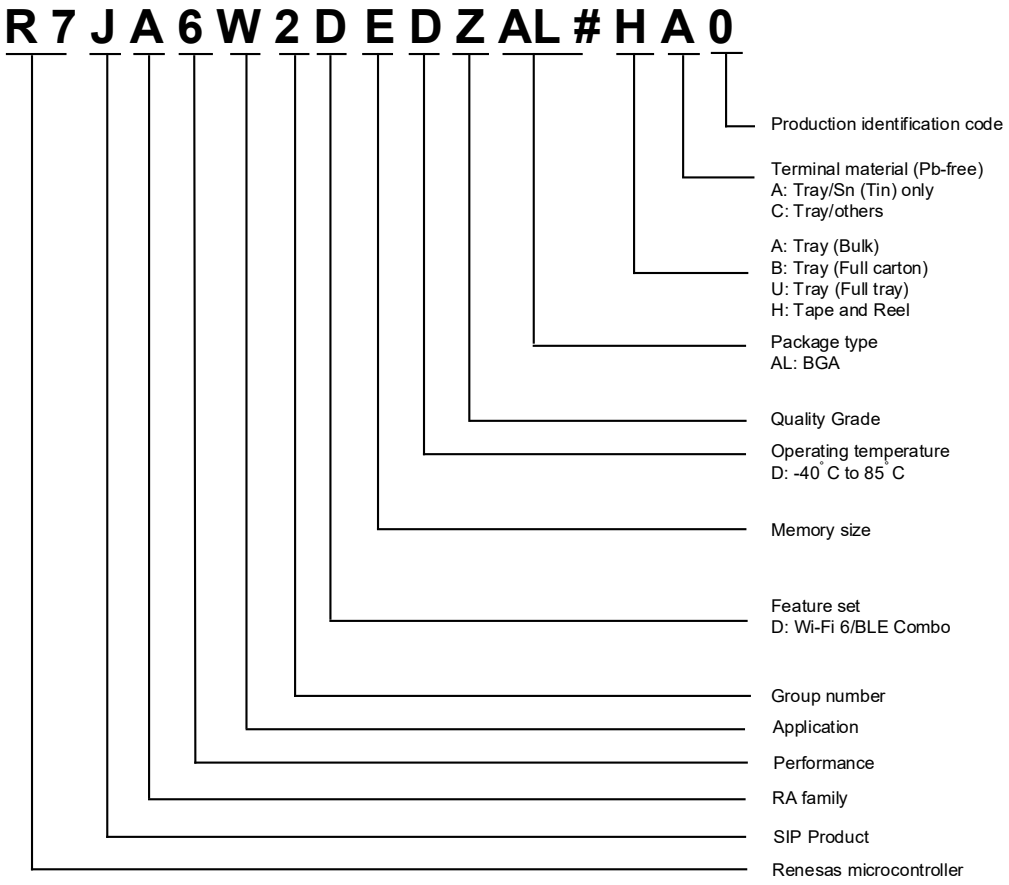


Figure 3. Part numbering scheme

Table 1. Product list

Product part number	Package code	Operating temperature	Package type	Shipment form	Pack quantity production
R7JA6W2DEDZAL	BW0093AA	-40 to 85°C	BGA	Reel	4000

3. Pinout

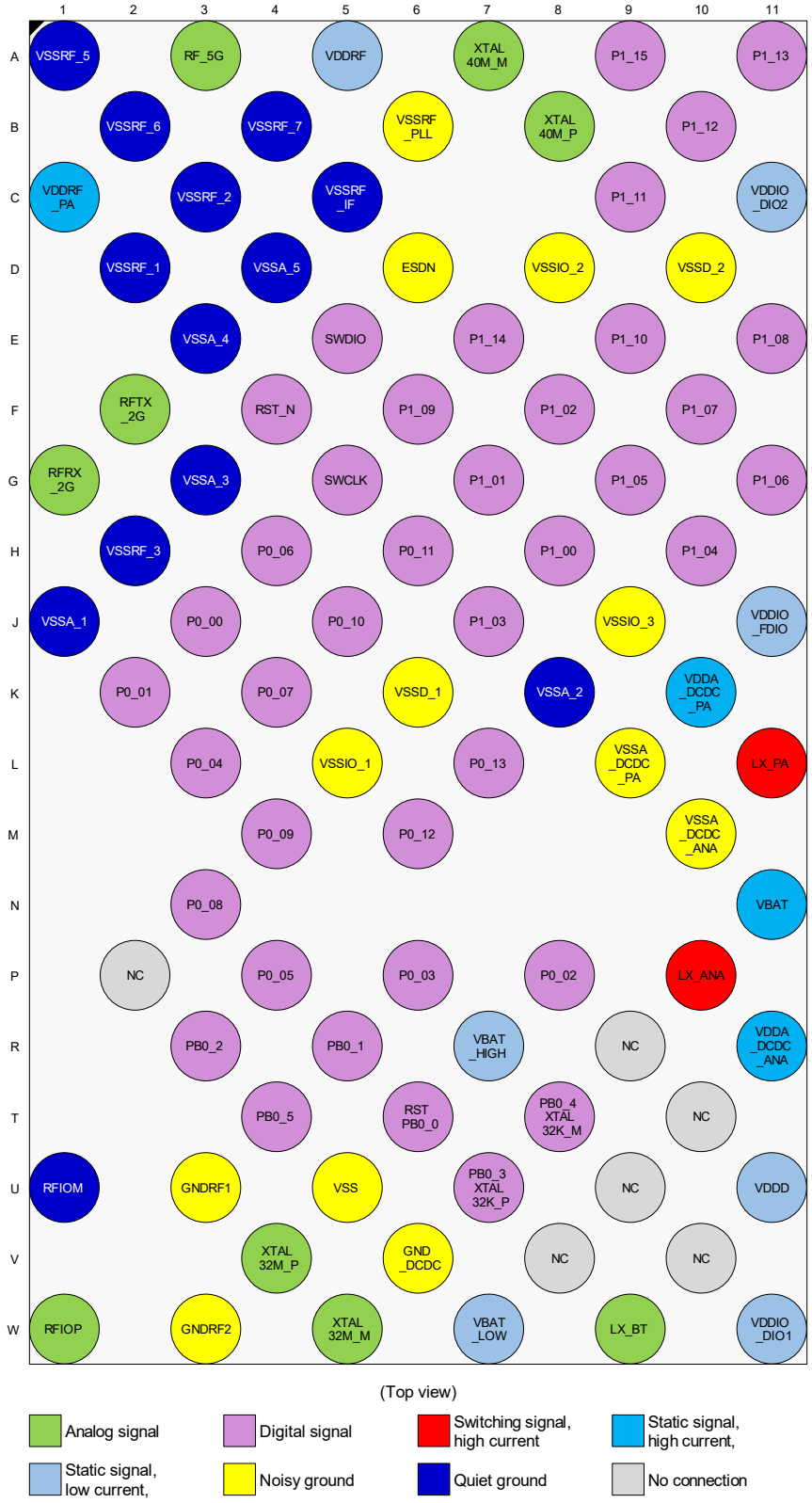


Figure 4. RA6W2 BGA93 pinout diagram (top view)

Table 2: Pin description

Ball	Ball name	Type	Related device	Description
A1	VSSRF_5	GND	Wi-Fi	RF ground.
A3	RF_5G	AIO	Wi-Fi	5G band transmitter output and recevier input, connect to RFSW.
A5	VDDRF	AI	Wi-Fi	1.37 V from DCDC_ANA.
A7	XTAL40M_M	AIO	Wi-Fi	40M XTAL oscillator output.
A9	P1_15	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor. ATB for analog test.
A11	P1_13	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor.
B2	VSSRF_6	GND	Wi-Fi	RF ground.
B4	VSSRF_7	GND	Wi-Fi	RF ground.
B6	VSSRF_PLL	GND	Wi-Fi	GND (RF_PLL).
B8	XTAL40M_P	AIO	Wi-Fi	40M XTAL oscillator input.
B10	P1_12	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor.
C1	VDDRF_PA	AI	Wi-Fi	PA power, 1.35 V from DCDC_PA.
C3	VSSRF_2	GND	Wi-Fi	GND (TX5G).
C5	VSSRF_IF	GND	Wi-Fi	GND (RXIF).
C9	P1_11	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor.
C11	VDDIO_DIO2	AI	Wi-Fi	Digital I/O power: 1.8~3.3 V (typ).
D2	VSSRF_1	GND	Wi-Fi	GND (TX2G).
D4	VSSA_5	GND	Wi-Fi	GND (rcore).
D6	ESDN	GND	Wi-Fi	GND (RF_DIG).
D8	VSSIO_2	GND	Wi-Fi	GND (digital I/O).
D10	VSSD_2	GND	Wi-Fi	GND (dcore).
E3	VSSA_4	GND	Wi-Fi	GND (rcore).
E5	SWDIO	DIO	Wi-Fi	INPUT. Serial Wire Debug clock.
E7	P1_14	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor. ATB for analog test.
E9	P1_10	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor.
E11	P1_08	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor.
F2	RFTX_2G	AIO	Wi-Fi	2G band transmitter input, connect to inductor.
F4	RST_N	DIO	Wi-Fi	INPUT. Device reset (active LOW). (VBAT power domain)
F6	P1_09	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor.
F8	P1_02	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor.
F10	P1_07	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor.
G1	RFRX_2G	AIO	Wi-Fi	2G band recevier input, connect to inductor.
G3	VSSA_3	GND	Wi-Fi	GND (rcore).
G5	SWCLK	DIO	Wi-Fi	INPUT/OUTPUT. Serial Wire Debug data I/O.
G7	P1_01	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor.
G9	P1_05	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor.

Ball	Ball name	Type	Related device	Description
G11	P1_06	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor.
H2	VSSRF_3	GND	Wi-Fi	GND (RX2G).
H4	P0_06	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor.
H6	P0_11	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor.
H8	P1_00	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor.
H10	P1_04	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor.
J1	VSSA_1	GND	Wi-Fi	GND (acore).
J3	P0_00	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor. Analog sharing (VBAT power domain).
J5	P0_10	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor.
J7	P1_03	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor.
J9	VSSIO_3	GND	Wi-Fi	GND (Digital I/O).
J11	VDDIO_FDIO	AI	Wi-Fi	Flash I/O power: 1.8 V from FDIO_LDO output, have to external capacitor 1 µF.
K2	P0_01	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor. Analog sharing (VBAT power domain).
K4	P0_07	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor. Analog sharing: GPADC input channel 4, ATB for IQ signal test.
K6	VSSD_1	GND	Wi-Fi	GND (dcore).
K8	VSSA_2	GND	Wi-Fi	GND (ana_ldo_buck).
K10	VDDA_DCDC_PA	AI	Wi-Fi	DCDC_PA output: 1.35 V, connect to VDDRF_PA (PA power), external capacitor: 10 µF.
L3	P0_04	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor. Analog sharing: GPADC input channel 1, ATB for IQ signal test.
L5	VSSIO_1	GND	Wi-Fi	GND (Digital I/O).
L7	P0_13	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor.
L9	VSSA_DCDC_PA	GND	Wi-Fi	DCDC_PA analog ground.
L11	LX_PA	AIO	Wi-Fi	DCDC_PA switching: 4.7 µH inductor (recommend: LQM21PN4R7MGH (Murata)).
M4	P0_09	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor.
M6	P0_12	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor.
M10	VSSA_DCDC_ANA	GND	Wi-Fi	DCDC_ANA, connect to VDDRF (1.375 V), external capacitor: 10 µF.
N3	P0_08	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor.
N11	VBAT	AI	Wi-Fi	Battery input: 1.9~3.6 V.
P2	NC		Wi-Fi	Not Connected.
P4	P0_05	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor. Analog sharing: GPADC input channel 2, ATB for IQ signal test.
P6	P0_03	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor. Analog sharing: XTAL32K_P (VBAT power domain).
P8	P0_02	DIO	Wi-Fi	INPUT/OUTPUT with pull-up/down resistor. Analog sharing: XTAL32K_M (VBAT power domain).

Ball	Ball name	Type	Related device	Description
P10	LX_ANA	AIO	Wi-Fi	DCDC_ANA switching: 4.7 µH inductor (recommend: LQM21PN4R7MGH (Murata)).
R3	PB0_2	DIO	Bluetooth LE	General Purpose I/O, JTAG I/F, SWCLK.
R5	PB0_1	DIO	Bluetooth LE	SWDIO for WLCSP (Optional).
R7	VBAT_HIGH	AIO	Bluetooth LE	**B**attery connection or DC-DC output in Buck mode, respectively. IO-supply.
R9	NC			Not Connected.
R11	VDDA_DCDC_ANA	AI	Wi-Fi	DCDC_ANA, connect to VDDRF (1.375 V), external capacitor: 10 µF.
T4	PB0_5	DIO	Bluetooth LE	SWDIO for WLCSP (Default).
T6	RST_PB0_0	DIO	Bluetooth LE	–
T8	PB0_4	DIO	Bluetooth LE	Analog output of the XTAL32K crystal osc.
T10	NC			Not Connected.
U1	RFIOM	AIO	Bluetooth LE	RF input/output, 50 Ω.
U3	GNDRF1	GND	Bluetooth LE	RF ground.
U5	VSS	GND	Bluetooth LE	Digital ground.
U7	PB0_3	DIO	Bluetooth LE	XTAL32K crystal or external clock (square wave).
U9	NC			Not Connected.
U11	VDDD	AI	Wi-Fi	VDDD 1.1 V regulator output.
V4	XTAL32Mp	AIO	Bluetooth LE	Crystal input for the 32 MHz XTAL.
V6	GND_DCDC	GND	Bluetooth LE	DC-DC ground.
V8	NC			Not Connected.
V10	NC			Not Connected.
W1	RFIOP	RF IP	Bluetooth LE	RF input/output, 50 Ω.
W3	GNDRF2	GND	Bluetooth LE	RF ground.
W5	XTAL32Mm	AIO	Bluetooth LE	Crystal output for the 32 MHz XTAL.
W7	VBAT_LOW	AIO	Bluetooth LE	Battery connection or DC-DC output in Buck mode, respectively. System supply.
W9	LX_BT	AIO	Bluetooth LE	Connection for the ext DC-DC inductor.
W11	VDDIO_DIO1	AI	Wi-Fi	Digital I/O power: 1.8~3.3 V (typ).

4. Application Information

Typical use cases are summarized in Table 3.

Table 3: RA6W2 use cases

Use case	# of antennas	Switch type	Reference
Wi-Fi/Bluetooth combo applications with Diplexer	1	Diplexer + SP2T	See Figure 5
Wi-Fi/Bluetooth combo applications with FEM	1	FEM (Bluetooth supported)	See Figure 6

4.1 Typical Wi-Fi/Bluetooth Combo Application with Diplexer

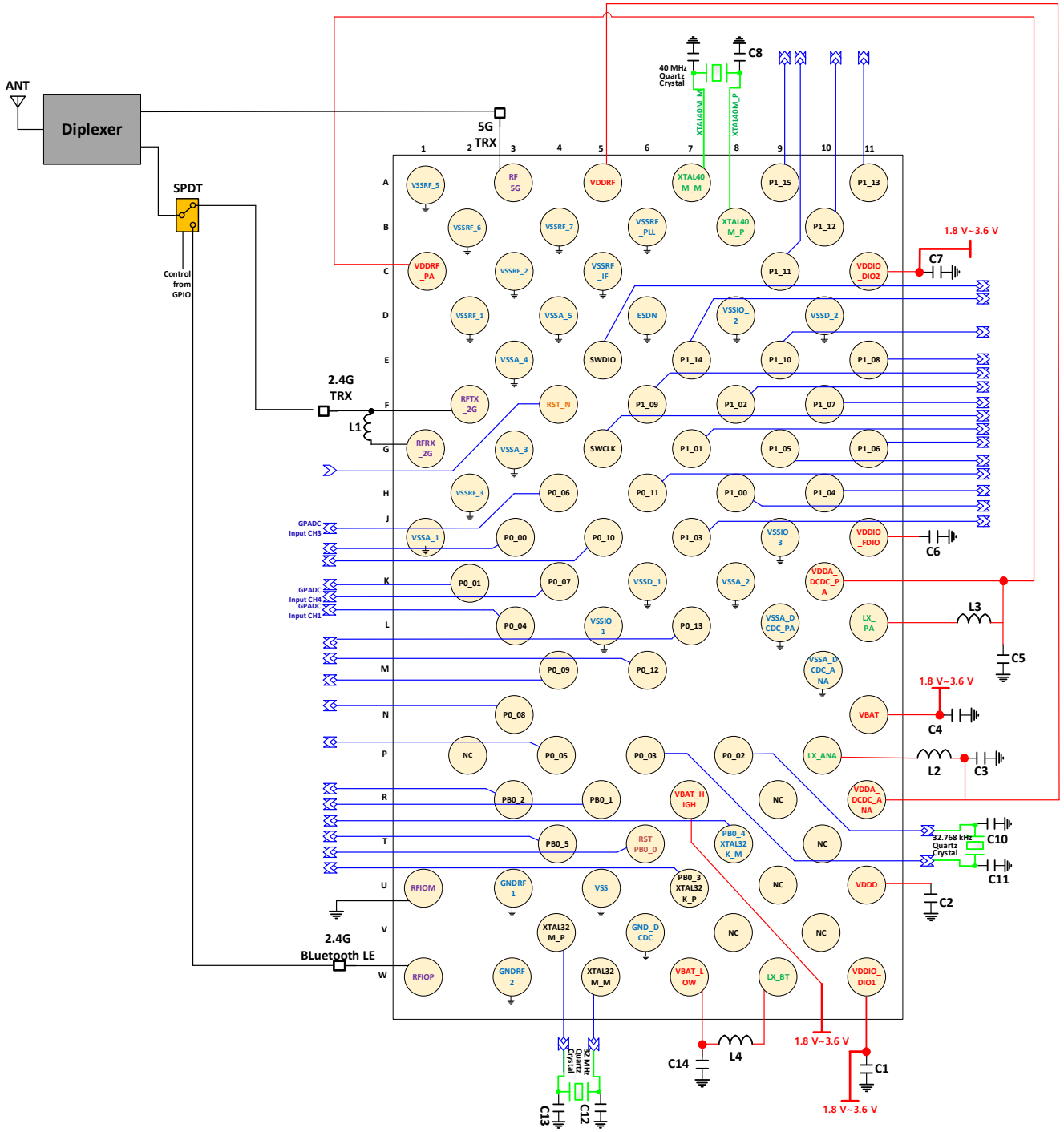


Figure 5. Wi-Fi/Bluetooth combo application with Diplexer

4.2 Typical Wi-Fi/Bluetooth Combo Application with FEM

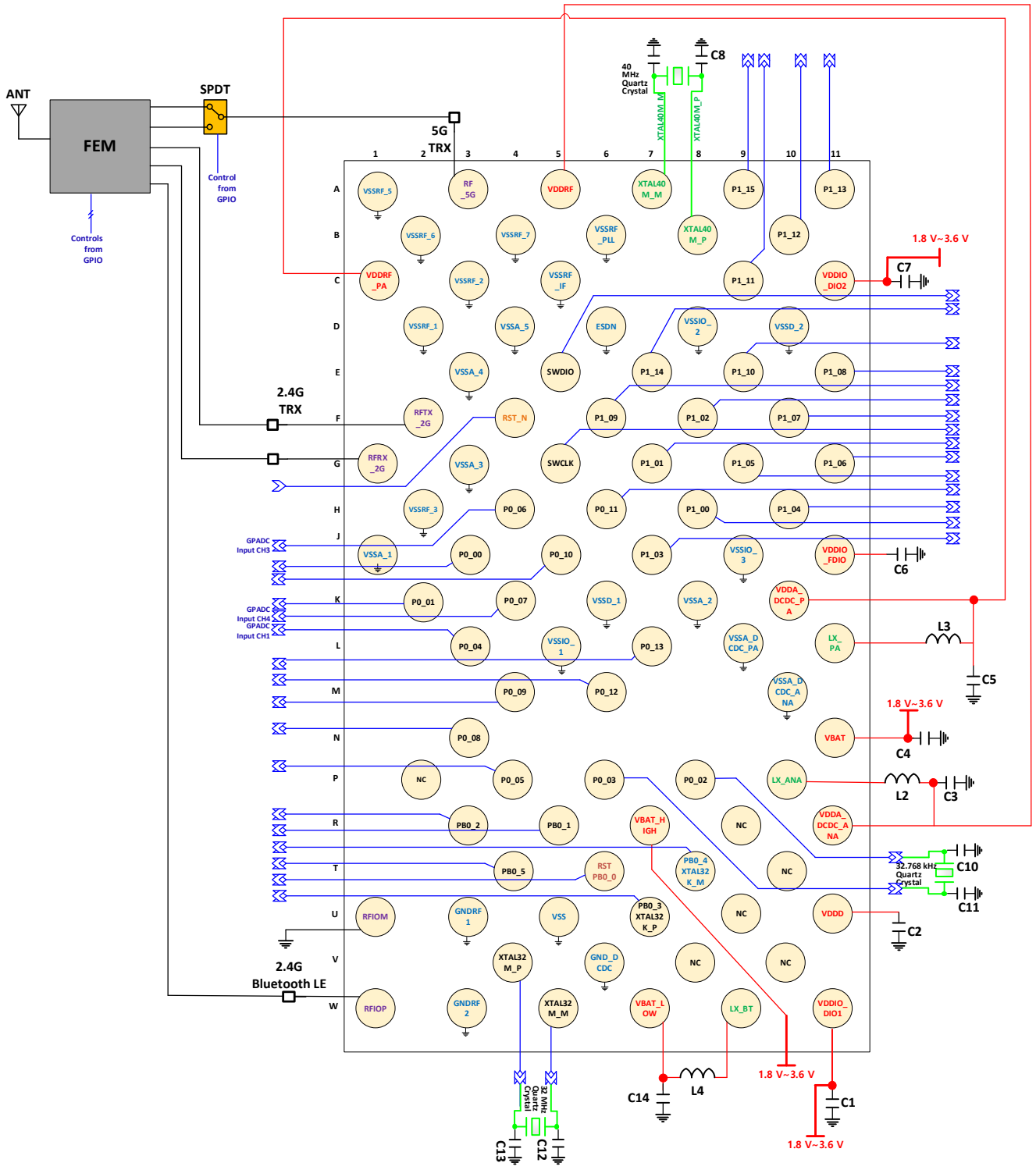


Figure 6. Wi-Fi Bluetooth combo application with FEM

The interconnection between Wi-Fi and Bluetooth LE consists of 4-wire UART, BLE_RST (Bluetooth LE reset), 1-wire coexistence signal, and the control signals of RF Switch (located internally to RRQ61051). All connections are external to RA6W2.

Table 4: RA6W2 Wi-Fi Bluetooth LE communication and co-existence interconnections

GPIO	Wi-Fi Function	GPIO	Bluetooth Function
P1_10	UART0_RX	P0_00	UART TX
P1_11	UART0_TX	P0_01	UART_RX
P1_12	UART0__CTS	P0_03	UART_RTS
P1_13	UART0__RTS	P0_04	UART_CTS
P1_14	BLE_ HW_RST	P0_02	BLE HW_RST
P1_15	COEX	P0_05	iBtAct
P0_06		RFSW1	RF Switch control
P0_07		RFSW2	RF Switch control

5. 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): VBAT = 3.3 V, TA = 25 ˚C. All radio measurements are performed with standard RF measurement equipment providing a source/load impedance of 50 Ω.

The specified Min and Max capacitor values define the range of the effective capacitance, which may vary over the applied voltage because of voltage derating. See the component manufacturer for the capacitor specifications.

5.1 Absolute Maximum Ratings

Stresses beyond those listed under Absolute Maximum Ratings may cause permanent damage to the device. These are stress ratings only, so functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specification are not implied. Exposure to Absolute Maximum Rating conditions for extended periods may affect device reliability.

⚠	**CAUTION**
Do not operate at or near the maximum ratings listed for extended periods of time. Exposure to such conditions can adversely impact product reliability and result in failures not covered by the warranty.	

Table 5: Absolute maximum ratings

Parameter	Description	Conditions	Min	Max	Unit
V_{ESD_CDM}	Electrostatic discharge voltage (Charged Device Model)		-500	500	V
V_{ESD_HBM}	Electrostatic discharge voltage (Human Body Model)		-2000	2000	V
V_{BAT_LIM}	Limiting battery supply voltage		-0.2	3.6	V
$V_{PIN\_LIM\_VDDIO\_DIO1}$	Limiting voltage on a pin		-0.2	3.6	V
$V_{PIN\_LIM\_VDDIO\_DIO2}$	Limiting voltage on a pin		-0.2	3.6	V
$V_{PIN\_LIM\_VDDIO\_FDIO}$	Limiting voltage on a pin		-0.2	3.6	V
$V_{PIN\_LIM\_VDDD}$	Limiting voltage on a pin		-0.1	1.22	V
$V_{PIN\_LIM\_VDDRF}$	Limiting voltage on a pin		-0.1	1.55	V
$V_{PIN\_LIM\_VDDRF\_PA}$	Limiting voltage on a pin		-0.1	1.55	V
$V_{BAT\_LIM\_LOW}$	Limiting supply voltage	Supply voltage in buck configuration	-0.2	3.6	V
$V_{BAT\_LIM\_HIGH}$	Limiting supply voltage	Battery voltage in buck configuration	-0.2	3.6	V
$V_{PIN\_LIM\_default}$	Limiting voltage on a pin		-0.2	3.6	V

Parameter	Description	Conditions	Min	Max	Unit
T_{STG}	Storage temperature		-50	150	°C

5.2 Recommended Operating Conditions

Table 6: Recommended operating conditions

Parameter	Description	Conditions	Min	Typ	Max	Unit
T_A	Ambient temperature		-40	25	85	°C
$V_{PIN\_VDDIO\_DIO1}$	Voltage on a pin		1.62		3.6	V
$V_{PIN\_VDDIO\_DIO2}$	Voltage on a pin		1.62		3.6	V
$V_{PIN\_VDDIO\_FIO}$	Voltage on a pin		1.62		3.6	V
V_{PIN_VDDD}	Voltage on a pin			1.1		V
V_{PIN_VDDRF}	Voltage on a pin			1.37		V
$V_{PIN\_VDDRF\_PA}$	Voltage on a pin			1.35		V
V_{BAT}	Battery supply voltage		1.9	3.3	3.6	V
$V_{BAT\_HIGH\_BUCK}$	Supply voltage.Battery voltage.	For OTP functionality refer to $V_{BAT\_HIGH\_OTP\_Program/Read}$	1.8		3.6	V
$V_{BAT\_LOW\_BUCK}$	DC-DC or LDO_LOW output		1.1		3.3	V
$V_{BAT\_HIGH\_OTP\_Program}$	Voltage range for OTP programming	Required temperature for programming: -40 °C to 85 °C	2.25		3.6	V
$V_{BAT\_HIGH\_OTP\_Read}$	Voltage range for OTP reading		1.62		3.6	V
$V_{PIN\_default}$	Voltage on a pin		-0.2		VBAT_HIGH+0.2	V

5.3 Crystal Oscillator 40 MHz – Recommended Operating Conditions

Table 7: a_xtal_xtal40M_3095_01 - Recommended operating conditions

Parameter	Description	Conditions	Min	Typ	Max	Unit
$V_{IN\_XTAL40M}$	Input voltage			1.1		V
$f_{XTAL\_XTAL40M}$	Crystal oscillator frequency			40		MHz
$\Delta f_{XTAL\_XTAL40M}$	Crystal frequency tolerance (including aging)		-20		20	ppm
$ESR_{XTAL40M}$	Equivalent series resistance				50	Ω

Parameter	Description	Conditions	Min	Typ	Max	Unit
$C_{L\_XTAL40M}$	Load capacitance		6	8	10	pF

5.4 XTAL32K – Recommended Operating Conditions

Table 8: a_xtal32k_3095_00 (u_a_xtal32k_3095_00) - Recommended operating conditions

Parameter	Description	Conditions	Min	Typ	Max	Unit
$f_{CLK\_EXT\_XTAL32K}$	External clock frequency			32.768		kHz
$\Delta f_{XTAL\_XTAL32K}$	Crystal frequency tolerance (including aging)		-250		250	ppm
$ESR_{XTAL32K}$	Equivalent series resistance			100		kΩ
$C_{L\_XTAL32K}$	Load capacitance			10		pF
$C_{0\_XTAL32K}$	Shunt capacitance			15		pF

5.5 GPADC – DC Characteristics

Table 9: a_adc_gpadc_sar12b1m_3095_00 (u_a_adc_gpadc_sar12b1m_3095_00) - DC characteristics

Parameter	Description	Conditions	Min	Typ	Max	Unit
V_{IN_GPADC}	Input voltage	VDD = 3.3/1.1 V; TA = 25 °C	0		1.4	V

5.6 GPADC – Electrical Performance

Table 10: a_adc_gpadc_sar12b1m_3095_00 (u_a_adc_gpadc_sar12b1m_3095_00) - Electrical performance

Parameter	Description	Conditions	Min	Typ	Max	Unit
SNR_{GPADC}	Signal to noise ratio	VDD = 3.3/1.1 V; TA = 25 °C		55.5		dB
THD_{GPADC}	Total harmonic distortion	VDD = 3.3/1.1 V; TA = 25 °C		65.3		dB
$SFDR_{GPADC}$	Spurious-free dynamic range	VDD = 3.3/1.1 V; TA = 25 °C		67.7		dB

5.7 RST_N Digital I/O – Recommended Operating Conditions

Table 11: a_pads_rstn_3095_00 - Recommended operating conditions

Parameter	Description	Conditions	Min	Typ	Max	Unit
$V_{BAT\_RST\_N}$	Battery supply voltage		1.62		3.6	V
$V_{IH\_RST\_N\_VBAT\_1V8}$	High-level input voltage		1.13			V
$V_{IL\_RST\_N\_VBAT\_1V8}$	Low-level input voltage				0.92	V

Parameter	Description	Conditions	Min	Typ	Max	Unit
$V_{IH\_RST\_N\_VBAT\_3V3}$	High-level input voltage		2.2			V
$V_{IL\_RST\_N\_VBAT\_3V3}$	Low-level input voltage				1.8	V

5.8 GPIO – Recommended Operating Conditions

Table 12: a_pads_digital_io_3095_00 - Recommended operating conditions

Parameter	Description	Conditions	Min	Typ	Max	Unit
V_{BAT_GPIO}	Battery supply voltage		1.62		3.6	V
$V_{IH\_GPIO\_1V8}$	High-level input voltage		1.26		1.8	V
$V_{IL\_GPIO\_1V8}$	Low-level input voltage		0		0.54	V
$V_{IH\_GPIO\_3V3}$	High-level input voltage		2		3.3	V
$V_{IL\_GPIO\_3V3}$	Low-level input voltage		0		0.8	V

5.9 GPIO – DC Characteristics

Table 13: a_pads_digital_io_3095_00 - DC characteristics

Parameter	Description	Conditions	Min	Typ	Max	Unit
$V_{OH\_GPIO\_1V8}$	High-level output voltage		1.2		1.8	V
$V_{OL\_GPIO\_1V8}$	Low-level output voltage		0		0.4	V
$R_{PU\_GPIO\_1V8}$	Pull-up resistance		10		60	kΩ
$R_{PD\_GPIO\_1V8}$	Pull-down resistance		10		60	kΩ
$V_{OH\_GPIO\_3V3}$	High-level output voltage		2.4		3.3	V
$V_{OL\_GPIO\_3V3}$	Low-level output voltage		0		0.4	V
$R_{PU\_GPIO\_3V3}$	Pull-up resistance		10		60	kΩ
$R_{PD\_GPIO\_3V3}$	Pull-down resistance		10		60	kΩ
$I_{IH\_GPIO\_3V3}$	High-level input current		-10	0.1	10	nA
$I_{IL\_GPIO\_3V3}$	Low-level input current		-10	-0.1	10	nA
$I_{IH\_GPIO\_1V8}$	High-level input current		-10	0.1	10	nA
$I_{IL\_GPIO\_1V8}$	Low-level input current		-10	-0.1	10	nA

5.10 RCX Oscillator

Table 14: RCX Oscillator - Timing characteristics

Parameter	Description	Conditions	Min	Typ	Max	Unit
Δf_{RC}	RCX oscillator frequency drift	100 ms time slot		100	500	ppm
$\Delta f_{RC}/\Delta V_{\_VBAT\_HIGH}$	Supply voltage dependency (V_{BAT_HIGH})		-500	80	500	ppm/V
$\Delta f_{RC}/\Delta V_{\_VBAT\_LOW}$	Supply voltage dependency (V_{BAT_LOW})		-500	200	3000	ppm/V
f_{RCX}	RCX oscillator frequency	At target fixed trim setting	13	15	17	kHz
$\Delta f_{RC}/\Delta T_{\_1}$	Temperature dependency	Temperature range -40 °C to 85 °C, RCX_BIAS at preferred value	-125		125	ppm/deg
$\Delta f_{RC}/\Delta T_{\_2}$	Temperature dependency	Temperature range -40 °C to 105 °C, RCX_BIAS at preferred value	-200		200	ppm/deg

5.11 XTAL32MHz Oscillator

Table 15: XTAL32MHz Oscillator - Recommended operating conditions

Parameter	Description	Conditions	Min	Typ	Max	Unit
P_{DRV_MAX}	Maximum drive power	Note 1			100	µW
V_{CLK_EXT}	External clock voltage	In case of external clock source on XTAL32Mp (XTAL32Mm floating or connected to mid-level 0.6 V)		0.9		V
$\varphi_{N\_EXT\_32M}$	Phase noise	f_C = 50 kHz; in case of external clock source			-130	dBc/Hz
Δf_{XTAL_TRIM}	Crystal frequency trim			2		ppm
f_{XTAL_32M}	Crystal oscillator frequency			32		MHz
Δf_{XTAL}	Crystal frequency tolerance	After optional trimming; including aging and temperature drift Note 2	-20		20	ppm
Δf_{XTAL_UNT}	Crystal frequency tolerance	Untrimmed; including aging and temperature drift Note 3	-40		40	ppm
$ESR_{\_1pF}$	Equivalent series resistance	C_0<1pF			200	Ω
$ESR_{\_3pF}$	Equivalent series resistance	C_0<3pF			80	Ω
$ESR_{\_5pF}$	Equivalent series resistance	C_0<5pF			50	Ω
C_L	Load capacitance	No external capacitors are required	4	6	8	pF

Note 1 Select a crystal which can handle a drive level of at least this specification.

Note 2 Using the internal varicaps a wide range of crystals can be trimmed to the required tolerance.

Note 3 Maximum allowed frequency tolerance for compensation by the internal varicap trimming mechanism.

5.12 Wi-Fi Radio

Table 16: r_radio_3095_wifi_rx_2g4_specs_00 (Iradio_RX_2G4) - DC characteristics

Parameter	Description	Conditions	Min	Typ	Max	Unit
$I_{BAT\_RX2G\_No\_Signal}$	Battery supply current	RX: No Signal input, VBAT = 3.3 V, VDDRF = 1.375 V, Ch1, TA = 25 °C		45 Note 1		mA
$I_{BAT\_RX2G\_11b\_1M}$	Battery supply current	RX: Modulated signal input, VBAT = 3.3 V, VDDRF = 1.375 V, Ch1, TA = 25 °C		48 Note 1		mA
$I_{BAT\_RX2G\_11b\_1M}$	Battery supply current	RX: Modulated signal input, VBAT = 3.3 V, VDDRF = 1.375 V, Ch1, TA = 25 °C		58		mA
$I_{BAT\_RX2G\_11n\_MCS7}$	Battery supply current	RX: Modulated signal input, VBAT = 3.3 V, VDDRF = 1.375 V, Ch1, TA = 25 °C		63		mA
$I_{BAT\_RX2G\_11ax\_MCS9}$	Battery supply current	RX: Modulated signal input, VBAT = 3.3 V, VDDRF = 1.375 V, Ch1, TA = 25 °C		64		mA

Note 1 CPU clock = 40 MHz, Low Current Mode for DTIM.

Table 17: r_radio_3095_wifi_rx_2g4_specs_00 (Iradio_RX_2G4) - AC characteristics

Parameter	Description	Conditions	Min	Typ	Max	Unit
$P_{SENS\_RX2G\_11b\_1M}$	Sensitivity (8% PER for 11b rates, 10% PER for 11g/n/ax rates)	802.11b, 1 Mbps DSSS, Ch1		-96.5		dBm
$P_{SENS\_RX2G\_11b\_2M}$	Sensitivity (8% PER for 11b rates, 10% PER for 11g/n/ax rates)	802.11b, 2 Mbps DSSS, Ch1		-93.1		dBm
$P_{SENS\_RX2G\_11b\_11M}$	Sensitivity (8% PER for 11b rates, 10% PER for 11g/n/ax rates)	802.11b, 11 Mbps CKK, Ch1		-86.2		dBm
$P_{SENS\_RX2G\_11g\_6M}$	Sensitivity (8% PER for 11b rates, 10% PER for 11g/n/ax rates)	802.11g, 6 Mbps OFDM, Ch1		-89.3		dBm
$P_{SENS\_RX2G\_11g\_9M}$	Sensitivity (8% PER for 11b rates, 10% PER for 11g/n/ax rates)	802.11g, 9 Mbps OFDM, Ch1		-88.8		dBm
$P_{SENS\_RX2G\_11g\_18M}$	Sensitivity (8% PER for 11b rates, 10% PER for 11g/n/ax rates)	802.11g, 18 Mbps OFDM, Ch1		-85.9		dBm
$P_{SENS\_RX2G\_11g\_36M}$	Sensitivity (8% PER for 11b rates, 10% PER for 11g/n/ax rates)	802.11g, 36 Mbps OFDM, Ch1		-79.6		dBm
$P_{SENS\_RX2G\_11g\_54M}$	Sensitivity (8% PER for 11b rates, 10% PER for 11g/n/ax rates)	802.11g, 54 Mbps OFDM, Ch1		-73.5		dBm
$P_{SENS\_RX2G\_11n\_MCS0}$	Sensitivity (8% PER for 11b rates, 10% PER for 11g/n/ax rates)	802.11n, MCS0, Ch1		-88.7		dBm

Parameter	Description	Conditions	Min	Typ	Max	Unit
$P_{SENS\_RX2G\_11n\_MCS7}$	Sensitivity (8% PER for 11b rates, 10% PER for 11g/n/ax rates)	802.11n, MCS7, Ch1		-70		dBm
$P_{SENS\_RX2G\_11ax\_MCS9}$	Sensitivity (8% PER for 11b rates, 10% PER for 11g/n/ax rates)	802.11ax, MCS9, Ch1		-64.1		dBm
$ACR_{RX2G\_11b\_11M}$	Wanted signal = 2412 MHz/-70 dBm Interferer signal = 2437 MHz Gmode = from AGC	802.11b, 11 Mbps CKK		43.5		dB
$ACR_{RX2G\_11g\_6M}$	Wanted signal = 2412 MHz/-79 dBm Interferer signal = 2437 MHz Gmode = from AGC	802.11g, 6 Mbps OFDM		38.5		dB
$ACR_{RX2G\_11g\_54M}$	Wanted signal = 2412 MHz/-62 dBm Interferer signal = 2437 MHz Gmode = from AGC	802.11g, 54 Mbps OFDM		24.5		dB
$ACR_{RX2G\_11n\_MCS0}$	Wanted signal = 2412 MHz/-79 dBm Interferer signal for ACR= 2437 MHz Gmode = from AGC	802.11n, MCS0		31		dB
$ACR_{RX2G\_11n\_MCS7}$	Wanted signal = 2412 MHz/-61 dBm Interferer signal for ACR = 2437 MHz Gmode = from AGC	802.11n, MCS7		19.5		dB
$ACR_{RX2G\_11ax\_MCS9}$	Wanted signal = 2412 MHz/-54 dBm Interferer signal for ACR = 2432 MHz Gmode = from AGC	802.11ax, MCS9		-3.5		dB

Table 18: r_radio_3095_wifi_rx_5g0_specs_00 (lradio_RX_5G0) - DC characteristics

Parameter	Description	Conditions	Min	Typ	Max	Unit
$I_{BAT\_RX5G\_No\_Signal}$	Battery supply current	RX: Ch36, No Signal input, VBAT = 3.3 V, VDDRF = 1.375 V, TA = 25 °C		53 Note 1		mA
$I_{BAT\_RX5G\_11a\_6M}$	Battery supply current	RX: Ch36, Modulated signal input, VBAT = 3.3 V, VDDRF = 1.375 V, TA = 25 °C		57 Note 1		mA
$I_{BAT\_RX5G\_11a\_6M}$	Battery supply current	RX: Ch36, Modulated signal input, VBAT = 3.3 V, VDDRF = 1.375 V, TA = 25 °C		68		mA
$I_{BAT\_RX5G\_11ax\_MCS7}$	Battery supply current	RX: Ch36, Modulated signal input, VBAT = 3.3 V, VDDRF = 1.375 V, TA = 25 °C		73		mA

Parameter	Description	Conditions	Min	Typ	Max	Unit
$I_{BAT\_RX5G\_No\_Signal}$	Battery supply current	RX: Ch165, No Signal input, VBAT = 3.3 V, VDDRF = 1.375 V, TA = 25 °C		53 Note 1		mA
$I_{BAT\_RX5G\_11a\_6M}$	Battery supply current	RX: Ch165, Modulated signal input, VBAT = 3.3 V, VDDRF = 1.375 V, TA = 25 °C		58 Note 1		mA
$I_{BAT\_RX5G\_11a\_6M}$	Battery supply current	RX: Ch165, Modulated signal input, VBAT = 3.3 V, VDDRF = 1.375 V, TA = 25 °C		69		mA
$I_{BAT\_RX5G\_11ax\_MCS7}$	Battery supply current	RX: Ch165, Modulated signal input, VBAT = 3.3 V, VDDRF = 1.375 V, TA = 25 °C		72		mA

Note 1 CPU clock = 40 MHz, Low Current Mode for DTIM.

Table 19: r_radio_3095_wifi_rx_5g0_specs_00 (lradio_RX_5G0) - AC characteristics

Parameter	Description	Conditions	Min	Typ	Max	Unit
$P_{SENS\_RX5G\_11a\_6M}$	PSDU length: 1000 octets	802.11a, 6 Mbps OFDM, Ch36		-89.2		dBm
$P_{SENS\_RX5G\_11a\_9M}$	PSDU length: 1000 octets	802.11a, 9 Mbps OFDM, Ch36		-89.1		dBm
$P_{SENS\_RX5G\_11a\_18M}$	PSDU length: 1000 octets	802.11a, 18 Mbps OFDM , Ch36		-86.4		dBm
$P_{SENS\_RX5G\_11a\_36M}$	PSDU length: 1000 octets	802.11a, 36 Mbps OFDM, Ch36		-80.1		dBm
$P_{SENS\_RX5G\_11a\_54M}$	PSDU length: 1000 octets	802.11a, 54 Mbps OFDM, Ch36		-73.4		dBm
$P_{SENS\_RX5G\_11n\_MCS0}$	PSDU length: 4096 octet; LGI; non-STMC	802.11n, MCS0, Ch36		-88.9		dBm
$P_{SENS\_RX5G\_11n\_MCS7}$	PSDU length: 4096 octet; LGI; non-STMC	802.11n, MCS7, Ch36		-69.6		dBm
$P_{SENS\_RX5G\_11ac\_MCS7}$	PSDU length: 4096 octet; LGI; non-STMC	802.11ac, MCS7, Ch36		-69.3		dBm
$P_{SENS\_RX5G\_11ax\_MCS7}$	PSDU length: 4096 octet; LGI; non-STMC	802.11ax, MCS7, Ch36		-70.1		dBm
$ACR_{RX5G\_11a\_6M}$	Wanted signal = 5500 MHz/-79 dBm Interferer signal for ACR = 5520 MHz Gmode = from AGC	802.11a, 6 Mbps OFDM		23.5		dB

Parameter	Description	Conditions	Min	Typ	Max	Unit
$ACR_{RX5G\_11a\_54M}$	Wanted signal = 5500 MHz/-62 dBm Interferer signal for ACR = 5520 MHz Gmode = from AGC	802.11a, 54 Mbps OFDM		8		dB
$ACR_{RX5G\_11n\_MCS0}$	Wanted signal = 5500 MHz/-79 dBm Interferer signal for ACR = 5520 MHz Gmode = from AGC	802.11n, MCS0		21.4		dB
$ACR_{RX5G\_11n\_MCS7}$	Wanted signal = 5500 MHz/-61 dBm Interferer signal for ACR = 5520 MHz Gmode = from AGC	802.11n, MCS7		5.4		dB
$ACR_{RX5G\_11ax\_MCS7}$	Wanted signal = 5500 MHz/-61 dBm Interferer signal for ACR = 5520 MHz Gmode = from AGC	802.11ax, MCS7		4.2		dB

Table 20: r_radio_3095_wifi_tx_2g4_specs_00 (Iradio_TX_2G4) - DC characteristics

Parameter	Description	Conditions	Min	Typ	Max	Unit
$I_{BAT\_TX\_2G\_11b\_1M}$	Battery supply current measured for EVM and SEM compliance	TX: VBAT = 3.3 V, TA = 25 ºC, Ch1, Pout = 17.2 dBm		296		mA
$I_{BAT\_TX\_2G\_11g\_6M}$	Battery supply current measured for EVM and SEM compliance	TX: VBAT = 3.3 V, TA = 25 ºC, Ch1, Pout = 18.2 dBm		350		mA
$I_{BAT\_TX\_2G\_11b\_1M}$	Battery supply current measured for EVM compliance	TX: VBAT = 3.3 V, TA = 25 ºC, Ch1, Pout = 20.2 dBm		498		mA
$I_{BAT\_TX\_2G\_11g\_6M}$	Battery supply current measured for EVM compliance	TX: VBAT = 3.3 V, TA = 25 ºC, Ch1, Pout = 20.2 dBm		517		mA
$I_{BAT\_TX\_2G\_11n\_MCS7}$	Battery supply current measured for EVM compliance	TX: VBAT = 3.3 V, TA = 25 ºC, Ch1, Pout = 14.7 dBm		217		mA
$I_{BAT\_TX\_2G\_11ax\_MCS9}$	Battery supply current measured for EVM compliance	TX: VBAT = 3.3 V, TA = 25 ºC, Ch1, Pout = 11.2 dBm		162		mA

Table 21: r_radio_3095_wifi_tx_2g4_specs_00 (Iradio_TX_2G4) - AC characteristics

Parameter	Description	Conditions	Min	Typ	Max	Unit
$P_{O(MAX)\_11b\_1M}$	Maximum Output Power measured for EVM and SEM compliance	802.11b, 1 Mbps DSSS, Ch1		17.4		dBm
$P_{O(MAX)\_11b\_2M}$	Maximum Output Power measured for EVM and SEM compliance	802.11b, 2 Mbps DSSS, Ch1		17.4		dBm

Parameter	Description	Conditions	Min	Typ	Max	Unit
$P_{O(MAX)\_11b\_5M5}$	Maximum Output Power measured for EVM and SEM compliance	802.11b, 5.5 Mbps CCK, Ch1		18.4		dBm
$P_{O(MAX)\_11b\_11M}$	Maximum Output Power measured for EVM and SEM compliance	802.11b,11 Mbps CCK, Ch1		18.3		dBm
$P_{O(MAX)\_11g\_6M}$	Maximum Output Power measured for EVM and SEM compliance	802.11g, 6 Mbps OFDM, Ch1		18.3		dBm
$P_{O(MAX)\_11g\_54M}$	Maximum Output Power measured for EVM and SEM compliance	802.11g, 54 Mbps OFDM, Ch1		15.5		dBm
$P_{O(MAX)\_11n\_MCS0}$	Maximum Output Power measured for EVM and SEM compliance	802.11n, MCS0, Ch1		16		dBm
$P_{O(MAX)\_11n\_MCS7}$	Maximum Output Power measured for EVM and SEM compliance	802.11n, MCS7, Ch1		15.1		dBm
$P_{O(MAX)\_11ax\_MCS9}$	Maximum Output Power measured for EVM and SEM compliance	802.11ax, MCS9, Ch1		11.7		dBm
$P_{O(MAX)\_11b\_1M}$	Maximum Output Power measured for EVM compliance	802.11b, 1 Mbps DSSS, Ch1		20.3		dBm
$P_{O(MAX)\_11b\_2M}$	Maximum Output Power measured for EVM compliance	802.11b, 2 Mbps DSSS, Ch1		20.3		dBm
$P_{O(MAX)\_11b\_5M5}$	Maximum Output Power measured for EVM compliance	802.11b, 5.5 Mbps CCK, Ch1		20.5		dBm
$P_{O(MAX)\_11b\_11M}$	Maximum Output Power measured for EVM compliance	802.11b,11 Mbps CCK, Ch1		20.3		dBm
$P_{O(MAX)\_11g\_6M}$	Maximum Output Power measured for EVM compliance	802.11g, 6 Mbps OFDM, Ch1		20.3		dBm
$P_{O(MAX)\_11n\_MCS0}$	Maximum Output Power measured for EVM compliance	802.11n, MCS0, Ch1		20.3		dBm

Table 22: r_radio_3095_wifi_tx_5g0_00 (Iradio_TX_5G0) - DC characteristics

Parameter	Description	Conditions	Min	Typ	Max	Unit
$I_{BAT\_TX5G\_11a\_6M}$	Battery supply current measured for EVM and SEM compliance	TX: Ch36, Modulated signal output, VBAT = 3.3 V, TA = 25 °C, Pout = 15.6 dBm		384		mA
$I_{BAT\_TX5G\_11a\_6M}$	Battery supply current measured for EVM compliance	TX: Ch36, Modulated signal output, VBAT = 3.3 V, TA = 25 °C, Pout = 19.1 dBm		725		mA
$I_{BAT\_TX5G\_11ax\_MCS7}$	Battery supply current measured for EVM compliance	TX: Ch36, Modulated signal output, VBAT = 3.3 V, TA = 25 °C, Pout = 12.1 dBm		250		mA
$I_{BAT\_TX5G\_11a\_6M}$	Battery supply current measured for EVM and SEM compliance	TX: Ch165, Modulated signal output, VBAT = 3.3 V, TA = 25 °C, Pout = 15.6 dBm		369		mA

Parameter	Description	Conditions	Min	Typ	Max	Unit
$I_{BAT\_TX5G\_11a\_6M}$	Battery supply current measured for EVM compliance	TX: Ch165, Modulated signal output, VBAT = 3.3 V, TA = 25 °C, Pout = 18.6 dBm		755		mA
$I_{BAT\_TX5G\_11ax\_MCS7}$	Battery supply current measured for EVM compliance	TX: Ch165, Modulated signal output, VBAT = 3.3 V, TA = 25 °C, Pout = 7.1 dBm		188		mA

Table 23: r_radio_3095_wifi_tx_5g0_00 (Iradio_TX_5G0) - AC characteristics

Parameter	Description	Conditions	Min	Typ	Max	Unit
$P_{O(MAX)\_11a\_6M}$	Maximum Output Power measured for EVM and SEM compliance	802.11a, 6 Mbps OFDM, Ch36		16		dBm
$P_{O(MAX)\_11a\_54M}$	Maximum Output Power measured for EVM and SEM compliance	802.11a, 54 Mbps OFDM, Ch36		14.1		dBm
$P_{O(MAX)\_11n\_MCS0}$	Maximum Output Power measured for EVM and SEM compliance	802.11n, MCS0, Ch36		15.3		dBm
$P_{O(MAX)\_11n\_MCS7}$	Maximum Output Power measured for EVM and SEM compliance	802.11n, MCS7, Ch36		12.2		dBm
$P_{O(MAX)\_11ac\_MCS7}$	Maximum Output Power measured for EVM and SEM compliance	802.11ac, MCS7, Ch36		12.2		dBm
$P_{O(MAX)\_11ax\_MCS7}$	Maximum Output Power measured for EVM and SEM compliance	802.11ax, MCS7, Ch36		12.1		dBm
$P_{O(MAX)\_11a\_6M}$	Maximum Output Power measured for EVM compliance	802.11a, 6 Mbps OFDM, Ch36		19.4		dBm
$P_{O(MAX)\_11n\_MCS0}$	Maximum Output Power measured for EVM compliance	802.11n, MCS0, Ch36		19.6		dBm

5.13 Bluetooth LE Radio

Table 24: Bluetooth LE Radio 1M - Recommended operating conditions

Parameter	Description	Conditions	Min	Typ	Max	Unit
f_{OPER}	Operating frequency		2400		2483.5	MHz
N_{CH}	Number of channels			40		1
f_{CH}	Channel frequency	K = 0 to 39		2402+K*2		MHz

Table 25: Bluetooth LE Radio 1M - DC characteristics

Parameter	Description	Conditions	Min	Typ	Max	Unit
$I_{BAT\_RF\_RX}$	Current at V_{BAT_LOW} = 1.1 V	Radio receiver and synthesizer active; T_A = 25 °C		4.3		mA

Parameter	Description	Conditions	Min	Typ	Max	Unit
$I_{BAT\_RF\_TX\_12}$	Current at V_{BAT_LOW} = 1.1 V	Radio transmitter and synthesizer active; power setting = 12; P_{OUT} = 2.5 dBm; T_A = 25 °C		7.9		mA
$I_{BAT\_RF\_TX\_9}$	Current at V_{BAT_LOW} = 1.1 V	Radio transmitter and synthesizer active; power setting = 9; P_{OUT} = 0 dBm; T_A = 25 °C		6.9		mA
$I_{BAT\_RF\_TX\_6}$	Current at V_{BAT_LOW} = 1.1 V	Radio transmitter and synthesizer active; power setting = 6; P_{OUT} = -3.5 dBm; T_A = 25 °C		5.7		mA

Table 26: Bluetooth LE Radio 1M - AC characteristics

Parameter	Description	Conditions	Min	Typ	Max	Unit
P_{SENS_CLEAN}	Sensitivity level	Dirty Transmitter disabled; DC-DC converter disabled; PER = 30.8 % Note 1		-94		dBm
P_{SENS_EPKT}	Sensitivity level	Extended packet size (255 octets) Note 1		-90.5		dBm
P_{SENS}	Sensitivity level	Normal Operating Conditions; DC-DC converter disabled; PER = 30.8 % Note 1		-93		dBm
$P_{SENS\_EPKT\_CLEAN}$	Sensitivity level	Dirty Transmitter disabled; DC-DC converter disabled; Extended packet size (255 octets) Note 1		-91.5		dBm
P_{I_MAX}	Maximum input power level	DC-DC converter disabled; PER = 30.8 % Note 1		0		dBm
P_{INT_IMD}	Intermodulation distortion interferer power level	worst-case interferer level @ f_1, f_2 with $2*f_1 - f_2 = f_0$, $\|f_1 - f_2\|$ = n MHz and n = 3, 4, 5; P_{WANTED} = -64 dBm @ f_0; PER = 30.8 % Note 1		-20.5		dBm
CIR_0	Carrier to interferer ratio	n = 0; interferer @ $f_1 = f_0 + n*1$ MHz; Note 1		7.5		dB
CIR_1	Carrier to interferer ratio	n = ±1; interferer @ $f_1 = f_0 + n*1$ MHz; Note 1		-1.5		dB

Parameter	Description	Conditions	Min	Typ	Max	Unit
CIR_{P2}	Carrier to interferer ratio	n = +2 (image frequency); interferer @ $f_I = f_0 + n*1$ MHz; Note 1		-27		dB
CIR_{M2}	Carrier to interferer ratio	n = -2; interferer @ $f_I = f_0 + n*1$ MHz; Note 1		-31		dB
CIR_{P3}	Carrier to interferer ratio	n = +3 (image frequency + 1 MHz); interferer @ $f_I = f_0 + n*1$ MHz; Note 1		-37.5		dB
CIR_{M3}	Carrier to interferer ratio	n = -3; interferer @ $f_I = f_0 + n*1$ MHz; Note 1		-43		dB
CIR_4	Carrier to interferer ratio	|n| ≥ 4 (any other BLE channel); interferer @ $f_I = f_0 + n*1$ MHz; Note 1		-41.5		dB
P_{BL_I}	Blocker power level	30 MHz ≤ f_{BL} ≤ 2000 MHz; P_{WANTED} = -67 dBm Note 1		5		dBm
P_{BL_II}	Blocker power level Note 2	2003 MHz ≤ f_{BL} ≤ 2399 MHz; P_{WANTED} = -67 dBm Note 1		0		dBm
P_{BL_III}	Blocker power level	2484 MHz ≤ f_{BL} ≤ 2997 MHz; P_{WANTED} = -67 dBm Note 1		0		dBm
P_{BL_IV}	Blocker power level	3000 MHz ≤ f_{BL} ≤ 12.75 GHz; P_{WANTED} = -67 dBm Note 1		5		dBm
L_{ACC_RSSI}	Level accuracy	Input power range -90 dBm to -20 dBm		3		dB
L_{RES_RSSI}	Level resolution	Input power range -90 dBm to -20 dBm		0.5		dB/LSB
ACP_{2M}	Adjacent channel power level	f_{OFS} = 2 MHz; Note 1		-46		dBm
ACP_{3M}	Adjacent channel power level	f_{OFS} ≥ 3 MHz; Note 1		-53		dBm
P_{O_12}	Output power level	RF_ATTR_REG[PA_POWER_SETTING]= 12		2.5		dBm
P_{O_11}	Output power level	RF_ATTR_REG[PA_POWER_SETTING]= 11		1.5		dBm
P_{O_10}	Output power level	RF_ATTR_REG[PA_POWER_SETTING]= 10		1		dBm

Parameter	Description	Conditions	Min	Typ	Max	Unit
P_{O_09}	Output power level	RF_ATTR_REG[PA_POWER_SETTING]= 9		0		dBm
P_{O_08}	Output power level	RF_ATTR_REG[PA_POWER_SETTING]= 8		-1		dBm
P_{O_07}	Output power level	RF_ATTR_REG[PA_POWER_SETTING]= 7		-2		dBm
P_{O_06}	Output power level	RF_ATTR_REG[PA_POWER_SETTING]= 6		-3.5		dBm
P_{O_05}	Output power level	RF_ATTR_REG[PA_POWER_SETTING]= 5		-5		dBm
P_{O_04}	Output power level	RF_ATTR_REG[PA_POWER_SETTING]= 4		-7		dBm
P_{O_03}	Output power level	RF_ATTR_REG[PA_POWER_SETTING]= 3		-10		dBm
P_{O_02}	Output power level	RF_ATTR_REG[PA_POWER_SETTING]= 2		-13.5		dBm
P_{O_01}	Output power level	RF_ATTR_REG[PA_POWER_SETTING]= 1		-19.5		dBm

Note 1 Measured according to Bluetooth® Low Energy Test Specification RF-PHY.TS/5.1.0

Note 2 Frequencies close to the ISM band can show slightly worse performance

Table 27: Bluetooth LE Radio 1M - Timing characteristics

Parameter	Description	Conditions	Min	Typ	Max	Unit
t_{HOLD}	Hold time of radio ldo's	Temperature from -40 °C to 55 °C	500			ms
t_{HOLD_EOC}	Hold time of the radio ldo's	Temperature from -40 °C to 85 °C	10			ms

6. System Overview

6.1 Core System Overview

The Wi-Fi subsystem contains the following components:

- **Arm® Cortex®-M33 CPU**

  The processor is used to run customer applications along with a full Wi-Fi stack (TCP/IP). The M33 provides 1.45 dMIPS/MHz (232 dMIPS at 160 MHz) and has a powerful 32 kB cache controller with 4-way associativity and 16-byte cache line size. Code can be executed from an external Flash device via the secure XIP cache controller, from the internal system or retention memory and from the internal ROM. This gives a high level of flexibility to operate in very low power scenarios.

  Depending on the application, the processor can be clocked by the internal PLL configurable from 2.5 MHz up to 160 MHz.

- **ROM**

  The internal 256 kB of ROM contains the secure boot loader as well as various library functions such as security and networking which can be used by the firmware to help reduce code size for applications running from internal SRAM.

- **OTP**

  The One-Time Programmable (OTP) memory is 2 kB in size and consists of two regions. A 256-byte secure region which is used to store sensitive keys used by the crypto subsystem and a 1792-byte general purpose region which can be used by user applications. Various information such as trim values for the clocks and Wi-Fi interface, MAC addresses, and boot configuration scripts are stored in the general-purpose region during production testing. The remaining OTP is available for use by user applications.

- **System Memory**

  The system memory is 704 kB of internal SRAM which is primarily used as data memory by the software. It also provides the ability to load and run code for implementing very low power applications. To achieve optimal power consumption, the system memory consists of three memory blocks (two blocks of 256 kB and one block of 192 kB) which can be independently powered off or placed into a low power content retention state. The Cortex M33 can access the system memory at 160 MHz with zero wait cycles.

- **Retention Memory**

  The retention memory is a 64 kB block of memory that is retained during the low power operating states of the system. This memory block includes data for maintaining the state of the Wi-Fi interface plus code which can execute when waking up to perform simple Wi-Fi tasks such as DTIM.

- **Octa/Quad SPI Controller (OQSPI)**

  The **OQSPI** controller interfaces to external Quad or Octa Flash devices which store the applications code and data. The controller provides a zero-overhead secure eXecute In Place (XiP) interface which decrypts the code/data on the fly at up to 80 MHz. This allows for the code/data to be stored securely in Flash and will only be accessible internally to the device. The OQSPI interface supports Flash devices of up to 64 MB.

- **Quad SPI Controller (QSPIC)**

  The QSPI controller can interface with an external non secure Flash or PSRAM. For increased read/write performance of the PSRAM, an 8 kB data cache with write-back capabilities can be enabled. The QSPI interface supports Flash or PSRAM devices of up to 64 MB.

- **Crypto Subsystem**

  The crypto subsystem provides hardware acceleration of symmetric algorithms (AES 128/192/256, ChaCha20), asymmetric algorithms (RSA, ECC), hash and HMAC algorithms (SHA-1, SHA224/256), and plus support for accelerating specific operations such as key derivation and secure signature processing. There is also a hardware based True Random Number Generator (TRNG) included for secure key generation and support for secure life cycle management.

- **UART/UART2/UART3**

  The UART interfaces are asynchronous serial interfaces with hardware flow control. The UART interfaces support both RS-232 and RS-485 physical protocols at data rates up to 5 Mbaud.

- **SPI/SPI2**

  The SPI interfaces are serial peripheral interfaces with master and slave capability for connection to SPI devices in master mode or being connected to by a host MCU in slave mode. They have separate RX/TX FIFOs (32 B) and support SPI clock rates up to 48 MHz.

- **I2C/I2C2**

  The I2C interfaces support master and slave capability with clock stretching and are used for communicating to sensors and/or a host MCU. Each controller includes a 32 locations deep FIFO (8-bits Rx, 9-bits Tx). They support slow, fast, and high-speed modes up to 3.4 Mbps.

- **Digital Audio Interface (I2S and PDM)**

  The digital audio interface (DAI) consists of both an I2S master/slave interface and a Digital MIC interface supporting PDM for higher quality digital microphone input plus an asynchronous Sampler Rate Converter (SRC) used to convert the input or output data to the required sample rate. A Fractional PLL (FPLL) generates the base frequency for the audio subsystem.

  The I2S interface supports either master or slave operation for stereo audio with two 32-bit channels at sample rates of 8 kHz, 12 kHz, 16 kHz, 24 kHz, 32 kHz, 44.1 kHz, and 48 kHz.

  The Pulse-density Modulation (PDM) interface supports either master or slave operation for stereo audio sample rates of 24 kHz, 32 kHz, or 48 kHz.

- **Analog to Digital Converter (ADC)**

  The ADC is a general purpose 4 channel 12-bit analog to digital converter which can be used for voltage measurement or audio sampling. It supports a sampling rate of up to 1 Msps.

- **General Purpose Timers/Pulse Width Modulators**

  There are 8 general purpose 32-bit timers which support PWM capabilities. These timers all support PWM generation, one capture channel to save a snapshot of the timer, up/down counting with free-running mode, selectable clock source and one-shot pulse generation with configurable width and edge detection counter modes. All timers support OneShot mode and automatic switching from OneShot mode to Counter mode.

- **Real Time Clock (RTC)**

  The Real Time Clock is a core feature of the Wi-Fi subsystem which provides a stable counter that is maintained during low power states such that it can accurately manage the reset and sleep states required to support low power Wi-Fi operation.

  The RTC can be clocked with a 32.768 kHz crystal or from a calibrated internal 32 kHz RC oscillator. The latter saves component costs.

  To manage sleep modes, the RTC coordinates the power management of various internal blocks and sequences them as necessary to enter and exit the various low power modes of operation. Wake-up from the sleep modes can be accomplished through a predefined timer event or through external events from various GPIO pins.

- **Watchdog Timer**

  The watchdog timer provides a mechanism to protect the system against possible system malfunctions. If there is either a SW error or a hardware configuration which disrupts the normal execution of code, the watchdog will trigger and perform a reset of the system.

- **DMA**

  Two DMA controllers are included which are designed to improve performance and offload the CPU from moving data through the system.

  The General Purpose DMA provides 16 channels which can perform memory-memory or memory-peripheral data transfers. The Fast DMA is specialized for transferring data between memory and the Wi-Fi interface.

- **Debug Interface (SWD)**

  A standard Serial Wire Debug (SWD) is provided to allow debugging of user applications during the development phase of a product. Once development is complete and the device is set to secure mode and the SWD interface is disabled to protect the device from being tampered with.

- **GPIO and Programmable Pin Assignment**

  By default, all digital I/O pins for the Wi-Fi subsystem are defined as GPIOs to give a total of 28 configurable I/Os. These can be configured by software to be mapped to any of the supported digital peripherals. High performance interfaces such as OQSPI, QSPI, SDIO and eMMC have fixed mapping whereas all other interfaces can be mapped to any pin through the programmable pin assignment (PPA) block.

- **SD/eMMC Host Controller**

  The SD/MMC Host Controller supports up to 80 MHz clock output in 1-bit, 4-bit, or 8-bit data bus mode. In 4-bit mode, two SD/SDIO/MMC 4.41 cards can be supported and one SD card operating at 1.8 V.

  The SD/MMC Host Controller is compliant with the following features:

  - Secure Digital (SD) memory version 3.0 and version 3.01
  - Secure Digital I/O (SDIO) version 3.0
  - Multimedia Cards (MMC version 4.41, eMMC version 4.5 and version 4.51).

- **SDIO Device Controller**

  The SDIO device controller is compliant with version 3.00 of the SD specification. It supports up to 80 MHz operation in 1-bit, 4-bit SD bus mode, and SPI Bus mode. This is used to provides a high bandwidth interface to a host MCU or AP for high-speed communications of commands and Wi-Fi date.

- **Wi-Fi Subsystem**

  The Wi-Fi subsystem consists of a 2.4/5 GHz radio, PHY, and MAC supporting the following features:

  - Wi-Fi 802.11a/b/g/n/ac/ax in 2.4 GHz and 5 GHz bands
  - 802.11a/b/g/n PHY/MAC 1x1 SISO
  - Wi-Fi 6 802.11ax PHY/MAC 1x1 SISO
  - 20 MHz channels for 2.4 GHz (802.11b/g/n/ax)
  - 20 MHz channels for 5 GHz (802.11a/n/ac/ax)
  - Compatible to the 802.11ac infrastructure in the 5 GHz band
    - MIMO 1x1
    - Station mode would support 5 GHz 802.11n mode compatible with 802.11ac.
  - Support Dynamic Frequency Selection in 20 MHz mode for all 5 GHz DFS bands in all regulatory domains
  - Wi-Fi 6 802.11ax IoT features such as TWT
  - Adjustable transmit power (1 dB steps) where the Access Point (AP) helps set the Station (STA) output power based on signal and noise conditions
  - Bluetooth Coexistence feature to coordinate sharing of the 2.4 GHz band between three devices.

- **Bluetooth® Coexistence**

  The Bluetooth® coexistence interface allows for sharing of the 2.4 GHz band between Wi-Fi and a companion Bluetooth device such that these devices minimize interference with each other.

  To support smart home systems which may also require coexistence with Zigbee or Thread devices, the coexistence interface provides enhanced features to coordinate the sharing of the 2.4 GHz band between Wi-Fi and two companion devices such as Bluetooth® LE and Thread.

- **Sleep Modes**

  There are several sleep modes supported by Wi-Fi subsystem so that the lowest possible power can be obtained while maintaining the Wi-Fi connection and performing Wi-Fi communications.

  The low power modes are defined as follows:

  - **Sleep 1 – Deep Sleep / Reset State (~0.3 µA)**

    In this state, power is applied (VBAT), but all internal circuitry is power off such that when reset is de-asserted, the device can start without waiting for the power to stabilize.

  - **Sleep 2 – Low Power Standby (~2.5 µA)**

    In this state, all power domains are off except for the RTC and wake-up pins such that a timer or external event can trigger the device to wake up from sleep.

  - **Sleep 3 – Low Power Retention (~3.7 µA)**

    In this state, all power domains are off except for the RTC, wake-up pins, and the retention memory. The retention memory keeps system information such as the Wi-Fi state so that a fast wake-up can occur to do simple network management such as DTIM processing. If required, a full system wake-up can also be triggered.

  - **Sleep 4 – Tickless Idle State**
    Tickless idle state is like retention mode, but it also maintains up to 704 kB of SRAM in a low power state which allows for the full application to maintain its state and wake up without requiring full software

reinitialization (in DPM mode retention memory is only 64kB). The MAC hardware can also be kept alive with a 32 kHz clock to maintain the timers and allow for faster wake-up time depending on power/performance requirements.

Table 28. Tickless Idle state

Sleeping Mode	Sate	Typical [uA]	Notes
DPM	DTIM 1	850	Depending on environment and associated AP
	DTIM 3	350	
	DTIM 6	180	
	DTIM 10	110	
Sleep 4	DTIM 1	3500	
	DTIM 3	1500	
	DTIM 6	700	
	DTIM 10	500	

- **Sensor Monitoring Mode**

  In sensor monitoring mode, the Wi-Fi subsystem is disabled and the Cortex-M33 runs at a very low clock executing code from internal SRAM allowing for a very low power operating mode where simple activities such as monitoring sensors and logging data can be performed.

6.2 Dual Band Wi-Fi Subsystem

The Wi-Fi subsystem integrates an ultra-low power Arm® Cortex®-M33 system processor with a dual band 802.11a/b/g/n/ax Wi-Fi core, memory, flexible peripheral interfaces, and power management functions.

Extremely low-power operation is accomplished by dynamically disabling elements which are not in use, thus allowing a near zero level of power consumption when not actively transmitting or receiving data. Such low-power operation can extend the battery life up to a year or more depending on the application.

Figure 7 shows an overview of the hardware features of the Wi-Fi subsystem.

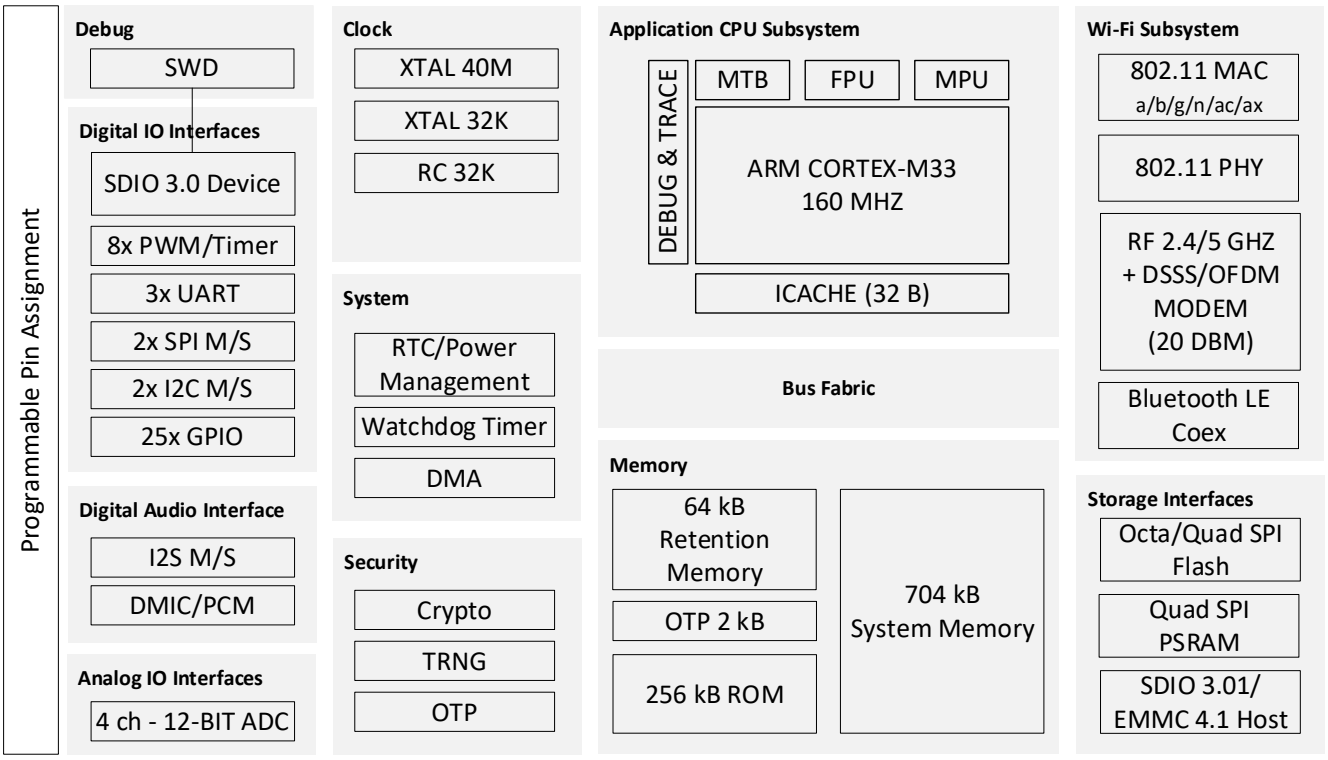


Figure 7. Hardware block diagram

Figure 8 shows an overview of the software features of the Wi-Fi subsystem.

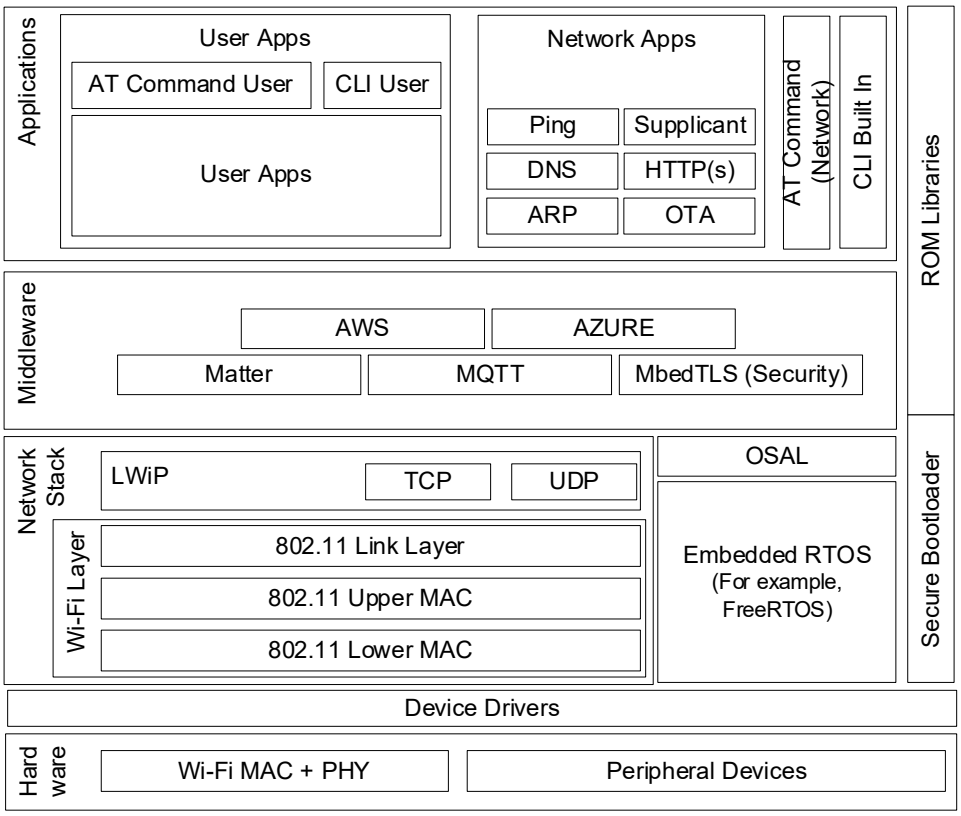


Figure 8. Software system diagram

6.3 Ultra-Low Power Bluetooth Subsystem

Bluetooth® LE subsystem is an ultra-low power subsystem integrating a 2.4 GHz transceiver and an Arm® Cortex®-M0+ microcontroller with a RAM of 48 kB and a One-Time Programmable (OTP) memory of 32 kB.

The radio transceiver, the baseband processor, and the qualified Bluetooth low energy stack are fully compliant with the Bluetooth Low Energy 5.1 standard. Bluetooth LE subsystem has dedicated hardware for the Link Layer implementation of Bluetooth LE and interface controllers for enhanced connectivity capabilities.

The Bluetooth LE 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 Bluetooth SIG as well as custom profiles are supported.

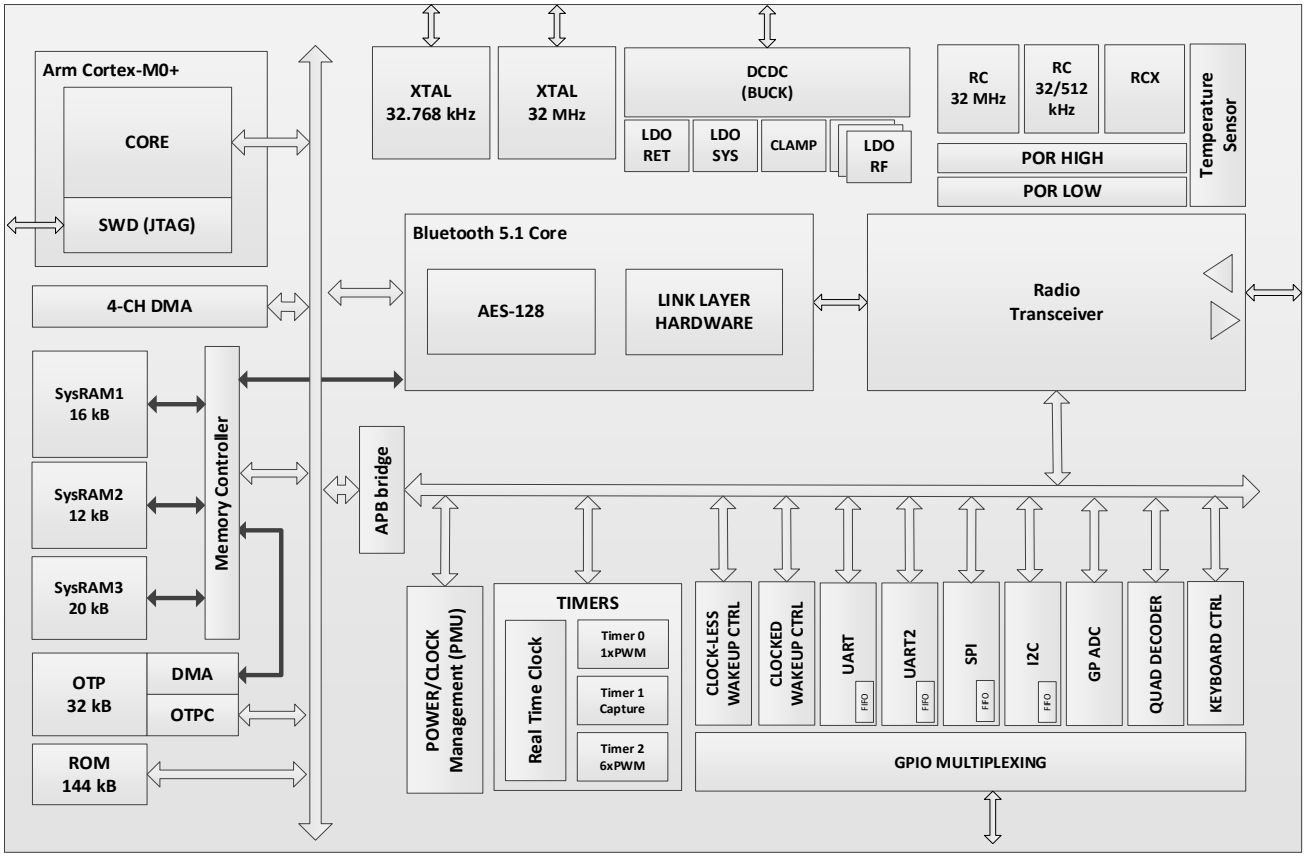


Figure 9. RA6W2 Bluetooth LE subsystem block diagram

7. Core System

7.1 Arm® Cortex®-M33

7.1.1 Introduction

The Arm® Cortex®-M33 is a small and highly energy efficient processor that is intended for microcontroller and deeply embedded applications. The processor is based on the Armv8-M architecture and is primarily used in systems where security is an important consideration.

The Cortex-M33 supports low-power operations through the Wait for Event (WFE) or Wait for Interrupt (WFI) functions which stop the CPU clock during times of inactivity. The system can also be placed into Extended Sleep mode by stopping the system clock and clock gating various features including the CPU. The state of the system and CPU are maintained in retention memory during extended sleep and is restored by software upon wake-up.

The register descriptions for NVIC, the System Control Block (SCB), and the System Timer (SysTick) of the Arm Cortex-M33 can be found on the Arm website.

Feature

- 160 MHz, 3-stage pipeline
- An in-order issue pipeline
- Armv8, Thumb-2 technology, Little endian
- Floating Point Unit (FPU) supporting single precision arithmetic
  - Combined multiply-add instructions for increased precision (Fused MAC)
  - Hardware support for conversion, addition, subtraction, multiplication with optional accumulation, division, and square root
  - Hardware support for denormal and all IEEE Standard 754-2008 rounding modes
  - 32x32-bit single-precision registers or 16x64-bit double-precision registers
  - Lazy floating-point context save
- DSP/SIMD instructions
  - Single-cycle 16/32-bit MAC
  - Single-cycle dual 16-bit MAC
  - 8/16-bit SIMD arithmetic
- Support for exception-continuable instructions
- Support for SWD and Micro Trace Buffer (MTB)
- Nested Vectored Interrupt Controller (NVIC).

7.1.2 Interrupts

The NVIC is closely integrated with the core to achieve low-latency interrupt processing.

Functions of the NVIC include:

- Configurable levels of interrupt priority from 8 to 256.
- Dynamic reprioritization of interrupts.
- Priority grouping. This enables selection of preempting interrupt levels and non-preempting interrupt levels.
- Support for tail-chaining and late arrival of interrupts. This enables back-to-back interrupt processing without the overhead of state saving and restoration between interrupts.

Table 29: Interrupt list

IRQ #	Name	Description
0	DMA	General Purpose DMA interrupt request.
1	FDMA	Fast DMA interrupt request.
2	UART	UART interrupt request.
3	UART2	UART2 interrupt request.
4	UART3	UART3 interrupt request.
5	I2C	I2C interrupt request.
6	I2C2	I2C2 interrupt request.
7	SPI	SPI interrupt request.
8	SPI2	SPI2 interrupt request.
9	DAI_TX	DAI TX interrupt request.
10	DAI_RX	DAI RX interrupt request.
11	SRC_IN	SRC input interrupt request.
12	SRC_OUT	SRC output interrupt request.
13	SDIO	SDIO interrupt request.
14	SDIO_WKUP	SDEMMC interrupt request.
15	TDES_CBC	TDES interrupt request.
16	PSK_SHA1	PSK_SHA1 interrupt request.
18	CLKCAL	Clock calibration interrupt request.
19	TIMER	TIMER interrupt request.
20	TIMER2	TIMER2 interrupt request.
21	TIMER3	TIMER3 interrupt request.
22	TIMER4	TIMER4 interrupt request.
23	TIMER5	TIMER5 interrupt request.
24	TIMER6	TIMER6 interrupt request.
25	TIMER7	TIMER7 interrupt request.
26	TIMER8	TIMER8 interrupt request.
27	CAPTIMER	GPIO capture interrupt request.
28	SYSPLL_LOCK	PLL480 MHz lock interrupt request.
29	AUX_ADC	AuxADC interrupt request.
30	RTC_IF_EXTWK	External wake-up interrupt request.
31	RTC_IF_BLACK	Voltage blackout interrupt request.
32	RTC_IF_BROWN	Voltage brownout interrupt request.
33	RTC_IF_PCNT	Pulse count interrupt request.
34	RTC_IF_BCF_MSR	Bus clock measure interrupt request.
35	RTC_IF_RTC_ACC	Access to RTC core interrupt request.
36	RTC_IF_EXP	RTC mirror free running count interrupt request.
37	RTC_IF_LMR	FRC to mirroring, loading done interrupt request.
38	MRM	Instruction cache Miss Rate Monitor interrupt request.
39	DCACHE_MRM	Data cache Miss Rate Monitor interrupt request.
40	CC312	CryptoCell-312 interrupt request.

IRQ #	Name	Description
41	GPIO_P0	GPIO port 0 toggle interrupt request.
42	GPIO_P1	GPIO port 1 toggle interrupt request.
43	FPLL_LOCK	FPLL lock interrupt request.
44	WIFI_HSU	Wi-Fi hardware security unit interrupt request.
45	WIFI_MODEM	Wi-Fi modem interrupt request.
46	WIFI_MACTIMER	Wi-Fi MAC TX/RX timer interrupt request.
47	WIFI_MACOTHER	Wi-Fi MAC TX/RX Misc interrupt request.
48	WIFI_MACRX	Wi-Fi MAC RX Trigger interrupt request.
49	WIFI_MACTX	Wi-Fi MAC TX Trigger interrupt request.
50	WIFI_MACPROT	Wi-Fi MAC protocol trigger interrupt request.
51	WIFI_MACINTGEN	Wi-Fi MAC general interrupt request.
52	WIFI_MACBCN	Wi-Fi MAC beacon reception interrupt request.
53	WIFI_RC	Wi-Fi Radio controller interrupt request.
54	SDEMMC	SDEMMC interrupt request.
55	SDEMMC_WKUP	SDEMMC wake-up interrupt request.

7.1.3 Debug

The Cortex M33 provides an SWD interface for real time software debugging.

Features

- Processor halt, single-step, processor core register access, vector catch, unlimited software breakpoints, and full system memory access.
- Hardware breakpoints and watchpoints:
  - A breakpoint unit supporting four to eight instruction comparators.
  - A watchpoint unit supporting two or four data watchpoint comparators.
- MTB provides 8 kB of trace data to support profiling and timestamping of the code execution.

7.2 Internal Memory Architecture

7.2.1 Introduction

The internal memory architecture consists of various memory types including ROM, System RAM, Retention RAM, and OTP.

7.2.2 ROM

ROM is a 256 kB read-only memory that includes the boot loader used to initialize the device and prepare for executing the main firmware. The ROM also includes various library functions for security, DTIM related Wi-Fi MAC drivers/Wi-Fi stack. These are used to reduce the code size of DTIM tasks that may be executing from internal SRAM.

7.2.3 System RAM

System RAM provides 704 kB of internal memory for applications running on the M33 and the Wi-Fi subsystem. The system RAM consists of three blocks which can independently be powered off or put into a low-power retention state to support minimizing power consumption for low-power applications.

Features

- Low latency access
  - 8-way interleaved to minimize latency

- Parallelize data flows from/to various masters in the system:
  - M33, code and data
  - DMA
  - Wi-Fi subsystem
  - SDIO
- Low-power retention mode to minimize power consumption
  - Can be selectively retained during Sleep modes
- Store data or code.

7.2.4 Retention RAM

Retention RAM is a 64 kB block of memory which is used during low-power Sleep modes to maintain the Wi-Fi operating state.

When operating in DPM mode, a small application is also stored in retention memory that is executed directly upon a wake-up event to check if there are any pending Wi-Fi requests. This minimizes wake-up time and reduces the need to power up the full device if there is no activity.

Features

- Low latency access
- Low-power retention mode to minimize power consumption
  - Retained during Sleep modes
- Store data or code.

7.2.5 OTP

The OTP memory is used to store and protect important information essential for mass production and the management of end products, such as boot information, MAC addresses, and serial numbers. It is also used for storing secret information that is used by the advanced security functions like secure boot, secure debug, and secure asset storage. This secret information is programmed during a secure manufacturing process and then locked so that it cannot be accessed directly by CPU read or write operations therefore protecting it from external access.

The OTP memory array supports write accesses of 1 bit and read accesses of 32 bits by executing read/write commands through the OTP controllers register interface.

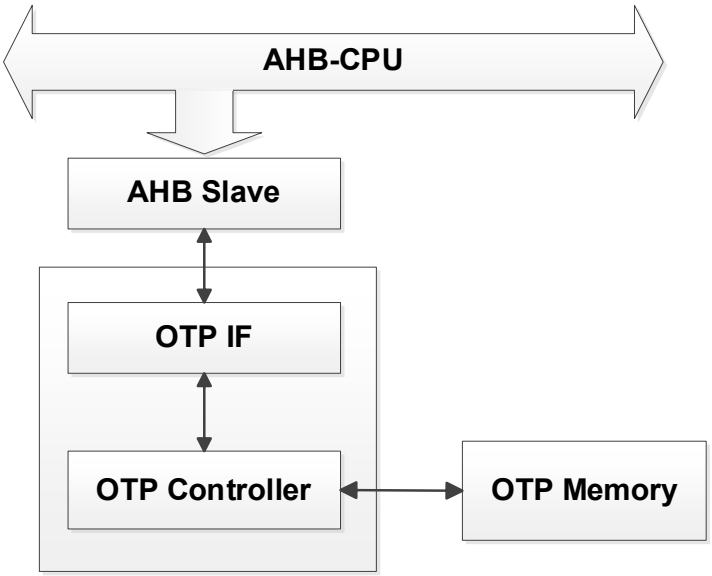


Figure 10. OTP block diagram

The OTP is a 2 kB one-time programmable memory which provides storage for permanent information.

Features

- Storage for device secrets to provide high level of security
  - These device secrets are only accessible by crypto hardware
- Storage for boot and system configuration information
  - MAC addresses
  - Calibration information
  - Configuration script
- Storage for user defined information.

Table 30: OTP memory map

Start index	End index	Size (byte)	Description	Notes
0x000	0x02C	180	Security parameters (lockable)	Hidden Locked for write access
0x02D	0x03F	76	User data (lockable)	Locked for write access
0x040	0x0FF	768	CS script	Configuration parameters (Note 1)
0x100	0x1FF	1024	User data	1 kB user data

Note 1 The configuration parameters consist of both chip manufacturing and customer manufacturing parameters.

The configuration script (CS) is an array of values stored in OTP that is parsed by the boot loader during startup. The values stored in the CS area are either commands which are used to do special configuration or data values such as MAC address, calibration data, or trim values for the Wi-Fi subsystem.

7.2.6 System Address Map

The address map is organized as shown in Table 31.

Table 31: System address map

Map	Region	Start Addr	End Addr	Size (kB)	PD	AMBA	Descriptions
Code	Remapped Devices	0000000	8000000	131072		AHB	
	SYS_RAM	8000000	80B0000	704	PD_SYS_MEM	AHB	MinAddr ~ MinAddr +32 kB*8*2 (8-way interleaved) MinAddr +32 kB*8*2 ~ MaxAddr (2-way interleaved)
	SYS_RAM (SDIO CIS MEM)	80AFC00	80AFE00	0.5	PD_SYS_MEM	AHB	memctrl_apbreg
	SYS_RAM (I-CACHE IVT)	80AFE00	80B0000	0.5	PD_SYS_MEM	AHB	ICACHE remapper
	Retention Memory	8600000	8610000	64	PD_RET	AHB	
	Retention Memory (M33 MTB MEM)	860F000	8610000	4	PD_RET	AHB	M33 MTB base address
	Retention Memory (DEBUG_SECTION)	860FEEC	8610000	0.27	PD_RET	AHB	
	Reserved						
	OQSPIF_C	9000000	A000000	16384	PD_SYS	AHB	
	OQSPIF_M	A000000	E000000	65536	PD_SYS	AHB	
	CACHE_SYS_MON_DATA	E000000	E002000	8	PD_SYS	AHB	

Map	Region	Start Addr	End Addr	Size (kB)	PD	AMBA	Descriptions
	CACHE_SYS_MON_TAG	E004000	E006000	8	PD_SYS	AHB	
	CACHE_C	E010000	E050000	256	PD_SYS	AHB	
	Mask ROM	F020000	F060000	256	PD_SYS	AHB	
Data	SYS_RAM	20000000	200B0000	704	PD_SYS_MEM	AHB	MinAddr ~ MinAddr +32 kB*8*2 (8-way interleaved) MinAddr +32 kB*8*2 ~ MaxAddr (2-way interleaved)
	SYS_RAM (SDIO CIS MEM)	200AFC00	200AFE00	0.5	PD_SYS_MEM	AHB	memctrl_apbreg
	SYS_RAM (I-CACHE IVT)	200AFE00	200B0000	0.5	PD_SYS_MEM	AHB	ICACHE remapper
	DCACHE_C	21000000	21008000	32	PD_SYS_MEM	AHB	
	DCACHE_M	21010000	21018000	32	PD_SYS_MEM	AHB	
	QSPIF2_C	22000000	23000000	16384	PD_SYS	AHB	
	QSPIF2_M	24000000	28000000	65536	PD_SYS	AHB	
	Retention Memory	28600000	28610000	64	PD_RET	AHB	
	Retention Memory (M33 MTB MEM)	2860F000	28610000	4	PD_RET	AHB	M33 MTB base address
	Retention Memory (DEBUG_SECTION)	2860FEEC	28610000	0.26953	PD_RET	AHB	
	Reserved						
	OQSPIF_C	29000000	2A000000	16384	PD_SYS	AHB	
	OQSPIF_M	2A000000	2E000000	65536	PD_SYS	AHB	
	CACHE_C	2E010000	2E050000	256	PD_SYS	AHB	
	AHB_DMA_B (SYSBUS)	2F000000	2F000400	1	PD_SYS	AHB	
Peri.	UART0	40000000	40000100	0.25	PD_SYS	AHB	
	UART1	40001000	40001100	0.25	PD_SYS	AHB	
	UART2	40002000	40002100	0.25	PD_SYS	AHB	
	SRC_FIFO_IF	40003000	40003100	0.25	PD_SYS	AHB	
	Reserved						
	SDIO Device	40010000	40010100	0.25	PD_SYS	AHB	
	SDEMMC Host	40011000	40011100	0.25	PD_SYS	AHB	
	Reserved						
	RTC Interface	40038000	40039000	4	PD_SYS	AHB	
	Reserved						
	VERSION	40070200	40070300	0.25	PD_SYS	APB	
	GPREG	40070300	40070400	0.25	PD_SYS	APB	
	AMBACORE (SYSBUS_ICM)	40070400	40070500	0.25	PD_SYS	APB	

Map	Region	Start Addr	End Addr	Size (kB)	PD	AMBA	Descriptions
	CRG_SYS	40070500	40070600	0.25	PD_SYS	APB	
	RFMON	40070600	40070700	0.25	PD_SYS	APB	
	DMA	40070700	40070800	0.25	PD_SYS	APB	
	KDMA	40070A00	40070B00	0.25	PD_SYS	APB	
	Reserved						
	TIMER/PWM1	40080000	40080100	0.25	PD_SYS	APB	
	TIMER/PWM2	40080100	40080200	0.25	PD_SYS	APB	
	TIMER/PWM3	40080200	40080300	0.25	PD_SYS	APB	
	TIMER/PWM4	40080300	40080400	0.25	PD_SYS	APB	
	TIMER/PWM5	40080400	40080500	0.25	PD_SYS	APB	
	TIMER/PWM6	40080500	40080600	0.25	PD_SYS	APB	
	TIMER/PWM7	40080600	40080700	0.25	PD_SYS	APB	
	TIMER/PWM8	40080700	40080800	0.25	PD_SYS	APB	
	Reserved						
	I2C1	40090000	40090100	0.25	PD_SYS	APB	
	I2C2	40090100	40090200	0.25	PD_SYS	APB	
	SPI1	40090200	40090300	0.25	PD_SYS	APB	
	SPI2	40090300	40090400	0.25	PD_SYS	APB	
	CRG_PER	40090400	40090500	0.25	PD_SYS	APB	
	AuxADC	40090500	40090600	0.25	PD_SYS	APB	
	CLKCAL	40090600	40090700	0.25	PD_SYS	APB	
	Reserved						
	GPIO	400B0000	400B0300	0.75	PD_SYS	APB	P0_04~P0_07: PD_SEN
	MEMCTRL	400B0800	400B0900	0.25	PD_SYS	APB	
	MEMCTRL_RET	400B0900	400B0A00	0.25	PD_SYS	APB	
	Reserved						
	CRG_TOP	400C0000	400C0100	0.25	PD_SYS	APB	
	CRG_COM	400C0200	400C0300	0.25	PD_SYS	APB	
	PLL_DIG (FPLL)	400C0300	400C0400	0.25	PD_SYS	APB	
	Watchdog	400C0700	400C0800	0.25	PD_SYS	APB	
	Reserved						
	CRG_APU	400E0000	400E0100	0.25	PD_SYS	APB	
	APU_AUD	400E0400	400E0500	0.25	PD_SYS	APB	
	APU_DSP	400E0800	400E0900	0.25	PD_SYS	APB	
	DAI	400E0900	400E0A00	0.25	PD_SYS	APB	
	SRC_IF	400E0C00	400E0D00	0.25	PD_SYS	APB	
	Reserved						
Ext.	External RAM	60000000	8FFFFFFF			AHB	(6090000 ~ 60D400FF: reserved)
	External Device	90000000	DFFFFFFF			AHB	

7.3 Clock Generation

7.3.1 Introduction

The Clock Generation block is part of the Clock and Reset Generator (CRG) block. The CRG block provides clocks to the CPU (CM33), bus structure and core clock of several peripherals (I2C, SPI) and the AuxADC. It also generates 240 MHz for the Wi-Fi subsystem, which in turn generates its local derived clocks.

The following clock sources are available to drive the specific subsystems:

- RCX (OSC32): ~ 32 kHz
- XTAL32K: 32.768 kHz external crystal clock source
- XTAL40M: 40 MHz external crystal clock source
- SYS PLL: 480 MHz PLL system clock source
- FPLL: 90.316 ~ 98.304 MHz clock for the digital audio interface.

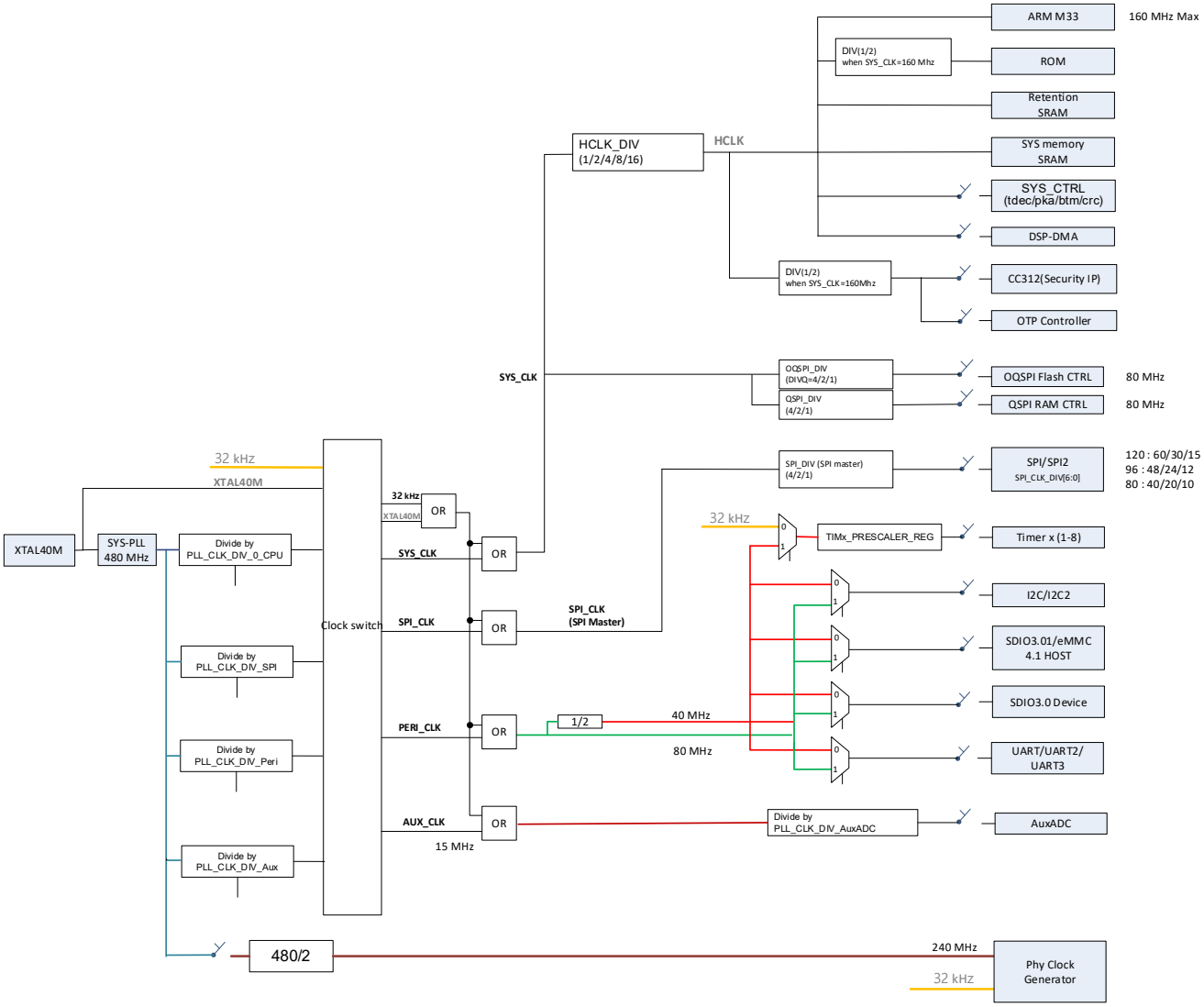


Figure 11. System clocktree diagram

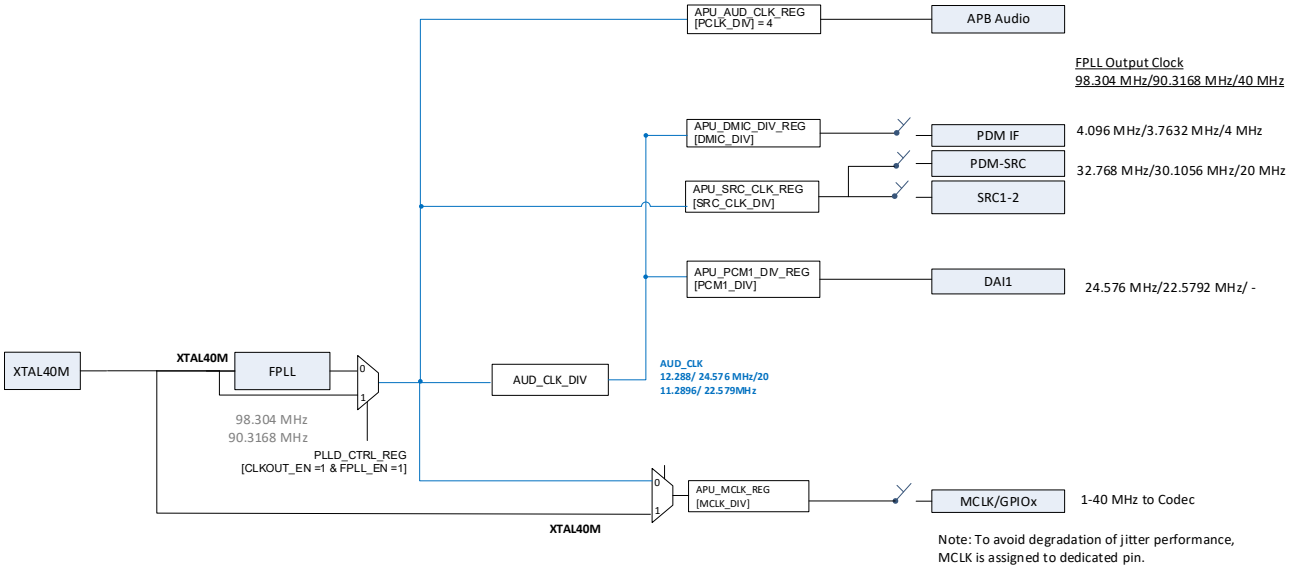


Figure 12. Digital audio interface clocktree diagram

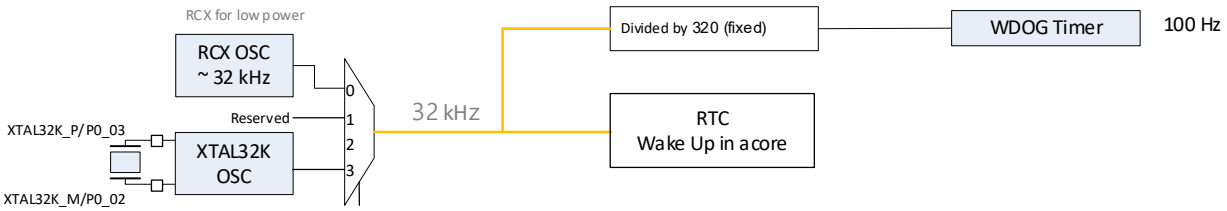


Figure 13. RTC clocktree diagram

7.3.2 System Clock (SYS_CLK, HCLK)

SYS_CLK is the source clock of HCLK divider block and OQSPI and QSPI divider block. It can be 32 kHz, 40 MHz (from the crystal), or a clock divided from the PLL. Table 32 lists the available clock frequencies.

Table 32: Available SYS_CLK frequencies

Clock source	SYS_CLK	HCLK	OQSPI/QSPI
XTAL32K/RC32K	32 kHz	32 kHz	N/A
XTAL40M	40 MHz	SYS_CLK divided by 1,2,4,8,16	SYS_CLK divided by 1,2,4
PLL480	160, 80, 40, 20 MHz	SYS_CLK divided by 1,2,4,8,16	SYS_CLK divided by 1,2,4

7.3.3 Peripheral Clocks (SPI_CLK, PERI_CLK, AUX_CLK)

The peripherals like UART, SDeMMC, SDIO, and SPI require 80 MHz whereas I2C needs 40 MHz. These are derived from the PLL clock. In XTAL mode, the 40 MHz crystal clock is supplied to the peripherals directly. AuxADC is used for internal calibration purposes of circuits such as internal temperature sensor. AUX_CLK is generated for this purpose.

7.3.4 Audio Clocks (AUD_CLK)

Audio clocks are generated from FPLL to cover various specific audio rate requirements such DAI, SRC, DMIC interface, and MCLK for external CODEC support.

7.3.5 RTC Clocks (32 kHz)

RTC clocks can be generated from external crystal (XTAL32K OSC) or internal oscillator (RCX OSC). It is used for watchdog timer and RTC timer.

7.4 Power Management

RTC block provides power management and function control for low-power operation. In normal operation, the RTC block is always powered on when RST_N is in HIGH state. The RTC block also has a control function for internal power supplying components such as LDOs, DC-DCs, and power switches.

7.4.1 Power-On Sequence

Figure 14 shows the sequence after the initial switching from power-off to power-on.

The RST_N disables the RTC block when the reset is asserted. When the reset is released, all the internal regulators are turned on automatically in the sequence predefined by the RTC block. At the initial phase of power-on sequence, RCX 32 kHz is used for RTC clock source.

When the external reset is released, LDOs for both XTAL and digital I/O are turned on shortly and then the DC-DC regulator is turned on according to the predefined interval. The enabling intervals can also be modified in the register settings after initial power-up.

The power-on sequence in an initial boot phase is as follows:

1. External reset assertion which makes the bias circuit to be disabled.
2. VBAT_POR asserts a reset to RTC block when external reset case or POR/BOR reset may occur depending on the VBAT voltage.
3. After reset is released from VBAT_POR, the RTC block enables RCX 32 kHz oscillator.
4. The RTC block uses the generated 32 kHz clock as a clock source.
5. The internal counter in RTC generates the enable signal for RCX 32K, DC-DC, and FDIO_LDO.
6. FDIO_LDO supplies power for the external flash related blocks, which are used for XiP after boot ROM sequences are completed.
7. DC-DC and DIG_LDO generate power for digital core sequentially.
8. After the RTC releases the reset to the digital core, the digital core starts the system boot sequences.

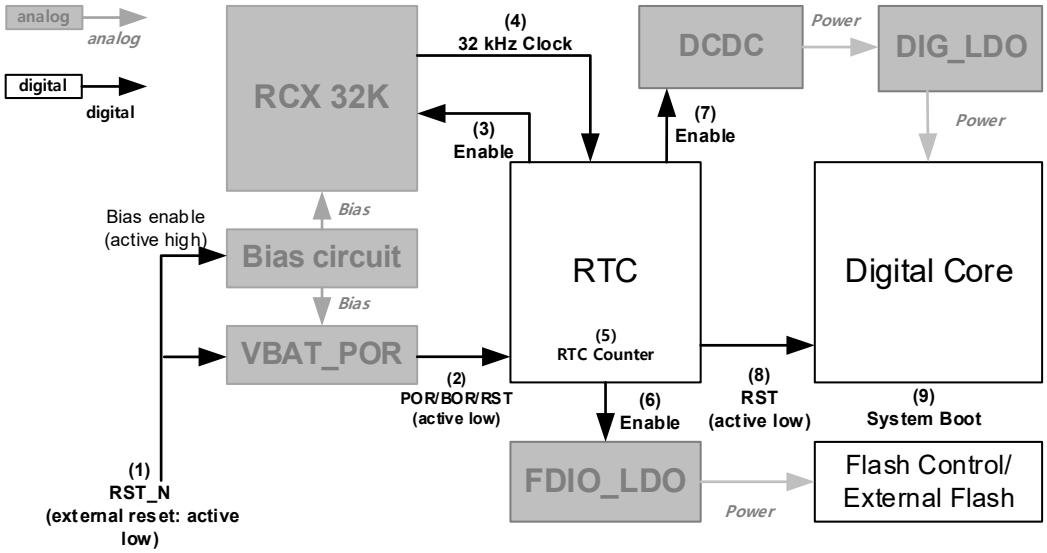


Figure 14. Power-on sequence

7.4.2 Power Management Unit

The DC-DC converters and LDOs to supply power to all internal sub-blocks. Power management does the on-off control of these regulators and is implemented through the register setting inside the RTC block.

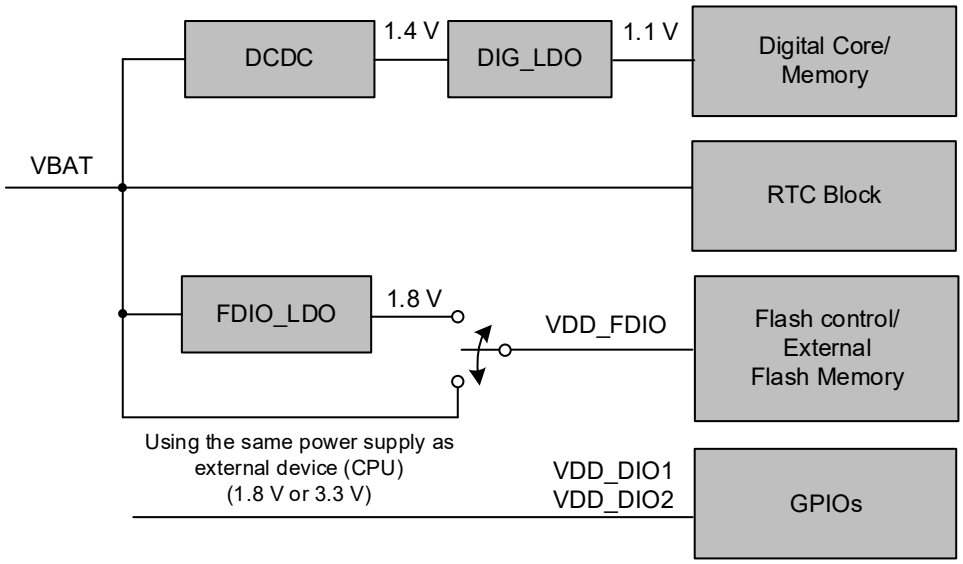


Figure 15. Power management block diagram

Details of the internal DC-DC converters and LDOs are explained:

- DC-DC converter: From the power supply of external VBAT input, it generates 1.4 V power for the digital LDO.
- DIG_LDO for digital blocks: From the DC-DC output, it generates 1.1 V power which is used for digital blocks.
- FDIO_LDO for I/O and external flash memory:
  - This LDO output is used only for 1.8 V digital I/O applications.
  - From external VBAT power input, it generates 1.8 V output voltage which is used for digital I/O power domain in 1.8 V digital I/O applications.
  - It is also used for external flash memory.
  - FDIO_LDO supports only 1.8 V.
  - For 3.3 V digital I/O applications, external power (3.3 V) is directly supplied for digital I/O power.

With the internal DC-DC converters and LDOs, all the power necessary for internal sub-blocks is sufficiently generated.

7.5 Sleep Modes

7.5.1 Introduction

There are several Sleep modes supported so that the lowest possible power can be obtained while maintaining the Wi-Fi connection and performing Wi-Fi communications.

The low-power modes are defined as follows:

- **Sleep 1 – Deep Sleep/Reset State (~0.3 µA)**
  In this state, power is applied (VBAT), but all internal circuitry is powered off such that when reset is de-asserted, the device can start without waiting for the power to stabilize.
- **Sleep 2 – Low Power Standby (~2.5 µA)**
  In this state, all power domains are off except for the RTC and wake-up pins such that a timer or external event can trigger the device to wake up from sleep.
- **Sleep 3 – Low Power Retention (~3.7 µA)**
  In this state, all power domains are off except for the RTC, wake-up pins, and the retention memory. The

retention memory keeps system information such as Wi-Fi state so that a fast wake-up can occur to do simple network management such as DTIM processing. If required, a full system wake-up can also be triggered.

- **Sleep 4/5 – Tickless Idle State (~200 µA/~500 µA)**
  Tickless idle state is like retention mode, but it also maintains up to 704 kB of RAM in a low-power state which allows for the full application to maintain its state and wake up without requiring full software reinitialization. The MAC hardware can also be kept alive with a 32 kHz clock to maintain the timers and allow for faster wake-up time depending on power/performance requirements.

7.5.2 Wake-Up Sources

Wake-up from sleep can be triggered by any of the following:

- Program the RTC block to wake up the device at a predefined time interval.
- GPIO trigger (immediate).
- GPIO counter (using the RTC in count mode, waiting to exceed a programmable threshold).
- ADC threshold sensing.

7.5.3 Sleep Mode Active Blocks Overview

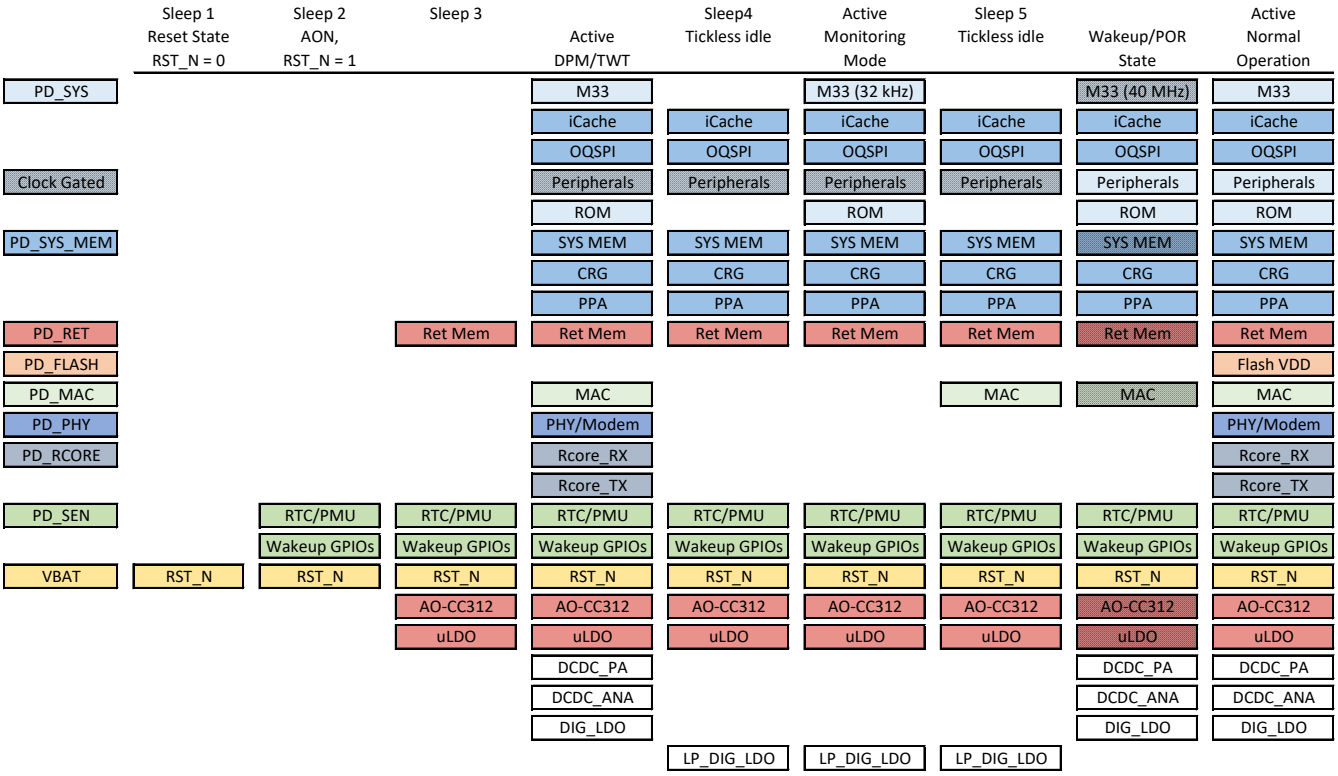


Figure 16. Sleep mode block overview

The blocks which are active during each Sleep mode are shown in Figure 16. Active DPM/TWT, Active Monitoring Mode and Active Normal Operation Mode are application-level examples to help the understanding of the power domain application. The color of each block represents the power domain. Wakeup/POR state explains the state after wake-up from Sleep or POR.

The following descriptions are summaries for each power domain.

- VBAT, DCDC_ANA, DCDC_PA, uLDO, LP_DIG_LDO, DIG_LDO
  VBAT is an external power supply for the overall system. From VBAT, DCDC_ANA, and DCDC_PA generate internal power for analog blocks and RF blocks (Rcore_RX, Rcore_TX). uLDO is connected to VBAT and supplied for the retention memory and the AO-CC312. This allows the low current in Sleep 3 mode. The LP_DIG_LDO is connected to DCDC_ANA and generates power for Tickless Idle States (Sleep 4/Sleep 5). DIG_LDO supplies power for the most digital block including M33 in Active mode.
- PD_SYS, PD_SYS_MEM, PD_RET

These power domains are required to execute codes in M33. PD_RET remains in state in Sleep 3. PD_RET domain includes 64 kB retention memory and is used to implement ultra-low-power DTIM.

- PD_FLASH

  This power domain is used to keep power in state for external NVM.

- PD_MAC, PD_PHY, PD_RCORE

  These power domains are required for the communication function of Wi-Fi. In Sleep 4 mode, only PD_MAC power domain is required to be on state.

- PD_SEN

  Excluding Sleep1, other sleep modes require this power domain in state. RTC timer and GPIO wake-up can be supported from PD_SEN domain. To leave Sleep 1 in Sleep 1 mode, use reset.

7.6 DMA

7.6.1 Introduction

There are two DMA controllers available which provide CPU independent transfers of data between peripherals and memory.

The General Purpose DMA (GP-DMA) provides 16 channels which can perform memory-memory or memory-peripheral data transfers.

The Fast DMA is specialized for transferring data from memory-to-memory and between memory and the Wi-Fi interface.

7.6.2 General Purpose DMA

The General-Purpose DMA controller supports data transfers to/from peripherals and memory through an independent DMA bus. It has sixteen channels to support the various peripherals available.

The GP-DMA controller off-loads the processor by performing transfers independent of the processor's execution allowing the processor to perform other operations and be notified by an interrupt after a transfer is completed. The DMA controller has eight levels of priority and is a bus master on the AHB-DMA bus with programmable priority level. The number of peripheral requests is multiplexed on the available channels to increase utilization of the DMA because not all peripherals are active at the same time.

Features

- Programmable source and destination addresses
- Programmable 16-bit transfer length and interrupt generation counters
- Programmable support of 8-bit,16-bit, and 32-bit transfers
- Programmable AHB bursts support (INCR4, INCR8, INCR)
- Request mode (DREQ_MODE) with programmable peripheral selection
  - Support also CIRCULAR, applicable only in DREQ_MODE
- Freeze option, applicable during memory-to-memory transfers as well
- Programmable bus error detection and IRQ generation
- Channel’s start/stop through setting/resetting the DMA_ON bit-field
  - Index and Circular index counters are reset in the next transfer
- Memory protection for general-purpose channels
  - Generate a bus error when the DMA R/W access is not allowed.

Figure 17 shows the mapping of peripherals to DMA channels.

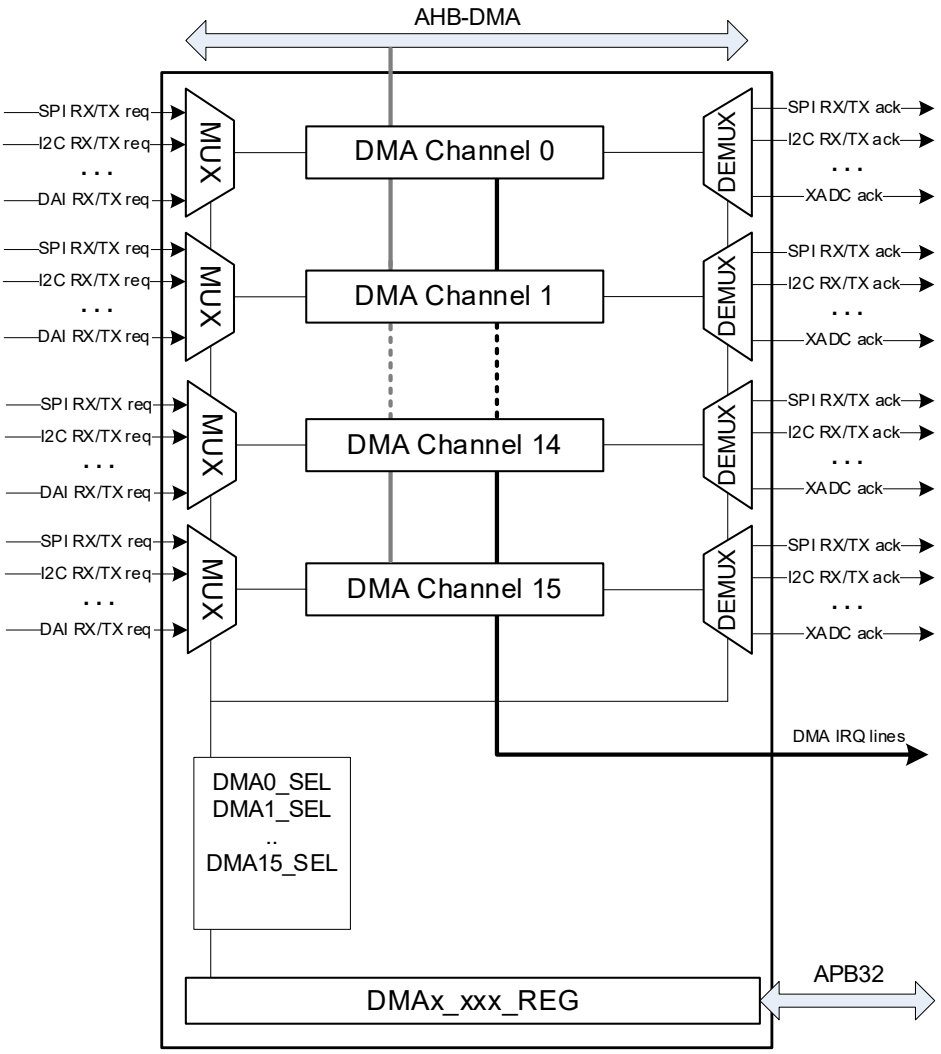


Figure 17. DMA channel mapping

The peripherals that support DMA data transfers are defined in Table 33.

Table 33: List of peripherals that support DMA data transfers

ID	Peripheral
0x0	SPI1_RX (Read)
0x1	SPI1_TX (Write)
0x2	SPI2_RX (Read)
0x3	SPI2_TX (Write)
0x4	UART0_RX (Read)
0x5	UART0_TX (Write)
0x6	UART1_RX (Read)
0x7	UART1_TX (Write)
0x8	UART2_RX (Read)
0x9	UART2_TX (Write)
0xA	I2C1_RX (Read)
0xB	I2C1_TX (Write)
0xC	I2C2_RX (Read)
0xD	I2C2_TX (Write)

ID	Peripheral
0xE	ADC[0]
0xF	ADC[1]
0x10	ADC[2]
0x11	ADC[3]
0x12	SRC_IN
0x13	SRC_OUT
0x14	DAI_TX
0x15	DAI_RX
Not available	Memory Read/Write (Note 1)

Note 1 If a DMA channel is not selected as a peripheral DMA, it can be used as a memory-to-memory DMA.

7.6.2.1 Input/Output Multiplexer

The multiplexing of peripheral requests is controlled by DMA_REQ_MUX_REG. Thus, if DMA_REQ_MUX_REG[DMAxy_SEL] is set to a certain (non-reserved) value, the TX/RX request from the corresponding peripheral should be routed to DMA channels y (TX request) and x (RX request) respectively. Similarly, an acknowledging de-multiplexing mechanism is applied. However, when two or more bit-fields (peripheral selectors) of DMA_REQ_MUX_REG have the same value, the less significant selector is given priority (see also the register's description).

7.6.2.2 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 or by a peripheral DMA request. If DREQ_MODE is 0, then a DMA channel is immediately triggered. If DREQ_MODE is 1, the DMA channel can be triggered by hardware interrupt.

If DMA starts, data is transferred from address DMAx_A_START_REG to address DMAx_B_START_REG for a length of DMAx_LEN_REG, which can be 8-bit, 16-bit, or 32-bit wide. The address increment is realized with an internal 13 bits counter DMAx_IDX_REG, which is set to 0 if the DMA transfer starts and is compared with the DMA_LEN_REG after each transfer. The register value is multiplied according to the AINC, BINC, and BW values before it is added to DMA_B_START_REG and DMA_B_START_REG. AINC or BINC must be 0 for register access.

If at the end of a DMA cycle and the DMA start condition is still true, the DMA continues. The DMA stops if DREQ_MODE is low or if DMAx_LEN_REG is equal to the internal index register. This condition also clears the DMA_ON bit.

If bit CIRCULAR is set to 1, the DMA controller automatically resets the internal index registers and continues from its starting address without intervention of the Arm Cortex-M33. If the DMA controller is started with DREQ_MODE = 0, the DMA should always stop, regardless of the state of CIRCULAR.

Each DMA channel can generate an interrupt if DMAx_INT_REG is equal to DMAx_IDX_REG. After the transfer and before DMAx_IDX_REG is incremented, the interrupt is generated.

For example, if DMA_x_INT_REG = 0 and DMA_x_LEN_REG = 0, there should be one transfer and an interrupt.

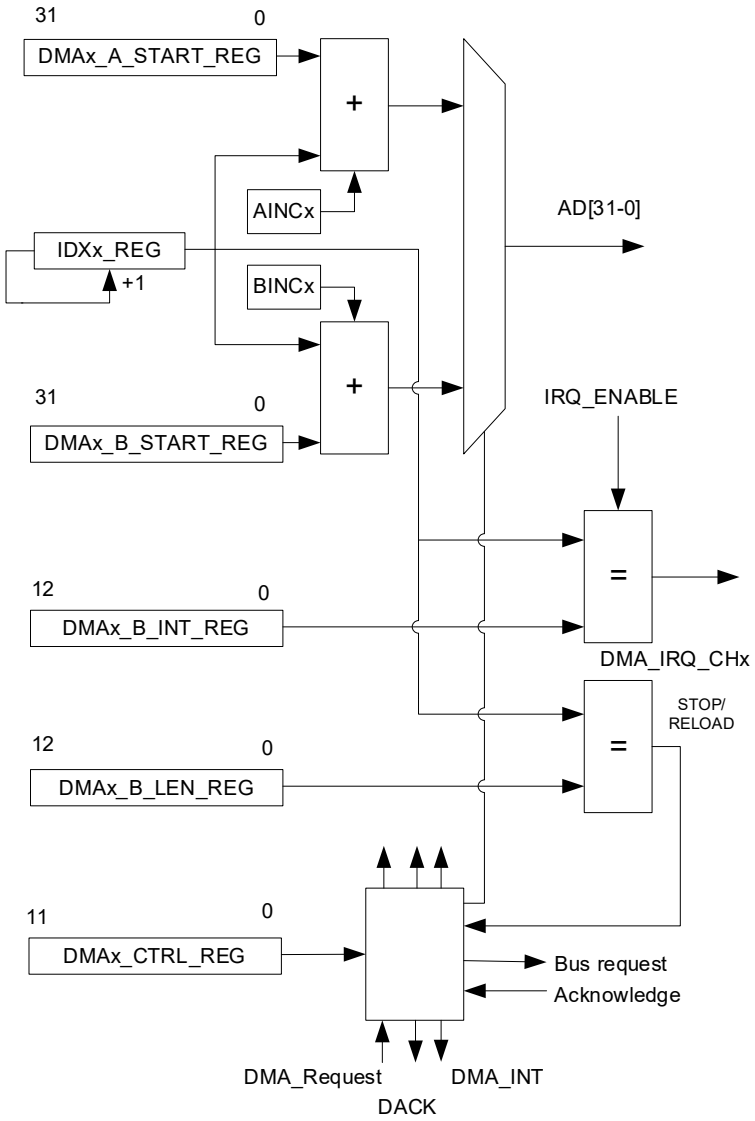


Figure 18. DMA channel diagram

7.6.2.3 DMA Arbitration

The priority level of a DMA channel can be set with bits DMA_PRIO[2-0]. These bits determine which DMA channel is activated in case more than one DMA channel requests DMA. If two or more channels have the same priority, an inherent priority applies (see register description).

With DREQ_MODE = 0, a DMA can be interrupted by a channel with a higher priority if the DMA_IDLE bit is set.

When DMA_INIT is set, however, the DMA channel currently performing the transfer locks the bus and cannot be interrupted by any other channels, until the transfer is completed, regardless of if DMA_IDLE is set. The purpose of DMA_INIT is to initialize a specific memory block with a certain value, fetched also from memory, without any interruption from other active DMA channels that may request the bus at the same time. Consequently, it should be used only for memory initialization, while when the DMA transfers data to/from peripherals, it should be set to 0. Also, AINC must be set to 0 and BINC to 1, when DMA_INIT is enabled.

NOTE
Memory initialization could also be performed without having the DMA_INIT enabled and by simply setting AINC to 0 and BINC to 1, provided that the source address memory value does not change during the transfer. However, it is not guaranteed that the DMA transfer is not interrupted by other channels of higher priority when these request access to the bus at the same time.

7.6.2.4 Freezing DMA Channels

Each channel of the DMA controller can be temporarily disabled by writing a 1 to freeze all channels at SET_FREEZE_REG. To enable the channels again, a 1 to bits at the RESET_FREEZE_REG must be written. There is no hardware protection from erroneous programming of the DMA registers.

The on-going Memory-to-memory transfers (DREQ_MODE = 0) cannot be interrupted. Therefore, in that case, the corresponding DMA channels are frozen after any on-going Memory-to-memory transfer is completed.

7.6.3 Fast DMA

The Fast DMA (F-DMA) controller performs bulk data transfers, reading data from the source address range, and writing the data to the destination address range. The Fast DMA is used for fast data transfers from memory to memory and between memory and the Wi-Fi subsystem.

Features

- Programmable transfer size from 1 byte to 1 MB
- Up to eight channels can be set at the same time
- Freeze option for each channel
- Support for buffer chaining
- Interrupt enable for each channel.

The basic unit of bus transmission is 32-bit and has a function to automatically correct address aligns, even if the source and destination addresses are not in word units.

For example, assuming that the transfer size is 23 bytes, the source base address is 0x001 for read access, and the destination base address is 0x102 for write access, the number of bytes per transaction is performed as follows:

- Source base address [1:0] = 0x1: the master read port of fast DMA performs read access with the following sequences:
  - 1 > 2 > 4 > 4 > 4 > 4 > 4 bytes
- Destination base address [1:0] = 0x2: the master write port of fast DMA performs write access with the following sequences:
  - 2 > 4 > 4 > 4 > 4 > 4 > 1 bytes

7.7 Hardware Accelerators

7.7.1 CRC Calculation

The cyclic redundancy check (CRC) is an efficient method for detecting corruption in the transmission and storage of data. The CRC calculation consists of an iterative algorithm involving XOR and shifts operations. The CRC calculator is mainly used to check the integrity of the firmware image stored in Flash.

Features

- Operate at a clock frequency up to system clock
- Support 8-bit, 16-bit, and 32-bit data paths
- Be a master on the bus and can operate over a region of memory independent of the CPU
- Support various methods of calculating the CRC as defined by the following generator polynomials:
  - CRC-32:
    $G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$
  - CRC-16 CCITT:
    $G(x) = x^{16} + x^{12} + x^5 + 1$
  - CRC-16 IBM:
    $G(x) = x^{16} + x^{15} + x^2 + 1$

7.7.2 Pseudo Random Number Generation

The Pseudo Random Number Generator (PRNG) is used to generate a stream of pseudo random numbers.

Features

- Operation clock frequency is the same as the system clock
- Support partial parallel processing of 8-bit, 16-bit, and 32-bit units.

Given a seed value from the TRNG, the PRNG generates a stream of random values based on a generator polynomial defined as follows:

$$G(x) = x^{31} + x^{28} + 1$$

If you set the same configuration in the PRNG after POR boot (not Software reset), the PRNG always generates a deterministic sequence which is the same pattern. The PRNG uses SEED input to avoid generating a predictable next pattern. However, this SEED is designed to accumulate the current calculation without resetting the poly calculation, to make its pattern different from the previous pattern. Additionally, the PRNG is designed not to be reset at runtime to prevent predictable patterns within the system as much as possible.

7.8 Watchdog Timer

7.8.1 Introduction

The watchdog timer provides a mechanism to detect if the software executing on the Cortex-M33 has halted because of a software issue or a system fault. This allows the system to recover either through a non-maskable interrupt to the Cortex-M33 or through a system reset.

Features

- A 13-bit down counter clocked at 100 Hz
- Clock sources: RC32K or XTAL32
- Generate:
  - NMI to Cortex-M33 if counter reaches 0
  - Hardware reset if counter reaches -16
- Protected by a lock bit to avoid accidental freeze
- Automatically froze when Cortex-M33 is in debug mode.

The 13-bits System Watchdog is supplied by the always on power domain (PD_AON) and is automatically enabled as soon as the system powers up, see Figure 19. It is decremented by one every 10 ms, clocked by RC32K/XTAL32K. The timer value can be accessed through WATCHDOG_REG that is set to its max value at reset which is a maximum watchdog time-out of 81 seconds.

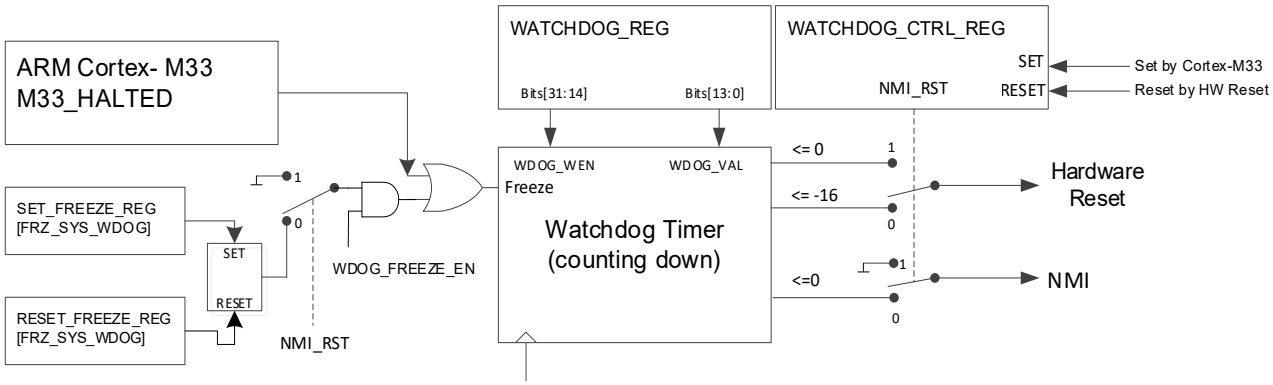


Figure 19. System watchdog block diagram

The watchdog timer can be configured to either generate an interrupt or system reset when a timeout occurs.

If the watchdog timer is configured to generate an interrupt, it generates an NMI to the Cortex-M33 when the watchdog timer reaches 0 and a hardware reset when the counter becomes less than or equal to -16. The NMI

handler must write any value > -16 to the WATCHDOG_REG to prevent the generation of a WDOG reset within 16x10 ms = 160 ms.

If the watchdog timer is configured to reset the system, it generates a system reset if the timer becomes less than or equal to 0.

The system watchdog can be frozen by either the Cortex-M33 directly by writing to a specific register or frozen automatically when the debugger is attached, and the Cortex-M33 CPU is halted during debugging. Even if the watchdog is frozen by the Cortex-M33, the watchdog resumes operation automatically when entering one of the Sleep modes where PD_SYS is turned off.

7.9 Brownout and Blackout Detection

The device enters a brownout or blackout condition whenever the input voltage dips below $V_{brownout}$ or $V_{blackout}$ (see Table 34). This condition must be considered during the design of the power supply routing, especially for battery operated applications. High current operations such as Wi-Fi TX may cause a dip in the supply voltage, potentially triggering a Brownout. When designing a battery-operated system, the overall circuit resistance must be considered which includes the internal resistance of the battery, the contact resistance of the battery holder (for example, four contacts for two AA batteries), the wiring resistance, and the PCB routing resistance.

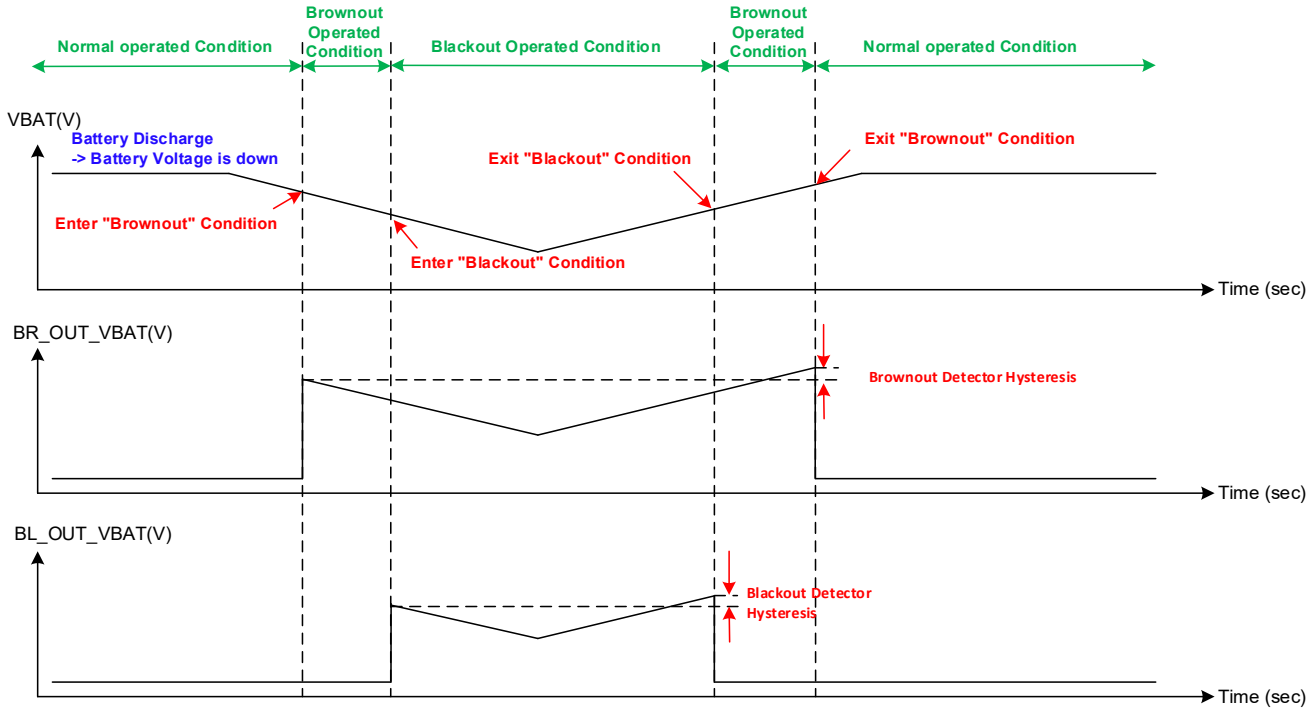


Figure 20. Brownout and blackout levels

Brownout and blackout conditions are monitored only when in normal operating mode (not during low-power Sleep modes). The blackout and brownout condition have hysteresis, and the blackout condition can be used for low battery level indication. The brownout condition can be used as an exit condition for low battery level state.

When blackout events occur, the proper handling is required for the low battery situation. After that, a brownout event occurs, recovering to the normal state is required.

When a brownout or blackout condition is entered, an interrupt can be generated for either case. The voltage level which triggers a brownout or blackout condition is configurable as shown in Table 34.

There is also a hardware POR, which monitors VBAT and asserts overall system reset. It detects 1.68 V rising level and 1.55 V falling level. Figure 21 shows the VBAT voltage level monitoring hardware. VBAT_POR generates a hardware reset and Blackout/Brownout detector generates the related interrupts.

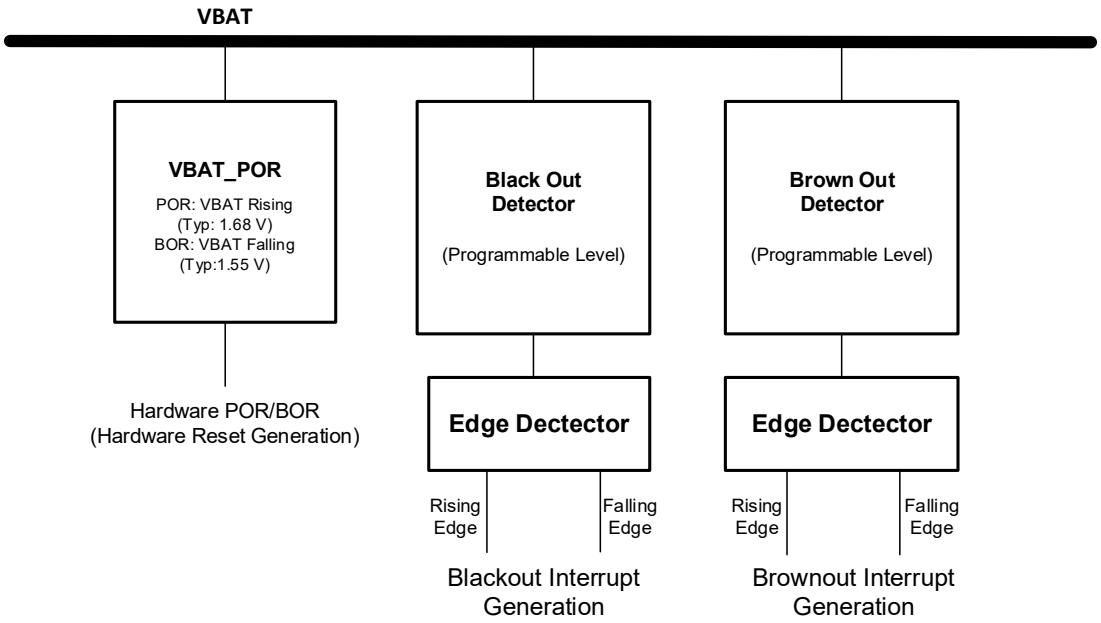


Figure 21. VBAT and POR, blackout/brownout detector

Table 34: $V_{brownout}$ and $V_{blackout}$ voltage levels

CTRL ID	HtoL (Avg.)	V_LtoH (Avg.)	Hysteresis (Avg.)
4	1.733	1.827	0.095
5	1.826	1.923	0.097
6	1.927	2.026	0.098
7	2.022	2.123	0.101
8	2.132	2.234	0.102
9	2.230	2.334	0.103
10	2.332	2.436	0.104
11	2.433	2.537	0.105
12	2.536	2.642	0.105
13	2.638	2.744	0.106
14	2.742	2.848	0.106
15	2.844	2.951	0.107

7.10 Security Features

The security engine provides acceleration of various crypto algorithms and provides protection for a secure Root of Trust (RoT) maintained within the device.

Features

- Secure XIP from Flash
  Secure XIP (eXecute In Place) supports AES-CTR description for XIP operation.
- Secure Crypto Engine
  - Symmetric algorithms: AES, DES/3DES, CHACHA
  - Hash/HMAC: SHA1/224/256
  - Asymmetric algorithms: RSA, DH, ECC
  - TRNG
- Secure boot

Secure boot and secure loading processes are required to ensure that only authorized software can be executed on a device. Unauthorized boot code should be detected and prevented.

- Secure debug (SWD)

  When the device enters the Secure mode, it can extract the SOC_ID. Every deployed device has a unique SOC_ID. If the image has a valid debug certificate, the internal security engine allows the debugger even if it is in Secure mode.

- Secure asset storage

  Secure Asset is a cryptographic service provided to protect data stored in external storage (Serial Flash memory). Data can be encrypted or decrypted with the provisioning key stored in the chip. Production-Line Provisioning is used to protect the data used in the mass production process, and Asset Provisioning is used to protect the data used during system operation.

- Device lifecycle management

  This mechanism enables the device to behave differently in each life cycle, protecting any security assets when they are introduced into the device and reducing the risk of IP theft and reverse engineering.

- TLS/DTLS protocol acceleration

  Hardware engines can support TLS/DTLS acceleration.

- DRBG (CTR_DRBG with AES, HMAC_DRBG)

  The device supports hardware based Deterministic Random Bit Generator (DRBG).

7.10.1 Crypto Engine

The hardware crypto engine provides acceleration of many crypto algorithms such as hashing, secret key generation, encryption/decryption, and sign/verify operations.

Table 35 shows the hardware accelerated crypto algorithms supported.

Table 35: Hardware accelerated crypto algorithms

Algorithm	Mode	Key sizes
AES	ECB, CBC, CTR, OFB, CMAC, CBC-MAC, AESCCM, AES-CCM*, AES-GCM	128 bits, 192 bits, and 256 bits
	XTS, CFB128	Supported by software with AES-ECB hardware acceleration
AES key wrapping	N/A	All (MbedTLS)
Chacha20 and Poly1305	N/A	256 bits
Diffie-hellman ▪ ANSI X9.42-2003: Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography ▪ Public-Key Cryptography Standards (PKCS) #3: Diffie Hellman Key Agreement Standard	N/A	1024 bits, 2048 bits, and 3072 bits
ECC key generation	N/A	NIST curves and 25519 curves
ECIES	N/A	NIST curves and 25519 curves
ECDSA	N/A	NIST curves and ED25519
ECDH	N/A	NIST curves and 25519 curves
Hash	SHA1, SHA224, and SHA256	N/A
	SHA384, SHA512, and MD5	Supported by software
HKDF	N/A	N/A
HMAC	SHA1, SHA224, SHA256, and MD5	N/A

Algorithm	Mode	Key sizes
KDF ▪ NIST SP 800-108: Recommendation for Key Derivation Using Pseudorandom Functions	CMAC or HMAC	N/A
KDF ▪ Password based	PBKDF2-HMAC	Supported by software (with SHA1 hardware support)
RSA PKCS#1 operations ▪ Public-Key Cryptography Standards (PKCS) #1 v2.1: RSA Cryptography Specifications ▪ Public-Key Cryptography Standards (PKCS) #1 v1.5: RSA Encryption	Encryption and signature schemes	2048 bits, 3072 bits, and 4096 bits
RSA key generation	N/A	1024 bits to 3072 bits
RSA key validation	N/A	1024 bits to 4096 bits

7.11 Debug Support

The device supports the low-pin-count Arm Serial Wire Debug (SWD) interface for real time debugging of the software. The Micro Trace Buffer (MTB) is included to provide support for tracing and real time profiling of code. The SWD protocol provides the same debug features as JTAG.

Features

- Processor halt, single-step, processor core register access, Vector Catch, unlimited software breakpoints, and full system memory access
- Hardware breakpoints and watchpoints:
  - A breakpoint unit supporting eight instruction comparators
  - A watchpoint unit supporting four data watchpoint comparators
- The MTB provides 8 kB of trace data to support profiling and timestamping of the code execution.

Figure 22 shows the SWD timing diagram.

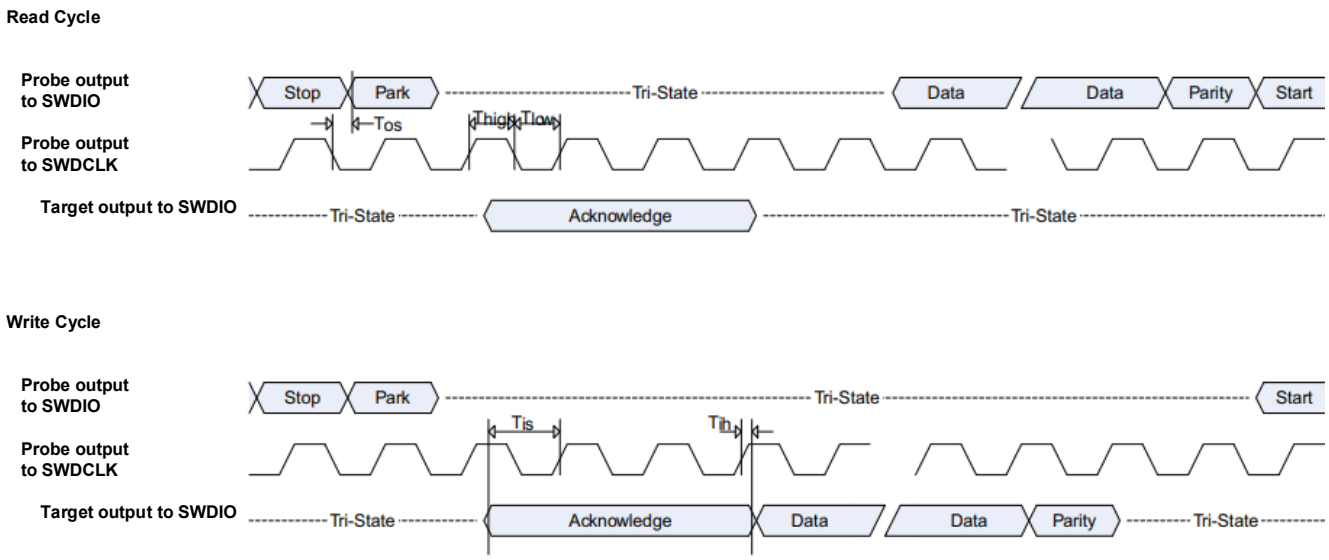


Figure 22. SWD timing diagram

Table 36 shows the SWD timing parameters.

Table 36: SWD timing parameters

Parameter	Parameter name	Min	Max	Unit
T_{high}	SWDCLK High period	10	500	us
T_{low}	SWDCLK Low period	10	500	us
T_{os}	SWDIO output skew to falling edge SWDCLK	-5	5	ns
T_{is}	Input Setup time required between SWDIO and rising edge SWDCLK	4		ns
T_{ih}	Input Hold time required between SWDIO and rising edge SWDCLK	1		ns

Table 37 shows the pin definition of the JTAG interface.

Table 37: SWD pin configuration

Pin name	Default pin assignment	I/O	Description
SWCLK	P1_16	I	Serial Wire Debug clock. The maximum clock frequency is 10 MHz.
SWDIO	P1_17	I/O	Serial Wire Debug data I/O.

Supported debug probes:

- Segger J-Link: https://www.segger.com/products/debug-probes/j-link/
- Segger J-Trace Pro:
  https://www.segger.com/products/debug-trace-probes/
  https://www.segger.com/downloads/jlink/
- Renesas E2 emulator:

https://www.renesas.com/us/en/software-tool/e2-emulator-rte0t00020kce00000r

8. Peripherals

8.1 UART

8.1.1 Introduction

The UART interface provides an industry compliant serial interface for communicating with other devices. Three independently configurable UARTs are available which support the RS-232 and RS-485 protocols.

- **Features**
  - Support RS-232 and RS-485 protocols
  - 32-byte deep transmit and receive FIFOs with a programmable FIFO level interrupt
  - Support for DMA
  - False start bit detection
  - Fully programmable serial interface characteristics:
    - Number of data bits per character (5, 6, 7, or 8 bits)
    - Optional parity bit (with odd or even select) and number of stop bits (1or 2)
    - Programmable flow control (CTS/RTS)
    - Programable baud rate generation:
    - Up to 921600 baud for UART
    - Up to 5 MBaud for RS-485.

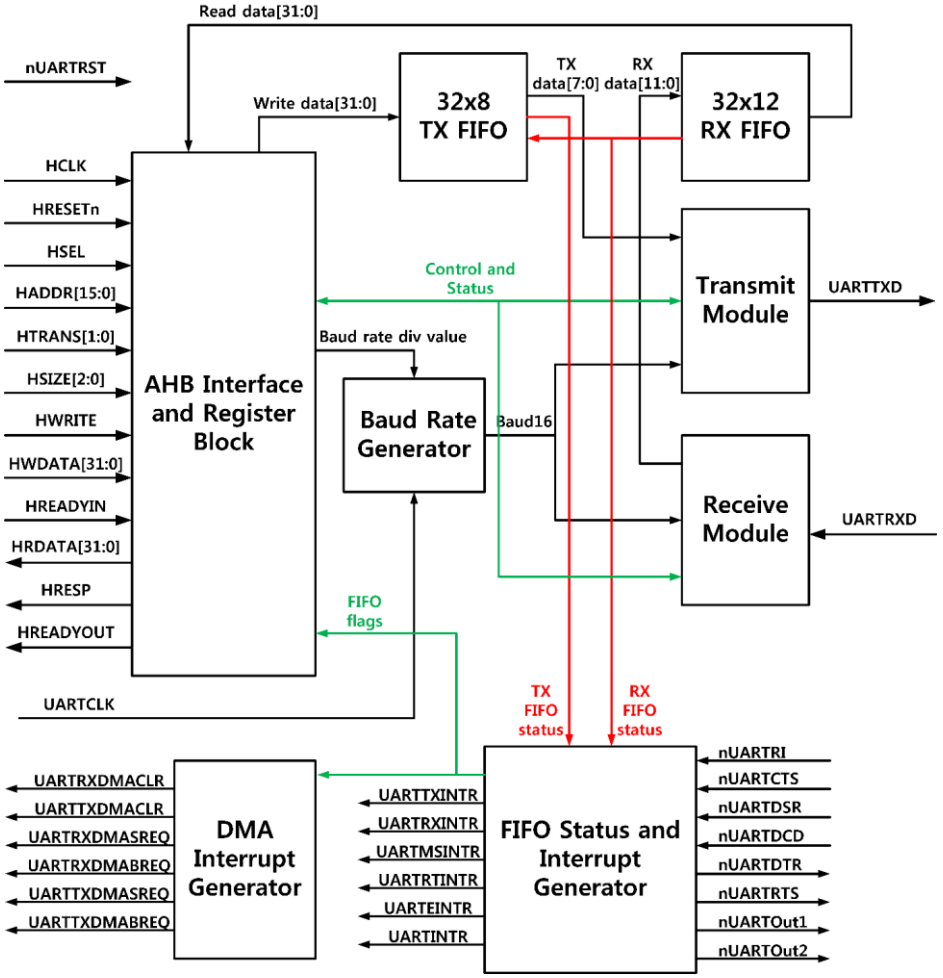


Figure 23. UART block diagram

8.1.2 RS-232

As the serial communication between the UART and the selected device is asynchronous, additional bits (start and stop) are inserted into the data line to indicate the beginning and end. With these bits, two devices can be synchronized. This structure of serial data accompanied by start and stop bits is referred to as a character, as shown in Figure 24.

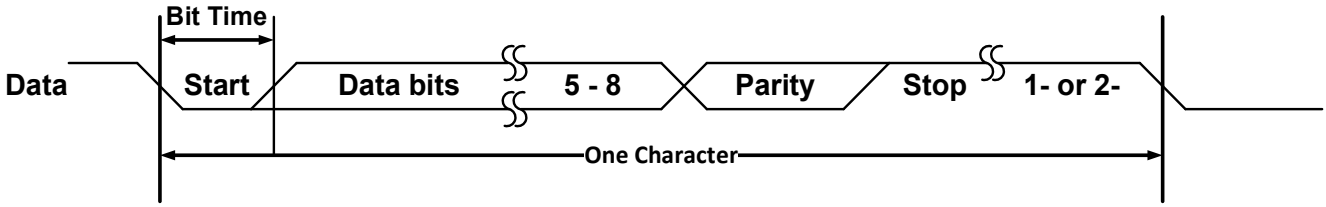


Figure 24. Serial data format

An additional parity bit may be added to the serial character. This bit appears between the last data bit and the stop bit(s) in the character structure. It provides UART with the ability to make simple error-checking on the received data.

The UART Line Control Register 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 or 2.

8.1.3 RS-485

Each UART can be configured to support the RS-485 serial protocol. When in RS-485 mode, an additional UART_TXDOE signal is provided to indicate the TXD active intervals. This signal can be assigned to any of the unused GPIO pins through the PPA.

8.1.4 Baud Rate

The clock that drives the UART block (FUARTCLK) is selectable between 32 kHz, 40 MHz, and 80 MHz depending on the system's operating mode.

The baud rate for each UART can be independently set by configuring its input clock divider with a divisor value consisting of an integer part and a fractional part.

These divisor values can be calculated:

baud rate divisor = (FUARTCLK/(16 × Baud rate))

The resulting baud rate devisor consists of an integer part and a fractional part.

The integer part is calculated:

integer part = integer (baud rate divisor)

Fractional part is calculated:

fractional part = integer ((baud rate divisor × 64) + 0.5)

8.1.5 Hardware Flow Control

The hardware flow control feature is fully selectable, and serial data flow is controlled by using RTS output and CTS input signals. Figure 25 shows how the two different UARTs can communicate using hardware flow control.

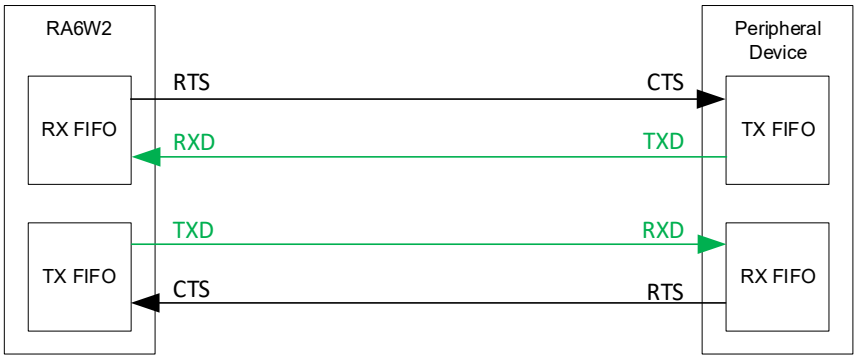


Figure 25. UART hardware flow control

When RTS flow control is enabled, RTS signal is asserted until the receive FIFO is filled up to programmed level. When CTS flow control is enabled, the transmitter can transmit the data when the CTS signal is asserted.

There are four modes of operation for flow control:

- Flow control is disabled
- Both RTS and CTS flow control are enabled
- Only CTS flow control is enabled
- Only RTS flow control is enabled.

8.1.6 Interrupts

Each UART has an associated interrupt in the interrupt vector table which is triggered by the following conditions:

- Overrun error
- Break error
- Parity error
- Framing error
- Receive timeout
- Transmit complete (based on FIFO trigger levels)
- Receive complete (based on FIFO trigger levels).

When an interrupt is triggered, the ISR checks the condition in the respective UART interrupt status register and then the appropriate action is taken.

8.1.7 DMA Interface

Each UART supports a connection to the GP-DMA so that transfers over the UART interface can be performed with minimal CPU overhead. Each UART supports a GP-DMA channel for both the receive and transmit functions.

8.1.8 Pin Configuration

The UART pins can be assigned to any of the unused GPIO pins through the Programmable Pin Assignment (PPA) function. Table 38 shows the pin definitions for the UART interfaces. The PPA provides a multiplexing function for the I/O pins of the on-chip peripherals. Any of the peripheral input or output signals can be freely mapped to any available GPIO port.

Table 38: UART pin configuration

Pin name	Default pin assignment	I/O
UART_RX	PPA	I
UART_TX	PPA	O
UART_CTS	PPA	I
UART_RTS	PPA	O

Pin name	Default pin assignment	I/O
UART_TXDOE	PPA	I/O
UART1_RX	PPA	I
UART1_TX	PPA	O
UART1_CTS	PPA	I
UART1_RTS	PPA	O
UART1_TXDOE	PPA	I/O
UART2_RX	PPA	I
UART2_TX	PPA	O
UART2_CTS	PPA	I
UART2_RTS	PPA	O
UART2_TXDOE	PPA	I/O

8.2 I2C Interface

8.2.1 Introduction

The I2C Interface is a bus that provides communications link between various peripheral devices 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 consisting of a serial data line (SDA) and a serial clock (SCL)
- Three speed modes are supported:
  - Standard mode (0 to 100 kbit/s)
  - Fast mode (<= 400 kbit/s)
  - High speed mode (<= 3.4 Mbit/s)
- Clock synchronization
- 32 entry deep transmit and receive FIFOs (32 x 8-bit RX, 32 x 10-bit TX)
- Master transmit, Master receive operation
- 7-bit or 10-bit addressing
- 7-bit or 10-bit combined format transfers
- Bulk transmit mode
- Configurable slave address (Default address of 0x055)
- Interrupt or Polled mode of operation
- Handle Bit and Byte waiting at both bus speeds
- Programmable SDA hold time
- DMA support.

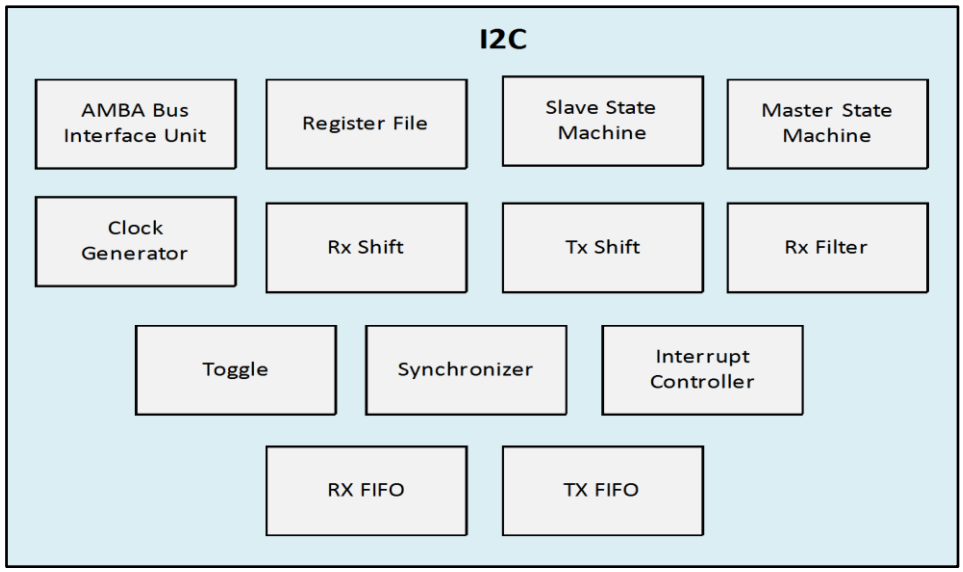


Figure 26. I2C Controller block diagram

8.2.2 I2C Behavior

The I2C can be configured 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 receiving data from and sending response data to the master. Data acknowledgement is sent by the device that receives data, which can be master or slave. The I2C protocol allows multiple masters to reside on the I2C bus. It 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 whether the master transmits data or receives data from the slave. The slave then sends an acknowledge pulse (ACK) 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 shown in Figure 27.

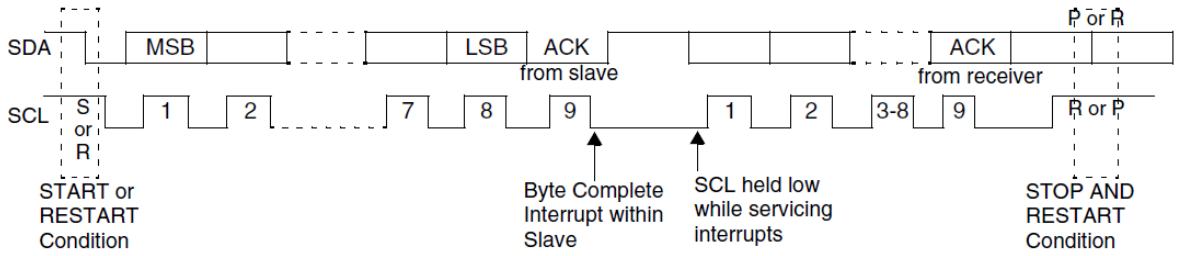


Figure 27. Data transfer on I2C bus

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.

8.2.2.1 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. Writing a 1 to I2C_DATA_CMD_REG causes the I2C to generate a STOP condition on the I2C bus; a STOP condition is not issued if this bit is not set, even if the transmit FIFO is empty.

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.

8.2.2.2 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 the other way round – 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.

8.2.3 I2C Protocols

The I2C Controller has the following protocols:

- START and STOP Conditions
- Addressing Slave Protocol
- Transmitting and Receiving Protocol
- START Byte Transfer Protocol.

8.2.3.1 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 as 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 28 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.

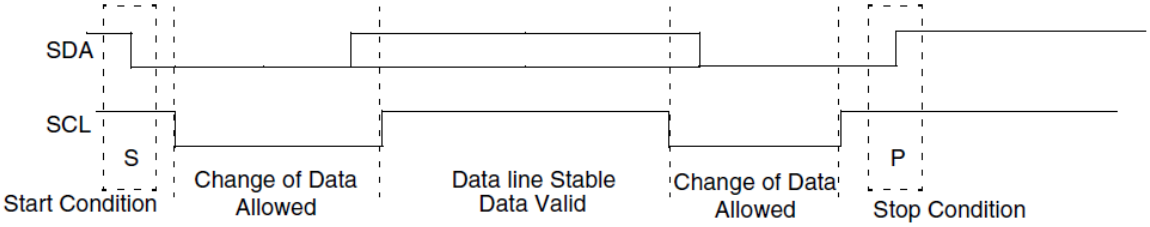


Figure 28. START and STOP conditions

NOTE
The signal transitions for the START/STOP conditions (see Figure 28) 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.

8.2.3.2 Addressing Slave Protocol

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

8.2.3.2.1 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 29. 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.

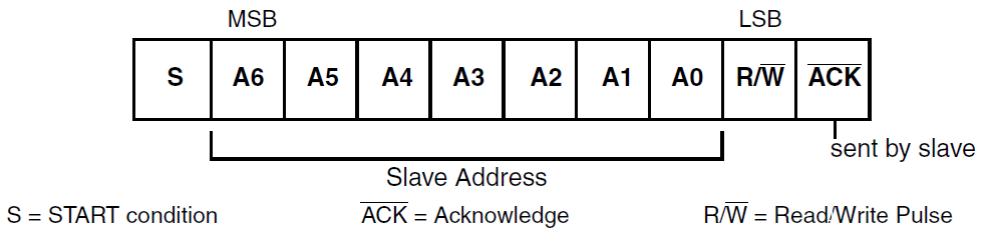


Figure 29. 7-bit address format

8.2.3.2.2 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 30 shows the 10-bit address format, and Table 39 defines the special purpose and reserves first byte addresses.

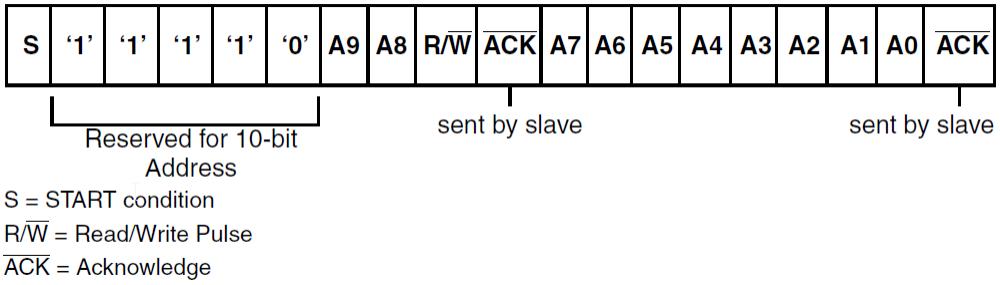


Figure 30. 10-bit address format

Table 39: I2C definition of bits in first byte

Slave address	R/W bit	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, see Section 8.2.3.3.3 "START BYTE Transfer Protocol".
0000 001	X	CBUS address. I2C Controller ignores these accesses.
0000 010	X	Reserved (Note 1)
0000 011	X	Reserved (Note 1)
0000 1XX	X	High-speed master code. For more information, see Section 8.2.4 "Multiple Master Arbitration".
1111 1XX	X	Reserved (Note 1)
1111 0XX	X	10-bit slave addressing.

Note 1 Use of the reserved addresses may cause incompatibilities with other I2C devices.

8.2.3.3 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.

8.2.3.3.1 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 31, then the slave-receiver responds to the master-transmitter with an acknowledge pulse after every byte of data is received.

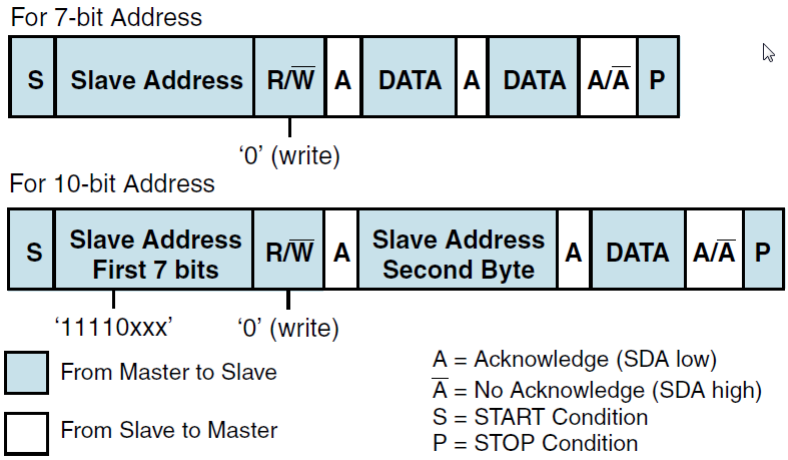


Figure 31. Master-Transmitter protocol

8.2.3.3.2 Master-Receiver and Slave-Transmitter

If the master is receiving data as shown in Figure 32, 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.

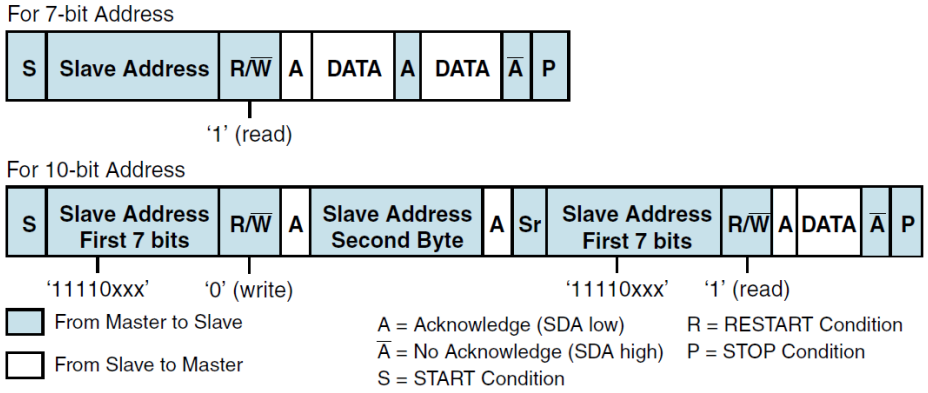


Figure 32. Master-Receiver protocol

8.2.3.3.3 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 shown in Figure 33. This allows the processor that is polling the bus to under-sample the address phase until 0 is detected. When the microcontroller detects a 0, it switches from the under-sampling rate to the correct rate of the master.

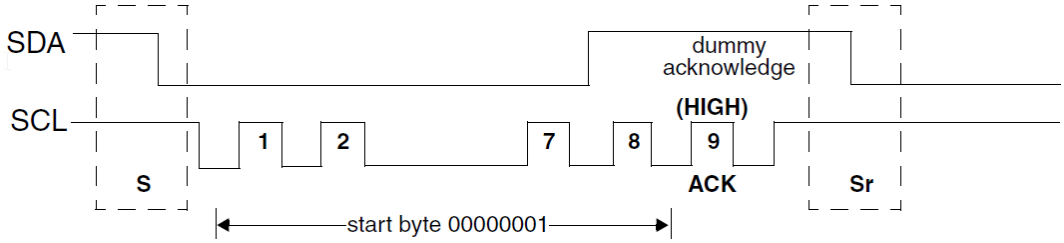


Figure 33. START byte transfer

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.

8.2.4 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. When 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 might go into the data phase. Figure 34 shows 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 at 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 or 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.

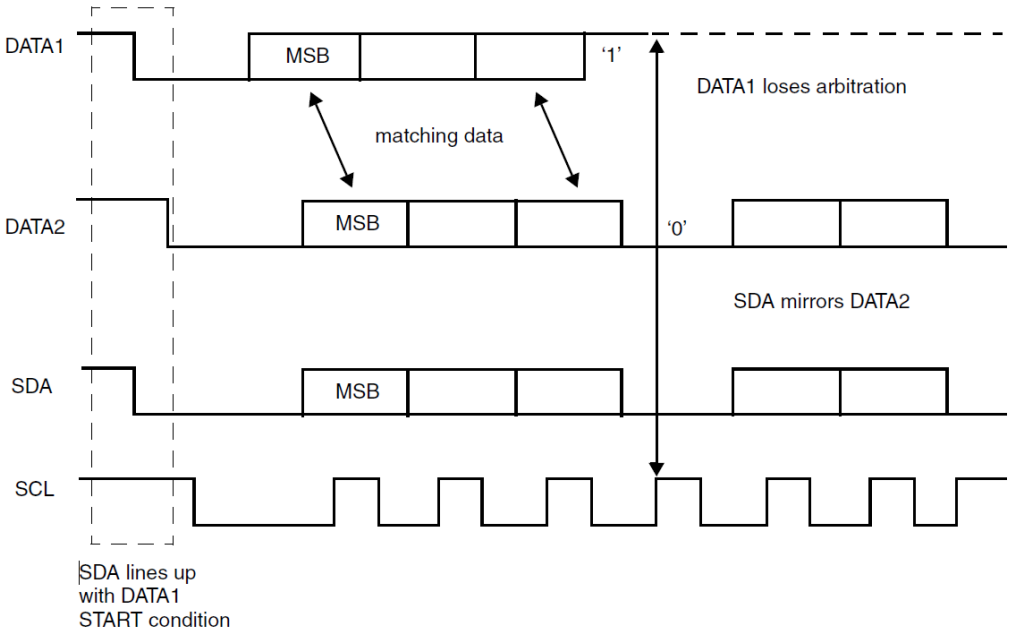


Figure 34. Multiple Master arbitration

8.2.5 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 count out their low time. The one with the longest low time, forces the other master into a HIGH wait state. Therefore, a synchronized SCL clock is generated as shown in Figure 35. Optionally, slaves may hold the SCL line low, to slow down the timing on the I2C bus.

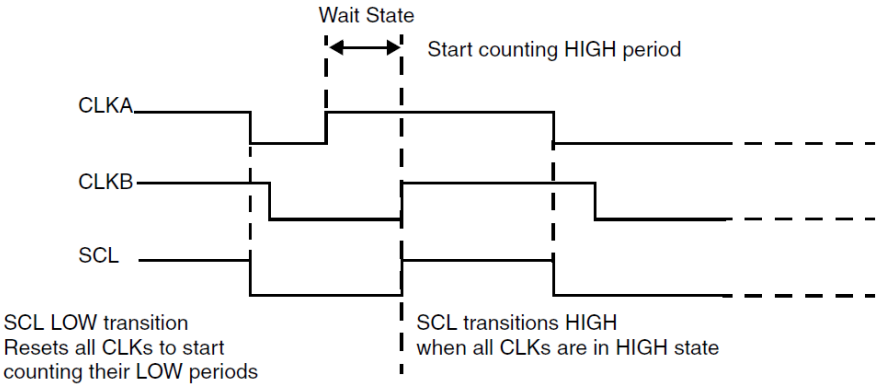


Figure 35. Multiple master clock synchronization

8.3 Digital Audio Interface (I2S and PDM)

8.3.1 Introduction

The Digital Audio Interface (DAI) is a 4-wire synchronous audio interface to audio Codecs. The I2S mode supports master or slave operation for stereo audio with two 32-bit channels. The PDM mode supports master or slave operation for sample rates of 24, 32, or 48 kHz.

Features

- I2S Master and Slave mode
- Sample rates: 8, 12, 16, 24, 32, 44.1, and 48 kHz
- Data word length: 16, 20, 24, or 32 bits
- Frame length: 32, 64, 128, 256 bits wide
- Maximum number of slots: 16
- Number of channels: 2
- Sample Rate Converter (SRC)
- Formats: Digital signal processing, left-justified, right-justified, and I²S
- Time division multiplexed mode, with push-pull in active and Hi-Z is non active slots
- Programmable active rising/falling edges of PCM_WCLK and PCM_BCLK
- Supports Pulse Density Modulation (PDM) for interfacing with DMICs.

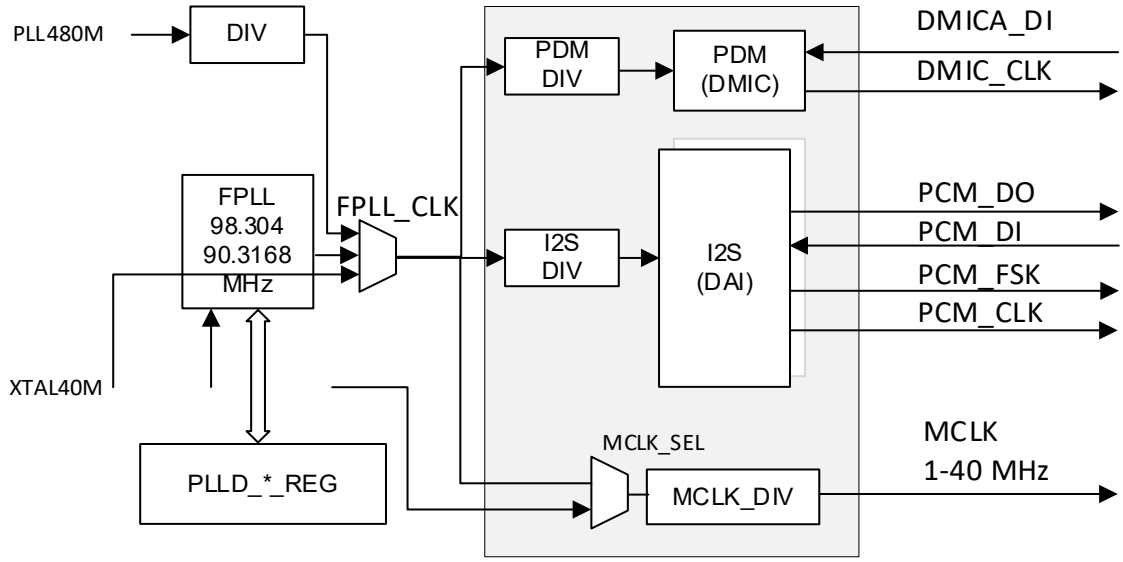


Figure 36. DAI block diagram

8.3.2 Interface Signals

The DAI is a four-wire serial interface. The pins and signals are mapped as shown in Table 40.

Table 40: DAI pins and signals

DAI signals	Description
PCM_BCLK	Bit clock (BCLK) INPUT/OUTPUT Maximum 24.576 MHz
PCM_WCLK	Word clock (WCLK) IN/OUT Maximum 192 kHz
PCM_DI	Input data (DATA_IN)

DAI signals	Description
PCM_DO	Output (DATA_OUT)
MCLK	Master clock (OUTPUT) to Codec (can be disabled if BCLK is selected as MCLK in the Codec
DMICA_DI	Input Data for the Digital Mic interface
DMIC_CLK	Clock for the Digital Mic interface

The MCLK signal is used to clock an external codec. This signal can be derived directly from the FPLL or XTAL40M. The PCM_DO output state can be configured according to Table 41.

Table 41: PCM_DO output states

Pxy_MODE_REG		DAI_DATA_OUT_CTRL_REG DAI2_DATA_OUT_CTRL_REG	PCM_DO state
PUPD	**PPOD**	**DATA_OUT_EN[1-0]**	
0,1,2	Don't care	Don't care	Hi-Z
3	Don't care	0 or 1	Hi-Z
3	0	2	0: Driving all slots (single master) 1: Driving all slots (single master)
3	0	3	**Active slot** 0: Driving 1: Driving **Non-active slot:** Hi-Z
3	1	3	**Active slot** 0: Driving 1: Open drain Hi-Z **Non-active slot:** Hi-Z

8.3.3 Master and Slave Modes

The DAI operates in either Master mode or Slave mode as shown in Figure 37.

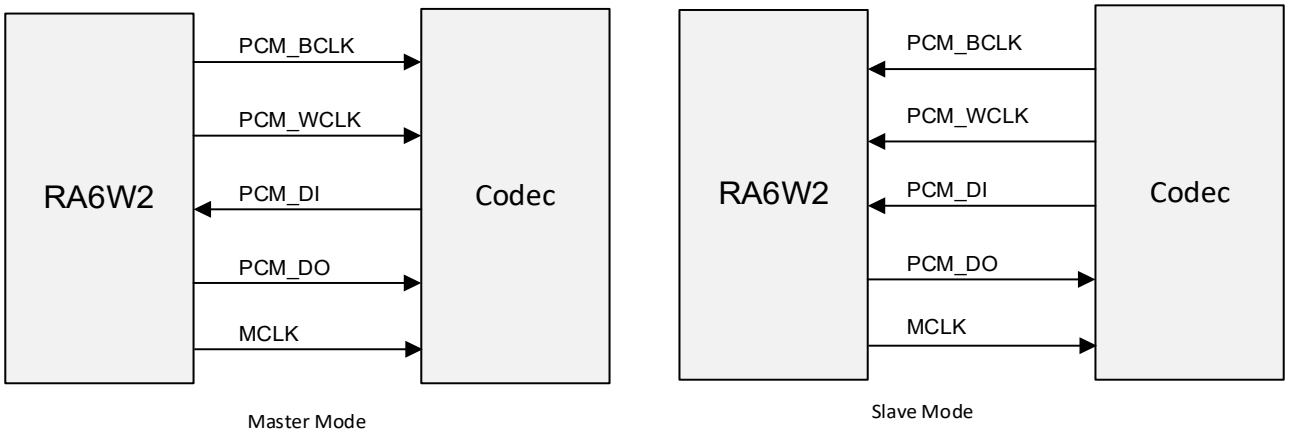


Figure 37. DAI Master and Slave modes

The bit clock (PCM_BCLK) samples receive data into the DAI through the PCM_DI pin and transmit through the PCM_DO pin. The word clock (PCM_WCLK) is the DAI data sample clock, synchronizing the sample frames for the DAI data channels.

In Master mode, the DAI provides synchronization clocks, PCM_BCLK and PCM_WCLK. In Slave mode, BCLK and WCLK must be provided externally.

8.3.4 DAI Slots

The DAI can serve up to 16 slots of 16 bits maximum and up to 8 slots of 32 bits maximum. The maximum frame length of DAI is 256 bits. A configurable slot offset determines the start of frame for Channel 1. The offset prevents conflict when two or more devices are on the bus.

8.3.5 DAI Slot Formats

8.3.5.1 I²S Format

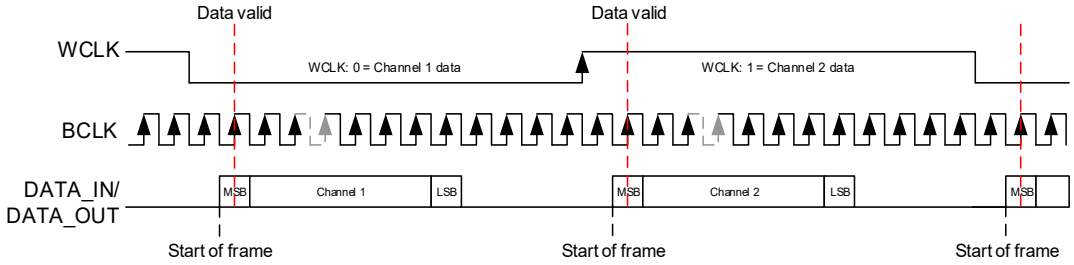


Figure 38. I²S format

In I²S format, the start of frame for Channel 1 is on the second falling edge of BCLK after a falling edge of WCLK. The MSB of Channel 1 is valid on the rising edge of BCLK after the start of frame condition.

The start of frame for Channel 2 is on the second falling edge of BCLK after a rising edge of WCLK. The MSB of Channel 2 is valid on the rising edge of BCLK after the start of frame condition.

8.3.5.2 DSP Format

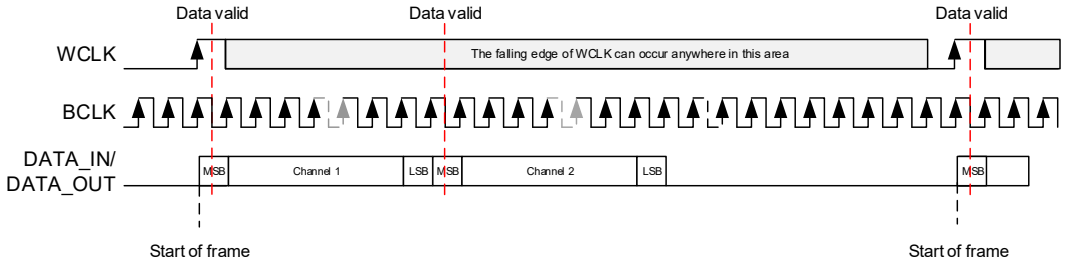


Figure 39. DSP format

In DSP format, the rising edge of WCLK starts the data transfer (start of frame) with the Channel 1 data first, immediately followed by Channel 2 data and any subsequent channels. Each data bit is valid on the falling edge of BCLK.

8.3.5.3 Left-Justified Format

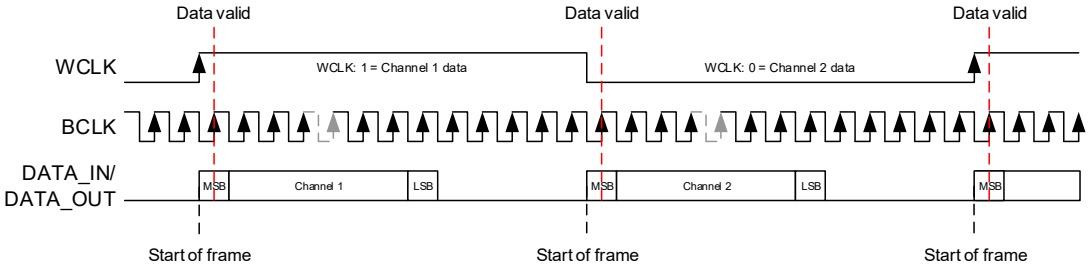


Figure 40. Left-justified format

In left-justified format (LJF), the MSB of Channel 1 is valid on the rising edge of BCLK following the rising edge of WCLK. The MSB of Channel 2 is valid on the rising edge of BCLK following the falling edge of WCLK.

8.3.5.4 Right-Justified Format

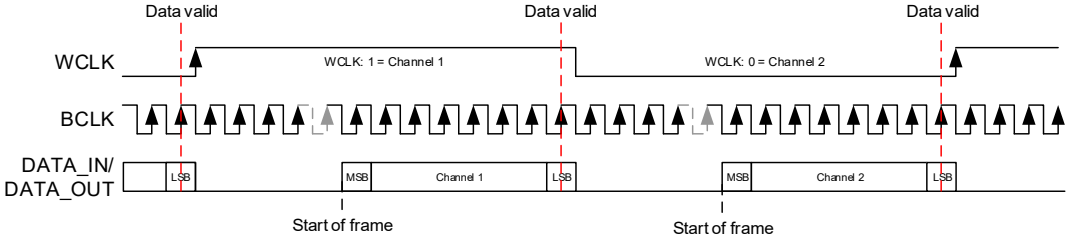


Figure 41. Right-justified format

In right-justified format (RJF), the LSB of Channel 1 is valid on the rising edge of BCLK preceding the falling edge of WCLK. The LSB of Channel 2 is valid on the rising edge of BCLK preceding the rising edge of the WCLK.

8.3.5.5 Time Division Multiplexing Mode

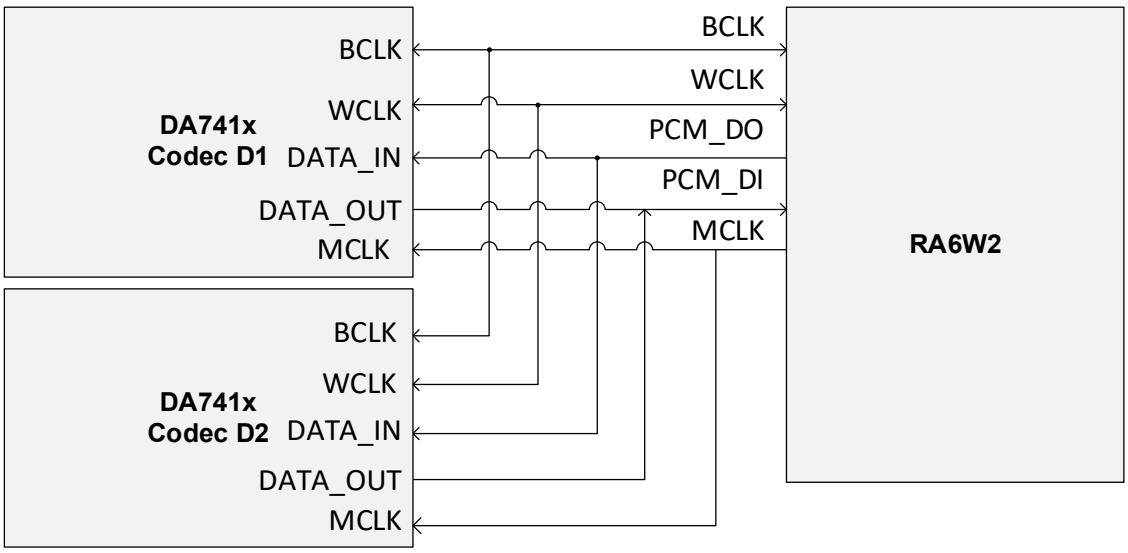


Figure 42. TDM configuration

Time division multiplexing (TDM) mode allows multiple devices to communicate on the same bus without conflicting, see Figure 42. The serial data pin is tri-stated whenever the output is not valid to allow other devices on the bus to drive the data line.

TDM mode is available in both Master and Slave modes. The TDM mode is an extension of LJF, see Figure 43 and for DSP format, see Figure 44.

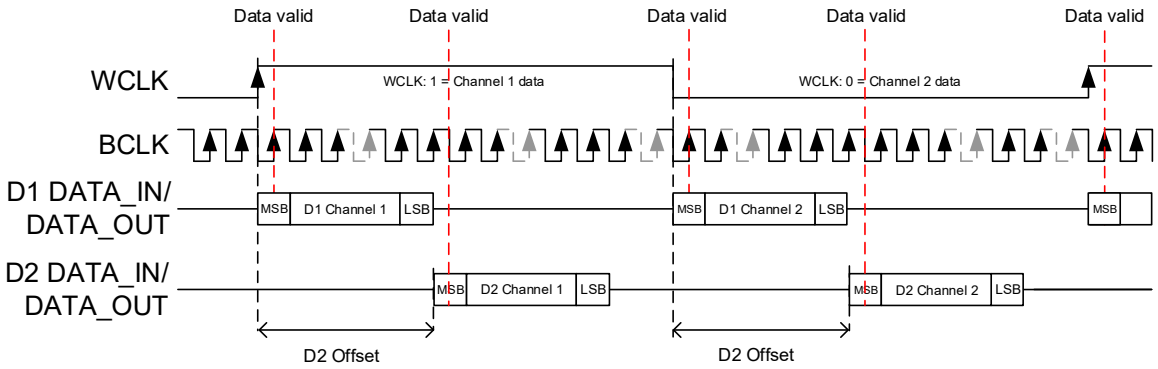


Figure 43. Two devices in LJF mode with TDM mode active

In LJF with TDM mode active, the Device 2 (D2) Channel 1 data is offset by a configurable number of BCLK cycles (D2 Offset) after the rising edge of WCLK. The D2 Channel 2 data is valid for the same number of BLCK cycles (D2 Offset) after the falling edge of WCLK.

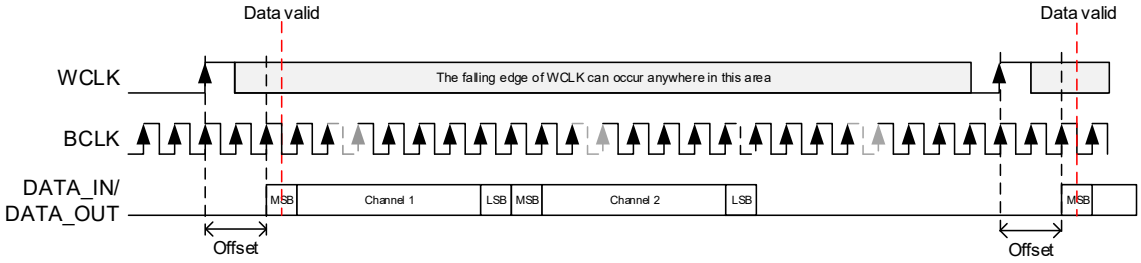


Figure 44. One device in DSP mode with offset from TDM mode

In DSP format with TDM mode active the start of frame is offset by a configurable number of BCLK cycles (Offset) from the rising edge of WCLK. The Channel 1 data is valid on the first falling edge of BCLK after the start of frame condition. Channel 2 data immediately follow Channel 1 data.

8.3.6 DAI Slot Assignment

Table 42 shows the slot assignment for the various formats depending on PCM_WCLK edge, WCLK_POL value, and SLOT_CNT up to 8 are depicted.

Table 42: DAI format summary

SLOT_CNT = 2										
WCLK_POL = 0	WCLKedge	T0	T1	T2	T3	WCLKedge	T4	T5	T6	T7
I2S	Falling	Slot1				Rising	Slot2			-
LJF	Rising	Slot1				Falling	Slot2			-
RJF	Rising				Slot1	Falling				Slot2
DSP	Rising	Slot1	Slot2							
WCLK_POL = 1										
I2S	Rising	Slot1				Falling	Slot2			
LJF	Falling	Slot1				Rising	Slot2			
RJF	Falling				Slot1	Rising				Slot2
DSP	Falling	Slot1	Slot2							

SLOT_CNT = 4										
WCLK_POL = 0	WCLKedge	T0	T1	T2	T3	WCLKedge	T4	T5	T6	T7
I2S	Falling	Slot1	Slot3			Rising	Slot2	Slot4		
LJF	Rising	Slot1	Slot3			Falling	Slot2	Slot4		
RJF	Rising			Slot1	Slot3	Falling			Slot2	Slot4
DSP	Rising	Slot1	Slot2	Slot3	Slot4					
WCLK_POL = 1										
I2S	Rising	Slot1	Slot3			Falling	Slot2	Slot4		
LJF	Falling	Slot1	Slot3			Rising	Slot2	Slot4		
RJF	Falling			Slot1	Slot3	Rising			Slot2	Slot4
DSP	Falling	Slot1	Slot2	Slot3	Slot4					
SLOT_CNT = 8										
WCLK_POL = 0	WCLKedge	T0	T1	T2	T3	WCLKedge	T4	T5	T6	T7
I2S	Falling	Slot1	Slot3	Slot5	Slot7	Rising	Slot2	Slot4	Slot6	Slot8
LJF	Rising	Slot1	Slot3	Slot5	Slot7	Falling	Slot2	Slot4	Slot6	Slot8
RJF	Rising	Slot1	Slot3	Slot5	Slot7	Falling	Slot2	Slot4	Slot6	Slot8
DSP	Rising	Slot1	Slot2	Slot3	Slot4		Slot5	Slot6	Slot7	Slot8
WCLK_POL = 1										
I2S	Rising	Slot1	Slot3	Slot5	Slot7	Falling	Slot2	Slot4	Slot6	Slot8
LJF	Falling	Slot1	Slot3	Slot5	Slot7	Rising	Slot2	Slot4	Slot6	Slot8
RJF	Falling	Slot1	Slot3	Slot5	Slot7	Rising	Slot2	Slot4	Slot6	Slot8
DSP	Falling	Slot1	Slot2	Slot3	Slot4		Slot5	Slot6	Slot7	Slot8
Applying offset (DAI_OFFSET_LSB_REG, DAI_OFFSET_MSB_REG]										
WCLK_POL = 0	WCLKedge	T0	T1	T2	T3	WCLKedge	T4	T5	T6	T7
I2S	Falling	Slot1				Rising	Slot2			
I2S (Offset)	Falling		Slot1			Rising		Slot2		

8.3.7 DAI PCM_CLK Generation

In Master mode, the Slot length can be programmed to be 16, 20, 24, or 32 bits. If the slot length is 24 bits and multiple slots are used, the slots are concatenated without padding bit. If padding bits are required (for example, for a DAI tester) use W_LEN is 32 bits and make the 8 LSB bits 0. In TDM mode with open drain selected, the 8 LSB bits can be set to 1 to allow other bus masters to use these slots.

The MSB of register or SRC input/output is first transmitted and received and the remaining bits in LSBs are not transmitted and received if word length is not equal to 32 bits wide.

In Slave mode, the DAI automatically detects the frame length (number of BLCKs per WCLK). In Master mode, the frame length is configurable to be 32, 64, 128, or 256 bits wide.

The PCM_CLK is automatically set to a maximum of 24.576 MHz, according to this formula:

- For PCM_CLK = DAI_SR * FRAME_LEN

The maximum number of possible slots is:

- For DSP format FRAME_LEN/W_LEN
- For I2S, LJF and RJF format for int(FRAME_LEN/(2*W_LEN))*2

Table 43 and Table 44 show typical examples of PCM clock settings. PCM_DIV has a mandatory value of 1.

Table 43: PCM clock generation examples (FPLL = 98.304 MHz, AUD_CLK_DIV = 4, PCM_DIV =1)

DAI_SR	FRAME_LEN	W_LEN	SLOT_CNT	PCM_CLK (kHz) (Auto set)	PCM_DIV (Mandatory)
0xB: 48 kHz	1: 64 bits	3: 32 bits	2	3072	1
0xB: 48 kHz	2: 128 bits	3: 32 bits	2	6144	1

Table 44: PCM generation example (FPLL = 90.3168 MHz, AUD_CLK_DIV = 4, PCM_DIV = 1)

DAI_SR	FRAME_LEN	W_LEN	SLOT_CNT	PCM_CLK (kHz) (Auto set)	PCM_DIV (Mandatory)
0x0A: 44.1 kHz	1: 64 bits	3: 32 bits	2	2822.4	1

8.4 SPI Master/Slave

8.4.1 Introduction

The Serial Peripheral Interface (SPI™) supports master and Slave modes of operation. The serial interface can transmit and receive from 4 to up to 32 bits in Master/Slave mode. The controller comprises separate TX and RX FIFOs and DMA handshake support. Slave mode clock speed is independent from the system clock speed. The controller can generate an interrupt upon data threshold reached in the TX or RX FIFOs.

Features

- Slave and Master mode for SPI/SPI2
- From 4-bit to up to 32-bit operation
- SPI/SPI2 SPI_CLK Master mode clock line up to 48 MHz
- SPI mode 0, 1, 2, and 3 support (clock edge and phase)
- Built-in separate 8-bit wide and 32-byte deep RX/TX FIFOs for continuous SPI bursts (SPI, SPI2)
- Maskable interrupt generation based on TX or RX FIFO thresholds
- DMA support.

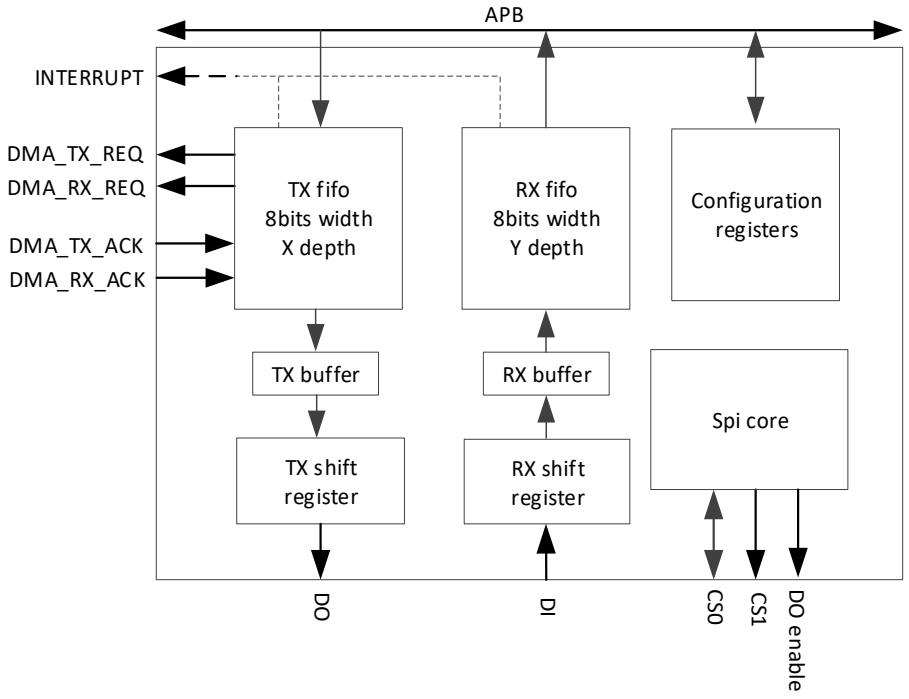


Figure 45. SPI block diagram

The SPI controller is responsible for the serialization/deserialization of the data in the RX and TX streams.

Two separate FIFOs are used to store data for RX and TX streams. Since a SPI word can be configured to be from four bits to up to 32 bits, one to four FIFO positions can be written/read at the same time. FIFOs contain logic implementing programmable thresholds comparison.

The SPI controller supports DMA requests and interrupt generation based on the FIFO thresholds. If enabled, a DMA request and/or interrupt is asserted with whether TX_FIFO level is low, or RX_FIFO level is high.

The SPI interface supports all four modes of operation and the corresponding polarity (CPOL) and phase (CPHA) of the SPI clock (SPI_CLK) are defined in Table 45.

Table 45: SPI modes configuration and SCK states

SPI mode	CPOL	CHPA	TX SPI_CLK	RX SPI_CLK	Idle SPI_CLK
0	0	0	Falling edge	Rising edge	Low
1	0	1	Rising edge	Falling edge	Low
2	1	0	Rising edge	Falling edge	High
3	1	1	Falling edge	Rising edge	High

To read from or to write to an external single byte Flash device in the SPI Master mode, a byte swap mechanism is implemented to allow for a proper placement of the bytes in a 16-bit word for the DMA to write to/read from the internal RAM. More specifically, when the SPI controller is configured as a master with DMA support and a 16-bit word width so that the bus utilization is increased compared to reading from an 8-bit device, the byte swap mechanism brings the least significant byte read and place it in the most significant byte in the 16-bit word. The controller automatically swaps the bytes to allow for placing the first byte read in the least significant byte of the 16-bit word.

8.4.2 SPI Timing

The SPI interface timing for Master and Slave modes is shown in Figure 46.

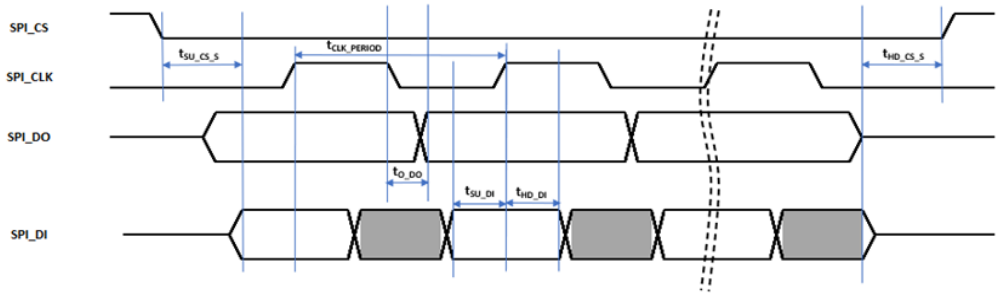


Figure 46. SPI timing (CPOL = 0, CPHA = 0)

Table 46: SPI timing parameters

Description	Parameter	Condition	Min	Typ	Max	Unit
SPI_CLK clock frequency (I/O = 3.3 V)	$t_{CLK\_M\_3V3}$	Master mode			48	MHz
SPI_CLK period (I/O = 3.3 V)	$t_{CLK\_PERIOD\_3V3}$	Master mode	20.84			ns
SPI_CLK clock frequency (I/O = 1.8 V)	$t_{CLK\_M\_1V8}$	Master mode			40	MHz
SPI_CLK period (I/O = 1.8 V)	$t_{CLK\_PERIOD\_1V8}$	Master mode	25			ns
SPI_CLK duty	t_{CLK_DUTY}	Master mode		50		%
SPI_DO Output delay	$t_{O\_DO\_M}$	Master mode	2		4	ns
SPI_DI Setup time	$t_{SU\_DI\_M}$	Master mode	9/4 (Note 1)			ns
SPI_DI Hold time	$t_{HD\_DI\_M}$	Master mode	5			ns
SPI_CLK clock (I/O = 3.3 V)	$t_{CLK\_S\_3V3}$	Slave mode			48	MHz
SPI_CLK clock (I/O = 1.8 V)	$t_{CLK\_S\_1V8}$	Slave mode			40	MHz
SPI_CS Setup time before first transfer	$t_{SU\_CS\_S}$	Slave mode	5			ns

Description	Parameter	Condition	Min	Typ	Max	Unit
SPI_CS Hold time after last transfer	$t_{HD\_CS\_S}$	Slave mode	5			ns
SPI_DO Output delay	$t_{O\_DO\_S}$	Slave mode			8.5	ns
SPI_DI Setup time	$t_{SU\_DI\_S}$	Slave mode	3			ns
SPI_DI Hold time	$t_{HD\_DI\_S}$	Slave mode	2			ns

Note 1 Use next edge decoding.

8.5 SDIO

8.5.1 Introduction

RA6W2 supports an SDIO 3.0 card interface suitable for memory card and I/O card applications with low-power consumption. The SDIO interface supports SPI, 1-bit SD, and 4-bit SD transfer modes at the full clock range of 0 to 80 MHz. The CIS and CSA areas are located inside the internal memory and the SDIO registers (CCCR and FBR) are programmed by the SD host.

Features

- Compliant with SDIO Specification 3.0
- Enhanced power management
- Support Read Wait Control, Suspend/ Resume operations for superior card performance
- Support SPI, 1-bit and 4-bit SD modes
- Support all SDIO form factors including standard, mini, and micro SDIO card
- Embedded SDIO ATA interface code
- Bus Master with Scatter Gather DMA
- Cyclic Redundancy Check (CRC7) (command), CRC16 (data) integrity
- Support direct R/W (IO52) and extended R/W (IO53).

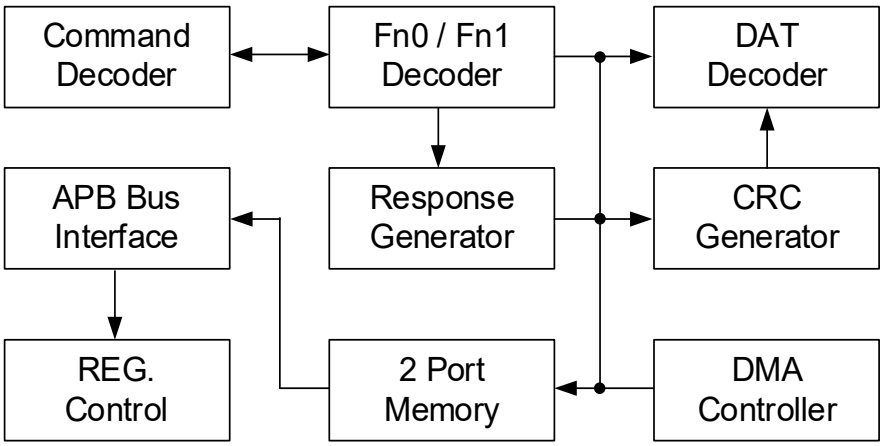


Figure 47. SDIO slave block diagram

The SDIO interface is assigned to specific pins because of the performance requirements of the pin. There are two configurations for the assignment of the SDIO pins (alternate 1 and alternate 2). All pins can also be assigned as either SDIO function or as a peripheral pin through the PPA. The pin configurations for the SDIO interface are shown in Table 47.

Table 47: SDIO pin configuration

Pin name	Pin assignment (Alt 1/Alt 2)	I/O	Description
SDIO_CMD	P1_10/P0_09	I/O	Command line
SDIO_CLK	P1_11/P0_08	I	Input serial clock

Pin name	Pin assignment (Alt 1/Alt 2)	I/O	Description
SDIO_D0	P1_12/P0_10	I/O	Bidirectional data line
SDIO_D1	P1_13/P0_11	I/O	Bidirectional data line
SDIO_D2	P1_14/P0_12	I/O	Bidirectional data line
SDIO_D3	P1_15/P0_13	I/O	Bidirectional data line

Figure 48 shows the timing diagram for the SDIO slave.

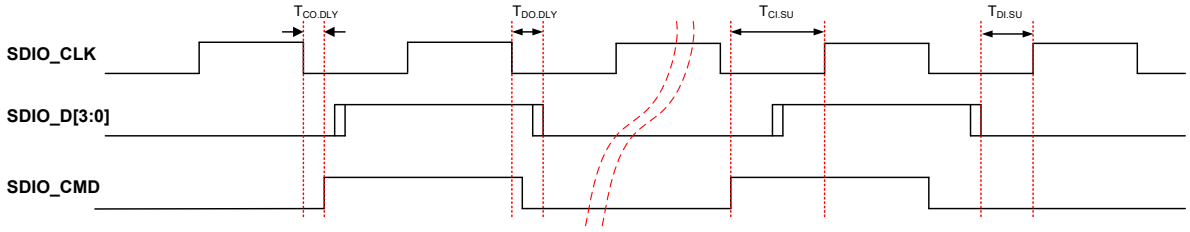


Figure 48. SDIO slave timing diagram

Table 48 lists the timing parameters for the SDIO slave.

Table 48: SDIO slave timing parameters

Parameter	Symbol	Min	Typ	Max	Unit
SDIO_CLK frequency	F_{SCLK}			80	MHz
SDIO_CLK clock duty			50		%
SDIO_CMD input setup time	$T_{CI.SU}$	3			ns
SDIO_CMD output delay time	$T_{CO.DLY}$			11 (Note 1)	ns
SDIO_D[3:0] input setup time	$T_{DI.SU}$	3			ns
SDIO_D[3:0] output delay time	$T_{DO.DLY}$			11 (Note 1)	ns

Note 1 SDIO signals can set previous output from half cycle.

The SDIO interface requires pull-up resistors to be connected between the signal lines and the supply to enable communication.

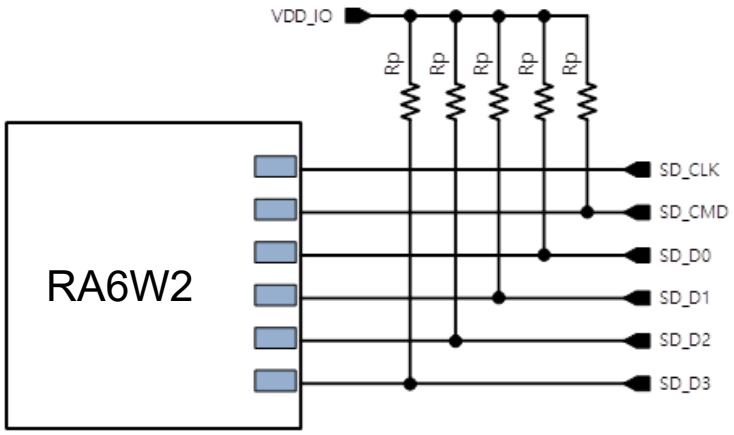


Figure 49. SDIO pull-up resistor

Pull-up resistor values may vary based on the board layout.

8.6 SD/eMMC Host Controller

8.6.1 Introduction

The SD/eMMC host interface of the RA6W2 provides access to SD or eMMC memory cards. The SD/eMMC host interface supports a 4-bit data bus with a maximum clock rate of 80 MHz giving a maximum data rate of 640 Mbps for octa mode, and 320 Mbps for quad mode.

Figure 50 shows the block diagram of the SD/eMMC host interface including the control register, clock control, command/response pipe, data pipe, and AHB master interface blocks.

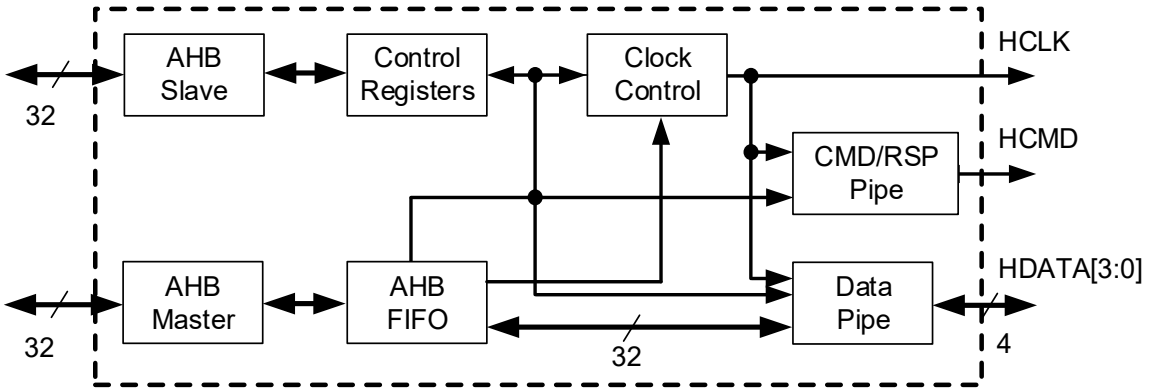


Figure 50. SD/eMMC block diagram

Figure 51 shows the timing diagram for the SD/eMMC master.

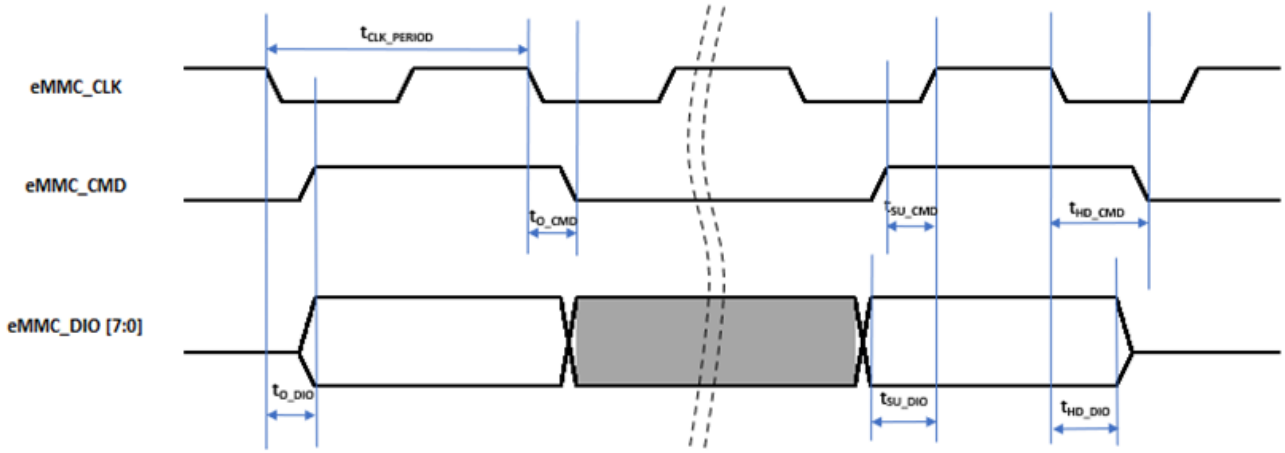


Figure 51. SD/eMMC host timing diagram

Table 49 lists the timing parameters for the SD/eMMC master.

Table 49: SD/eMMC host timing parameters

Description	Parameter	Min	Typ	Max	Unit
eMMC_CLK clock frequency	t_{CLK_M}			80	MHz
eMMC_CLK period	t_{CLK_PERIOD}	12.5			ns
eMMC_CLK duty	t_{CLK_DUTY}		50		%
eMMC_CMD Output delay	t_{O_CMD}	1		2	ns
eMMC_DIO Output delay	t_{O_DIO}	1		2	ns
eMMC_CMD Setup time	t_{SU_CMD}	3			ns
eMMC_CMD Hold time	t_{HD_CMD}	0			ns
eMMC_DIO Setup time	t_{SU_DIO}	2			ns

Description	Parameter	Min	Typ	Max	Unit
eMMC_DIO Hold time	t_{HD_DIO}	0			ns

The SD/eMMC interface is assigned to specific pins because of the performance requirements of the pin. There are two configurations for the assignment of the SD/eMMC pins (alternate 1 and alternate 2). All pins can also be assigned as either SD/eMMC function or as a peripheral pin through the PPA. The pin configurations for the SD/eMMC interface are shown in Table 50.

Table 50: SD/eMMC pin configuration

Pin name	Pin assignment (Alt 1/Alt 2)	I/O	Description
eMMC_CMD	P1_10/P0_09	O	Output serial clock
eMMC_CLK	P1_11/P0_08	I/O	Command line
eMMC_DIO0	P1_12/P0_10	I/O	Bi-directional data line
eMMC_DIO1	P1_13/P0_11	I/O	Bi-directional data line
eMMC_DIO2	P1_14/P0_12	I/O	Bi-directional data line
eMMC_DIO3	P1_15/P0_13	I/O	Bi-directional data line
eMMC_DIO4	P1_00/P0_04	I/O	Bi-directional data line
eMMC_DIO5	P1_01/P0_05	I/O	Bi-directional data line
eMMC_DIO6	P1_02/P0_06	I/O	Bi-directional data line
eMMC_DIO7	P1_03/P0_07	I/O	Bi-directional data line

8.7 Octa/Quad SPI Flash Controller – With Secure XIP

8.7.1 Introduction

The Octa/Quad SPI Controller (OQSPIC) provides a low pin count interface to standard Serial Peripheral Interface (SPI) and a high-performance Dual/Quad/Octa SPI Interface.

In Automatic mode, the OQSPIC provides transparent access octa/quad flash memory for Execute-In-Place (XIP). In combination with the CPU cache, it provides comparable performance to executing code from standard parallel Flash.

In Manual mode, the serial Flash memory can be accessed by the memory mapped registers of OQSPIC. All instructions supported by Flash memory can be programmed using the registers of the OQSPIC.

Features

- SPI modes:
  - Single: Data transfer using two unidirectional pins
  - Dual: Data transfer using two bidirectional pins
  - Quad: Data transfer using four bidirectional pins
  - Octa: Data transfer using eight bidirectional pins (VFBGA package only)
- Auto mode: up to 64 MB transparent code access for XIP (Execute-In-Place) and Data access with 3-byte and 4-byte addressing modes.
- Manual mode: Direct register access using the QSPIC register file.
- Up to 80 MHz QSPI clock. Clock modes 0 and 3. Master mode only.
- Vendor independent Instruction Sequencer.
- Support for single access and high-performance burst mode in combination with the cache controller (in Auto mode).
- Use of special read instruction for a specific (programmable) wrapping burst access.
- Erase suspend/resume to support for code and data storage.
- Decrypt on-the-fly (AES-256b-CTR) capability while in Auto mode operation.

- Retention mode for I-cache memory.

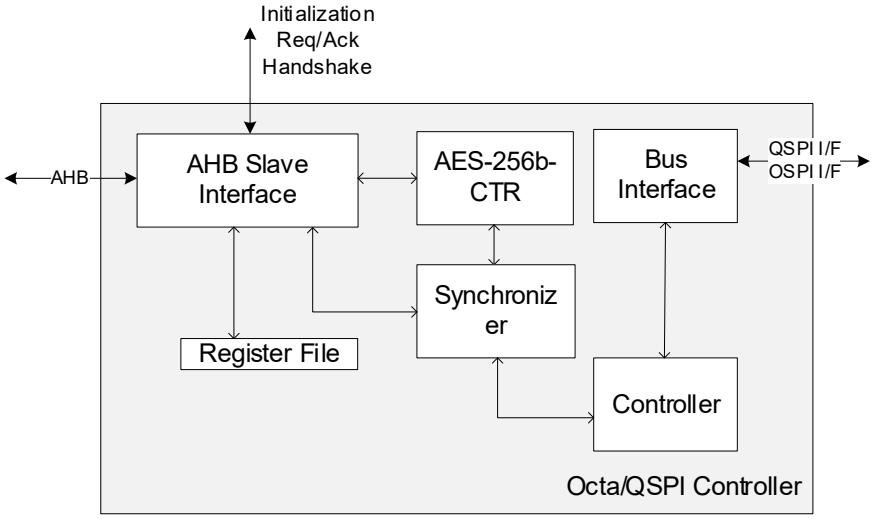


Figure 52. OQSPI flash controller block diagram

The OQSPIC implements all protocols related to the functionality of Flash memory. It contains a finite state machine (FSM) that generates all necessary signaling to the OQSPI bus and realizes all features of the Auto mode operation. Moreover, it manages all data transfers between the two interfaces (the AHB and the OQSPIC).

The Bus Interface block controls the OQSPIC signals at the lowest level while the Synchronizer implements "stretching" or "shortening" of the signals that cross two clock domains.

8.7.1.1 Interface

The OQSPIC interface is assigned to specific pins because of the performance requirements of the pin. All pins can be assigned as either OQSPI function or as a peripheral pin through the PPA. The pin configurations for the OQSPIC interface are shown in Table 51.

Table 51: OQSPI pin configuration

Pin name	Pin assignment	I/O	Description
OQSPI_CLK	P1_08	O	Output serial clock
OQSPI_CS	P1_09	O	Active low output Chip select
OQSPI_D0	P1_04	I/O	▪ MOSI (output) in single SPI mode ▪ D0 (bidirectional) in Quad/Octa SPI mode
OQSPI_D1	P1_05	I/O	▪ MISO (input) in single SPI mode ▪ D1 (bidirectional) in Quad/Octa SPI mode
OQSPI_D2	P1_06	I/O	▪ WPn Write Protect output in single SPI mode ▪ D2 (bidirectional) in Quad/Octa SPI mode
OQSPI_D3	P1_07	I/O	▪ HOLDn/Resetn output in Single SPI mode ▪ D3 (bidirectional) at Quad/Octa SPI mode
OQSPI_D4	P1_00	I/O	D4 (bidirectional) at Octa SPI mode
OQSPI_D5	P1_01	I/O	D5 (bidirectional) at Octa SPI mode
OQSPI_D6	P1_02	I/O	D6 (bidirectional) at Octa SPI mode
OQSPI_D7	P1_03	I/O	D7 (bidirectional) at Octa SPI mode

The OQSPIC drives all data pins constantly except for the case when a read is performed. The time for changing the direction of the pads is at least 1.5 x OQSPIF_CLK (OQSPIF_CLK being the clock that the Flash operates on). In this way, data lines are always terminated, thus reducing unnecessary power consumption.

The default state of the OQSPIF_IOx pins is 1. This state is applied to the pins as soon as the OQSPIC clock is enabled even if no access to the external Flash has yet been triggered. This value is valid only after the OQSPIF_CS is pulled low (an access to the external Flash occurs).

8.7.1.2 SPI Modes

The OQSPIC supports the following SPI standards:

- Single: Data transfer using two unidirectional pins. The OQSPIC supports communication with any single/dual/quad or octa SPI Flash memory. However, the Single SPI interface does not support bus modes 1 and 2, full-duplex communication, or any SPI Slave mode.
- Dual: Data transfer using two bidirectional pins.
- Quad: Data transfer using four bidirectional pins.
- Octa: Data transfer using eight bidirectional pins.

8.7.1.3 Access Modes

The serial Flash connected to the OQSPIC can be accessed in one of the modes:

- Auto mode
- Manual mode.

These modes are mutually exclusive. The serial Flash can operate only in one of the two modes. In Auto mode, 3-byte and 4-byte addressing modes are supported. With OQSPIF_CTRLMODE_REG[QSPIC_USE_32BA] = 0, up to 16 MB OQSPIC (3-byte addressing) can be accessed. If OQSPIF_USE_32BA = 1, 4-byte addressing is enabled for accessing up to 64 MB OQSPIC Flash.

8.7.1.3.1 Auto Mode

In Auto mode, read access from the serial Flash memory is fully transparent to the CPU. A read access at the interface is translated by the OQSPIC into the respective SPI bus control commands needed for the Flash memory access. When the Auto Mode is disabled, any access (reading or writing) is ignored. When Auto mode is enabled, only read access is supported. Write access causes hard faults. A read access can be single access, incremental burst, or wrapping burst. Wrapping burst is supported even when the Flash device does not support any special instruction for wrapping burst. Special read instruction can be used for specific (programmable) wrapping burst access. When Flash supports special instruction for wrapping burst access, it reduces access time (less wait states). For maximizing the utilization of the bus and minimizing the number of wait states, it is recommended to use burst access. However, non-sequential random access is supported with the cost of more wait states.

8.7.1.3.2 Manual Mode

In Manual mode, the Flash memory is controlled by a register file. All instructions that are supported by Flash memory can be programmed using the register file. Moreover, the mode of interface (SPI, Dual SPI, Quad SPI, or Octa SPI) and the mode of operation (Auto or Manual mode) can be configured using this register file. The register file supports the following data sizes for reading and writing accesses: 8 bits, 16 bits, and 32 bits.

8.7.1.4 Endianness

The OQSPIC operates in little-endian mode. For 32-bit or 16-bit access (for read and write operations) to a serial Flash memory, the least significant byte comes first. For 32-bit access, the byte ordering is: data [7:0], data [15:8], data [23:16], data [31:24] while for 16-bit access the byte ordering is: data [7:0], data [15:8].

8.7.1.5 Erase Suspend/Resume

The OQSPI Flash can be used for Data Storage, combining the EEPROM functionality and Program storage on one single device. For this purpose, the Octa/Quad SPI ERASE/SUSPEND ERASE/RESUME commands are automatically executed as shown in Figure 53.

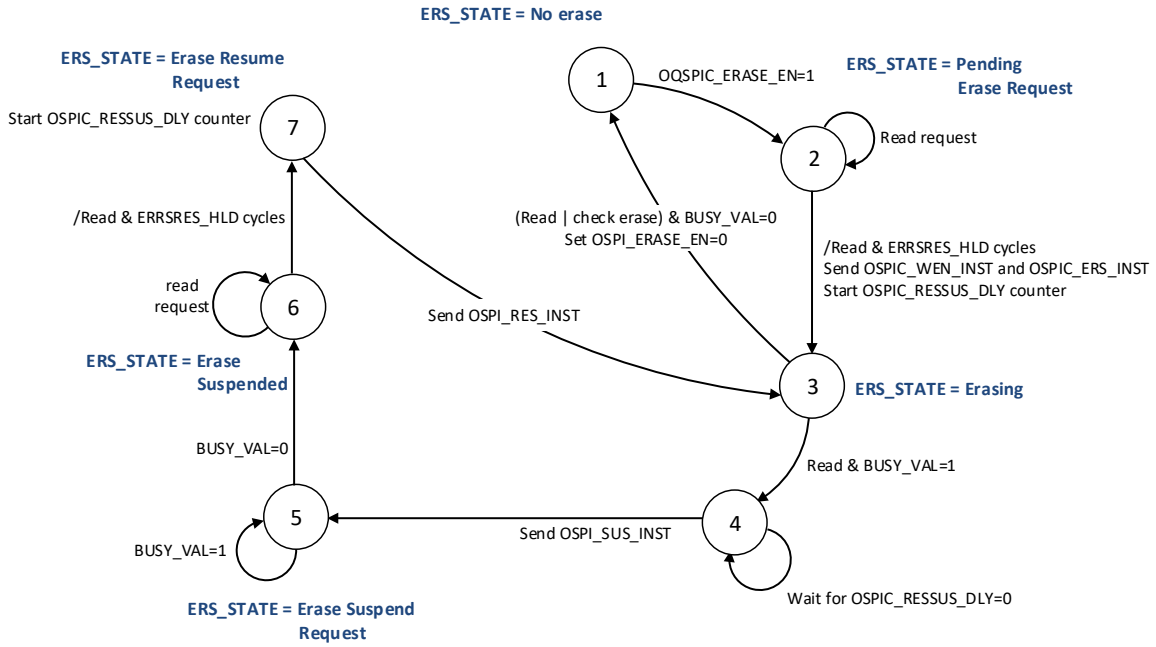


Figure 53. Erase suspend/resume in Auto mode

8.7.1.6 On-the-fly Decryption

The OQSPIC supports decryption of the data retrieved from the Flash device, only when in Auto mode. The Flash contents should be encrypted already, using the same algorithm.

The address range, which is decrypted automatically by the controller, is defined in the OQSPIC by using two configuration registers, one for the start address (OQSPIF_CTR_SADDR_REG register), and the other for the end address (OQSPIF_CTR_EADDR_REG register). The defined address range is 1024 bytes aligned. All addresses, which are outside of this range, are not automatically decrypted but fetched as is.

The on-the-fly decryption feature is based on the CTR mode of the AES encryption algorithm. The AES algorithm process blocks of 128 bits. That means, the input and output block of the AES is 128 bits or 16 bytes and as a result, data to be processed by the algorithm are fragmented into blocks of 16 bytes. The key size that is used for the AES algorithm is 256 bits.

For the AES-256-CTR algorithm, only the cipher part of the AES algorithm is required. The general idea of the encryption process is based on the encryption of a 128-bit counter block (CTR). The initial value of the CTR is labeled as CTR0. The first counter block (CTR0) is encrypted with the help of the AES cipher, and the encrypted result is XORed with the first 16 bytes of the plaintext data (P0) to be encrypted. The counter block is incremented by one (CTR1 = CTR0 + 1) and is encrypted again. The result is XORed with the next 16 bytes of the plaintext (P1) and so on until all plaintext data is encrypted.

The AES CTR decryption is the same process as encryption. By XORing again, the ciphertext with the same encrypted counter value, the plaintext is retrieved. The decryption process can be described by equations:

For j = 1 to m, do $CTR_j = CTR_{j-1} + 1$

For j = 0 to m, do $P_j' = AES\_CIPH_k(CTR_j) \oplus C_j$

Because access to the Flash memory is random, the structure of the CTR block is selected to simplify the process. The total size of the counter block is 128 bits or 16 bytes, namely: $CTRB_0$, $CTRB_1$, $CTRB_2$, $CTRB_3$, …, $CTRB_{14}$, $CTRB_{15}$.

The first 8 bytes ($CTRB_0$ - $CTRB_7$) of the counter block comprise the NONCE value and are programmed in the QSPI Controller in configuration registers (OQSPIF_CTR_NONCE_*_REG). This is typically a random value and is the same for all the CTR_i blocks.

The next four bytes of the counter block ($CTRB_8$-$CTRB_{11}$) are always zero.

The last four bytes of the counter block ($CTRB_{12}$-$CTRB_{15}$) are produced automatically by the hardware based on the 32-bit address offset OFFSET_ADDR [31:0] inside the encrypted range, where the data that should be decrypted are placed. If FLASH_ADDR[31:0] is the absolute address of a specific byte inside the encrypted

range, the offset address is OFFSET_ADDR = FLASH_ADDR – OQSPIF_CTR_SADDR_REG[OSPIC_CTR_SADDR]. The four least significant bits of the address offset are truncated and the four most significant bits of the CTRB12 are padded with zeros. Thus, the zero value in bytes $CTRB_{12}$-$CTRB_{15}$ of the CTR is used for the first 16 bytes of the address range that is encrypted, the value 1 is used for the second 16 bytes of the address range, and so forth.

The final form of the counter block is the following:

{64 bits NONCE, 32 bits 0x0, 4 bits 0x0, OFFSET_ADDR[31:4]}

The four least significant bits of the address offset OFFSET_ADDR[3:0] define which of the AES_CIPHk(CTR_j) byte should be used for the decryption of a specific byte of the encrypted block C_i.

In this way, the CTR block, which should be used for the decryption of a specific byte, can be calculated immediately by the address of the byte in the Flash and the start address of the encrypted range. This counter block supports up to 4 GB data that covers the maximum supported size for the Flash devices.

8.8 Quad SPI RAM/Flash Controller – PSRAM

8.8.1 Introduction

The Quad SPI Controller (QSPIC) provides a low pin count interface to serial QSPIC Flash and RAM devices. The QSPIC supports the standard SPI and a high-performance Dual/Quad SPI Interface. The QSPIC RAM feature provides a low-cost RAM extension for infrequently used data and can be used in combination with the data cache controller.

The QSPIC automatically generates all the control signals for the QSPI bus needed to access data from the serial quad memory. The controller has a vendor independent register file that provides a rich set of control fields for a wide range of Flash and RAMs.

The QSPIC provides transparent memory mapped RAM access.

Features

- SPI modes:
  - Single: Data transfer through two unidirectional pins
  - Dual: Data transfer through two bidirectional pins
  - Quad: Data transfer through four bidirectional pins.
- Auto mode: up to 64 MB memory mapped Read/Write Data access with 3-byte and 4-byte addressing modes
- Manual mode: Direct register access using the QSPIC register file
- Clock modes 0 and 3. Master mode only
- Vendor independent Instruction Sequencer
- In Auto mode, the Flash control signals are fully programmable
- Use of special read instruction for a specific (programmable) wrapping burst access
- Erase suspend/resume to Support for Code and Data storage
- Data Cache controller can be enabled or bypassed.

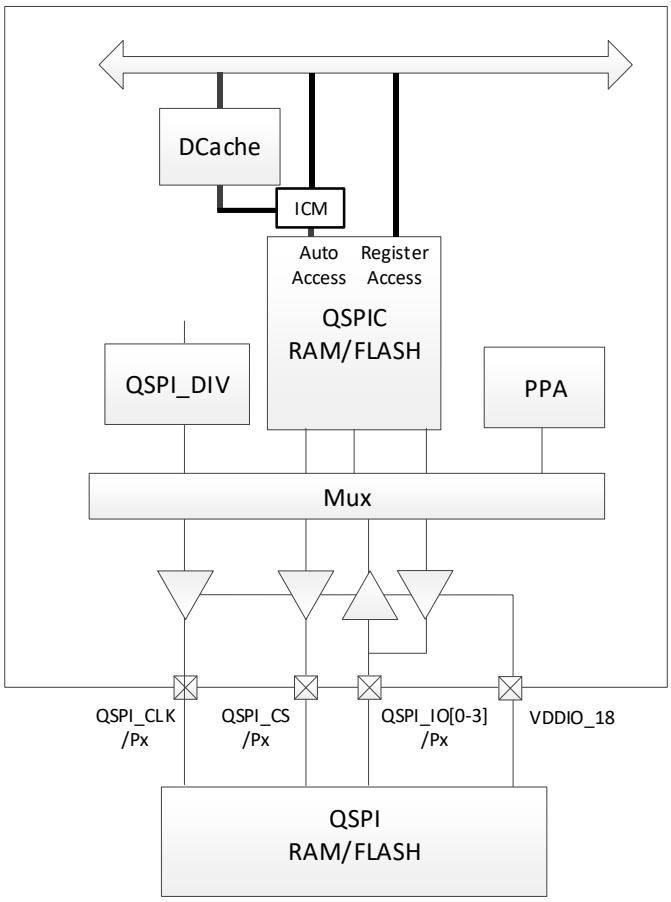


Figure 54. Quad SPI RAM/Flash controller

8.8.2 Interface

The QSPIC interface is assigned to specific pins because of the performance requirements of the pin. All pins can be assigned as either QSPIC function or as a peripheral pin through the PPA. Table 52 shows the pin configurations for the QSPIC interface.

Table 52: QSPI pin configuration

Pin name	Pin assignment	I/O	Description
QSPI_CLK	P0_08	O	Output serial clock
QSPI_CS	P0_09	O	Active low output Chip select
QSPI_D0	P0_10	I/O	MOSI (output) in single SPI mode D0 (bidirectional) in Quad SPI mode
QSPI_D1	P0_11	I/O	MISO (input) in single SPI mode D1 (bidirectional) in Quad SPI mode
QSPI_D2	P0_12	I/O	WPn Write Protect output in single SPI mode D2 (bidirectional) in Quad SPI mode
QSPI_D3	P0_13	I/O	HOLDn/Resetn output in Single SPI mode D3 (bidirectional) at Quad SPI mode

8.8.3 SPI Modes

The QSPIC supports the following SPI standards:

- Single: Data transfer through two unidirectional pins. The QSPIC supports communication with any single/dual or Quad SPI Flash memory. In contradiction to the standard SPI interface, the supported Single SPI interface does not support the bus modes 1 and 2, full-duplex communications, or any SPI Slave mode.

- Dual: Data transfer through two bidirectional pins.
- Quad: Data transfer through four bidirectional pins.

8.8.4 Access Modes

Access to a serial memory (Flash or RAM) connected to the QSPIC can be done in two modes:

- Auto mode
- Manual mode.

These modes are mutually exclusive. The serial memory can be controlled only in one of the two modes. The registers which control the mode of operation can be used at any time.

In Auto mode, 3-byte and 4-byte addressing modes are supported. With QSPIC_USE_32BA=0, up to 16 MB serial memory (3-byte addressing) can be accessed. If QSPIC_USE_32BA = 1, the 4-byte addressing is enabled for accessing up to 64 MB serial memory.

8.8.4.1 Auto Mode Flash Access

In Auto mode (QSPIC_AUTO_MD=1), the read access to a serial Flash memory is performed in a fully transparent way through the SPI bus. A read access to the memory space, where the external memory is mapped, is translated by the controller to the respective SPI bus command sequence, which is needed for the retrieving of the requested data from the serial Flash memory.

When the Auto mode is disabled (QSPIC_AUTO_MD =0), any access (reading or writing) to the mapped memory space is ignored by the controller.

Only read access is supported when the connected external device is a Flash memory (QSPIC_SRAM_EN = 0). Write access causes a hard fault at the CPU.

The read access can be single access or incremental burst or wrapping burst access. The wrapping burst is supported even when the controlled serial Flash does not support any special instruction for wrapping burst. Special read instruction can be used for specific (programmable) wrapping burst access. When a serial Flash supports a special instruction for wrapping burst access, this feature saves access time (less wait states). For maximizing the utilization of the bus and minimizing the number of wait states, it is recommended to be used burst accesses. However, non-sequential random access is supported with the cost of more wait states.

8.8.4.2 Auto Mode RAM Access

When it is connected to a serial RAM device, the QSPIC controller can provide read/write functionality.

The special configuration register must be programmed to enable the RAM functionality (QSPIC_SRAM_EN=1). If the external device is a Flash, the Auto mode must also be enabled (QSPIC_AUTO_MD=1). If the Auto mode is disabled (QSPIC_AUTO_MD=0), any access (reading or writing) in the memory space, where the external device has been mapped, is ignored by the QSPIC.

The read access in the memory space of the external serial RAM, is done in a fully transparent way through the QSPI bus. The capability of the controller to handle the various types of read accesses, is the same as the Flash device. Single access, incremental burst or wrapping burst are all supported.

Write access to the memory space where the external memory is mapped, does not cause a hard fault to the Cortex-M0. In the contrary, the write access is interpreted by the QSPIC in the respective QSPI bus protocol and the write data is stored in the external RAM device.

The controller is capable of handling write accesses of all kinds of burst: single access, incremental burst or wrapping burst access. The throughput that can be achieved varies depending on the burst length, the word width, the cost of the protocol of the external memory device, and the frequency of the QSPI clock.

Burst access provides the highest throughput. The non-sequential random accesses are supported at the cost of more wait states. The maximum throughput that can be achieved depends on the burst length.

8.8.4.3 Manual Mode

In manual mode, the external serial memory is controlled through a register file. All instructions that are supported by the serial memory can be programmed by using the register file. Moreover, the mode of interface (SPI, Dual SPI, Quad SPI) and the mode of operation (Auto or Manual mode), can be configured through this register file. The register file supports the following data sizes for reading and writing accesses: 8 bits, 16 bits, and 32 bits.

8.8.5 Endianness

The QSPI controller operates in little-endian mode. For 32-bit or 16-bit access (for read and write operations) to a serial memory, the least-significant byte comes first. For 32-bit access, the byte ordering is: data [7:0], data [15:8], data [23:16], data [31:24] and for 16-bit access the byte ordering is: data [7:0], data [15:8].

8.8.6 Erase Suspend/Resume

A QSPI Flash memory can be used for data storage, combining the EEPROM functionality with Program storage in one single device. For this purpose, the QSPI ERASE/SUSPEND ERASE/RESUME are automatically executed as shown in Figure 55.

To store data in QSPI Flash memory, the sector designated for storage must be erased first. The ERASE/SUSPEND ERASE/RESUME process is only meaningful if the external device is a serial Flash memory (QSPIC_SRAM_EN = 0).

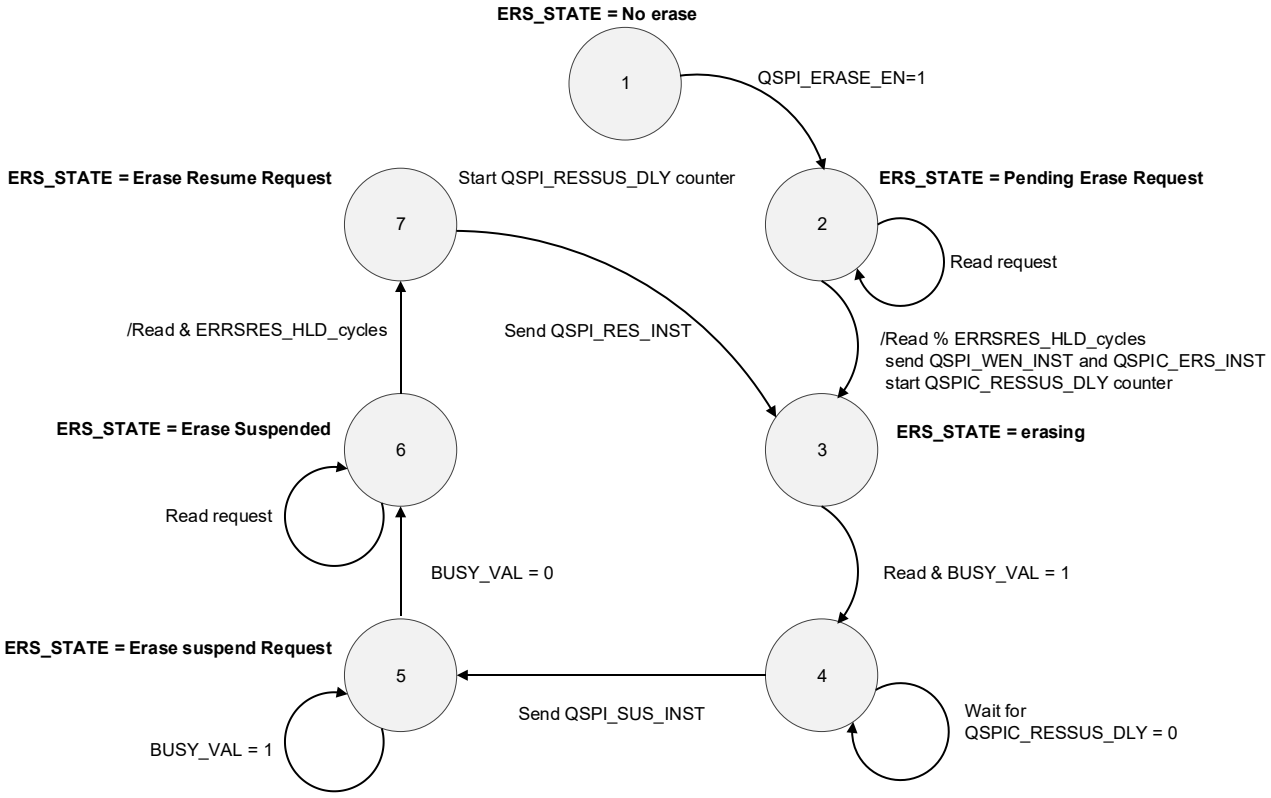


Figure 55. Erase suspend/resume in Auto mode

NOTE
QSPI_RESSTS_DLY is counted with the QSPI_CLK, so before changing the QSPI_CLK, make sure that QSPI_RESSTS_DLY is set large enough to meet the timing parameter requirements.

8.8.7 Low Power Considerations

To reduce the power dissipation in the QSPI Flash, the QSPI_CLK must always be the highest possible system clock to keep the burst access to the Flash as short as possible. The CPU must run as slow as possible for minimum power.

For the lowest power with slow CPU (for example, 2 MHz) and high QSPI_CLK (for example, 40 MHz) bit QSPIC_CTRLMODE_REG[QSPIC_FORCENSEQ_EN] must be set to 1. This enables split burst mode, reducing the power dissipation during active burst only, while disabling the Flash when the burst is done compared to high efficiency burst. These two modes are explained in the following two figures:

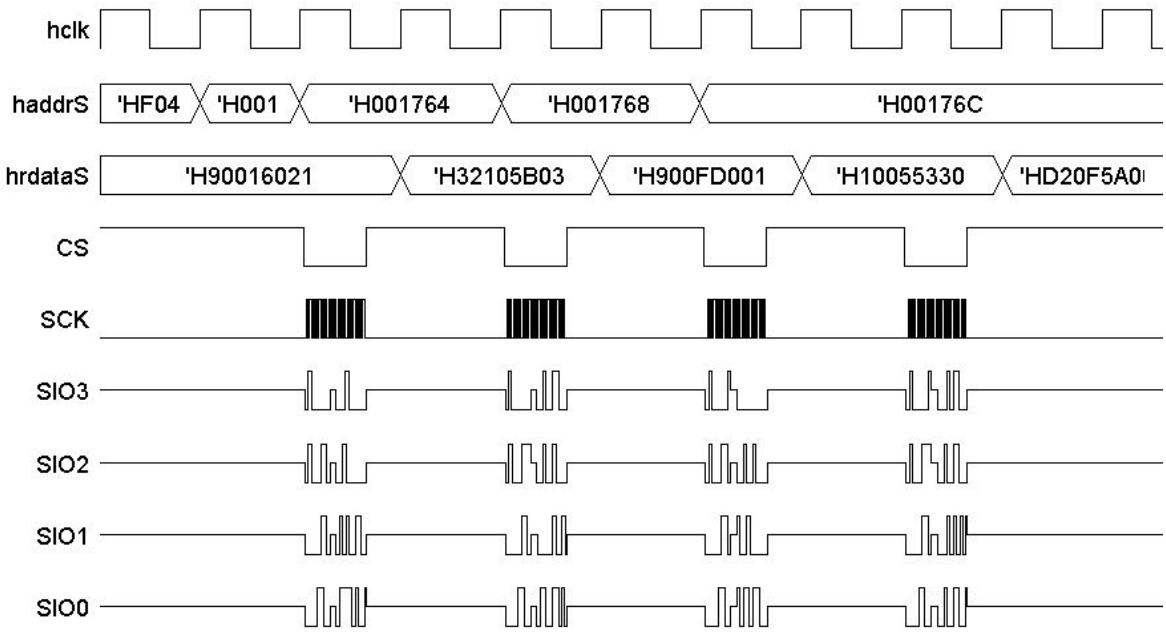


Figure 56. QSPI split burst timing for low power (QSPI_FORENSEQ_EN = 1)

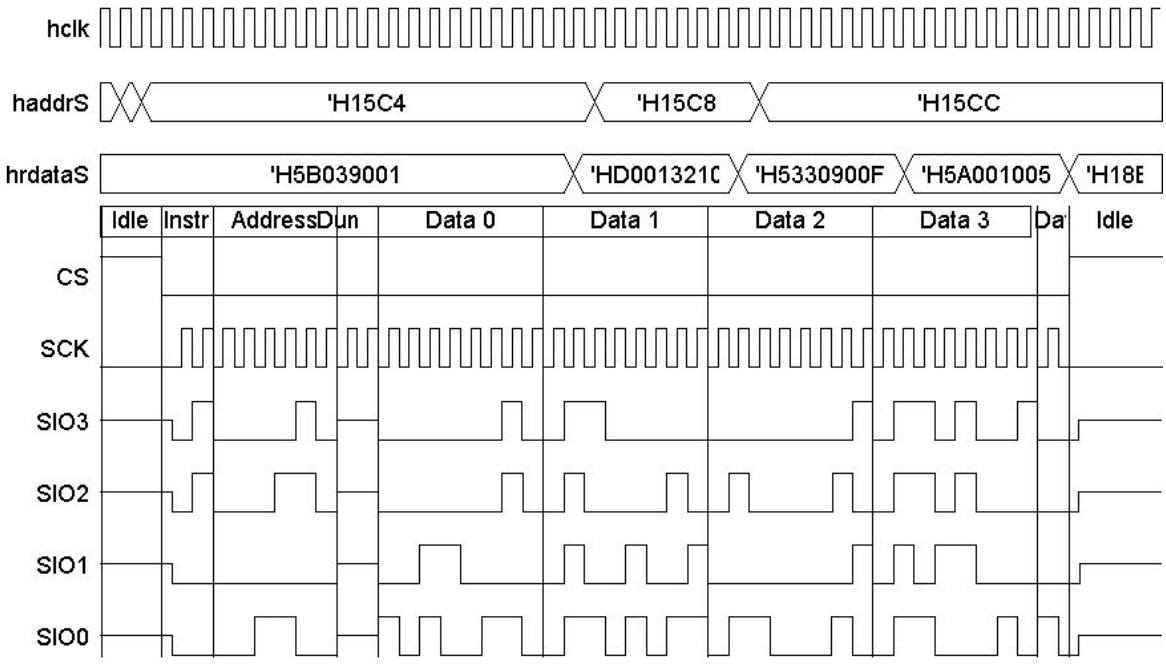


Figure 57. QSPI burst timing for high performance (QSPI_FORENSEQ_EN = 0)

If QSPI_FORCENSE_EN = 0, the QSPIC reads data in a burst address1/extra/dummy/data1, data2, dataN, keeping the QSPI_CS low during the complete burst. If set to 1, the burst is split into non-sequential accesses address1/extra/dummy/data1, address2/extra/dummy/data2, making the QSPI_CS high between the accesses.

8.9 General Purpose Timers/PWMs

8.9.1 Introduction

The Timers block contains eight 32-bit wide programmable timers which include PWM capabilities. All Timers reside in the system power domain.

Features

- Eight 32-bit general purpose timers
- Pulse Width Modulated signal (PWM), one per timer block
- PWM start synchronization
- Dual GPIO multi event capture with interrupt generation (4 GPIOs for Timer)
- One shot mode generated by a GPIO toggle or a register write
- 5-bit clock pre-scaler with selectable clock source XTAL32K or XTAL40M
- Up/down counting capability with free running mode
- Active while system is in Sleep modes 4 and 5
- Dedicated interrupt line per timer, common capture event interrupt
- GPIO edges (positive or negative) counting function
- Timers 1 and 5 support extra functions:
  - Single GPIO single event capture (only first edge)
  - Automatic switching from one-shot to counter mode capability.
  - PWM sync with other timers in the timer group:
    - Timer 1 is grouped with timers 2, 3, and 4
    - Timer 5 is grouped with timers 6, 7, and 8.

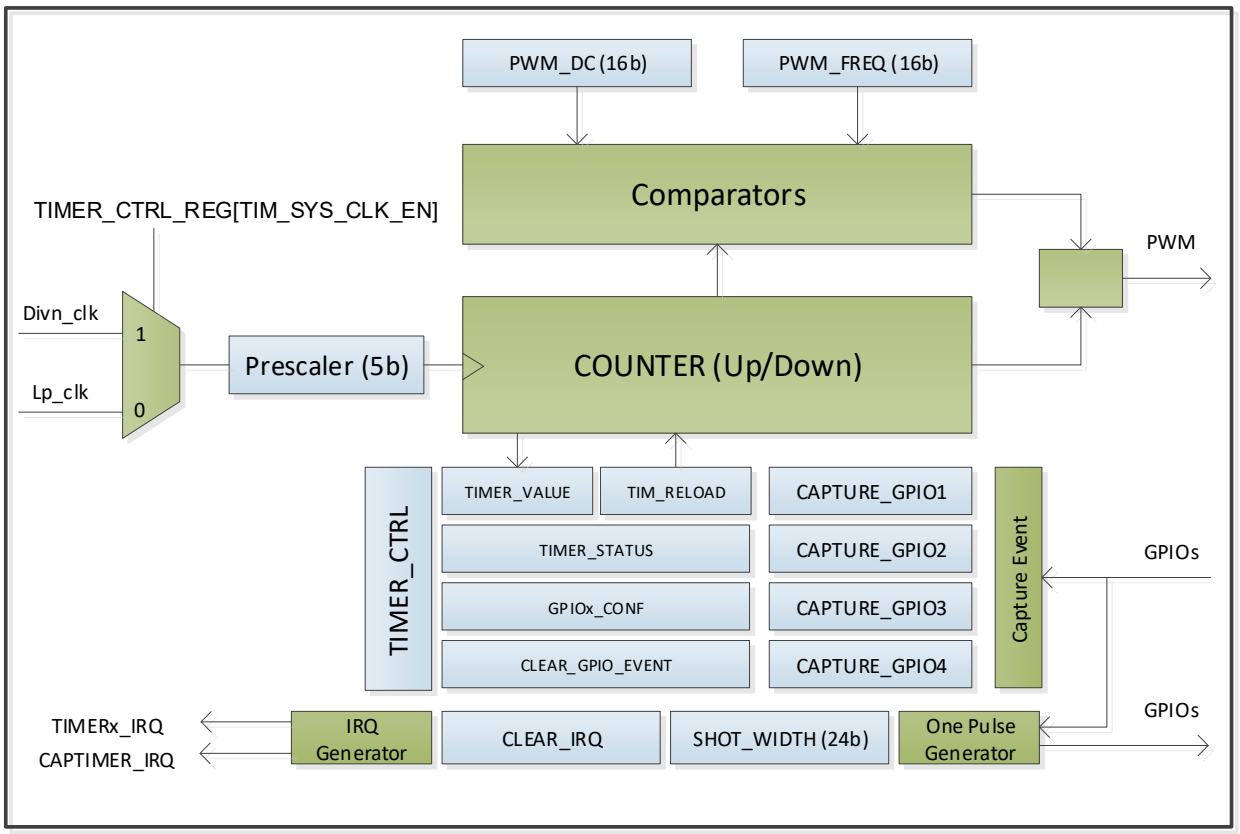


Figure 58. General purpose timers block diagram

There are eight 32-bit timers (Timers 1 - 8) with timer 1 and timer 5 supporting two extra capture channels. The general timer block diagram is shown in Figure 58. All timers reside in the system power domain (PD_SYS) and support the features listed in Table 53.

Table 53: Timer features overview

Feature	Timer (1)	Timer (5)	Timer (2/3/4/6/7/8)
Counter width	32-bit	32-bit	32-bit
Free-Running Counter, PWM generation, and Edge detection counter	✓	✓	✓
PWM start synchronization (Note 1)	✓	✓	✓
Event capturing channels	4	4	2
Event capture IRQ	×	✓	×
Timer IRQ	✓	✓	✓
One-Shot	✓	✓	✓
One-Shot with auto switch	✓	✓	×
First event capture	✓	✓	×

Note 1 PWM sync is grouped Timer 1/2/3/4 and Timer 5/6/7/8.

Each timer can be clocked with either the 40 MHz clock source or the 32 kHz clock for low power operation. Also, for each timer, a five-bit prescaler can be used to further reduce the clock resulting in a minimum timer frequency of 1 kHz and a maximum of 40 MHz.

8.9.2 Timer Modes of Operation

8.9.2.1 Free-Running Counter

Each timer block has a configurable free running up/down counter that triggers an interrupt when passing from the pre-configured value and immediately returns to 0 or pre-load depending on whether it is counting up or down. The timers can be instructed to continuously count upwards and wrap around when reaching its boundaries.

8.9.2.2 PWM Generation

Each timer has a Pulse Width Modulation (PWM) output (TIMx_PWM) which can be mapped to any available GPIO pin without affecting the actual counter while running.

The PWM frequency is defined by further dividing the prescaled clock by a maximum of 2^{16}+1. The duty cycle duration of the generated PWM signal can be configured in TIM_PWM_FREQ steps. If the minimum PWM frequency is selected, only 50% duty cycle is possible.

PWM synchronization is supported for all timers. It is assumed that all the timers are running on the same clock source while PWM frequency and duty cycle might be different.

8.9.2.3 Event Capturing

The GPIO multi event capture provides latching snapshots of the free-running clock into two registers. The difference of these two registers indicates the duration between two trigger events.

NOTE
Timer 1 and Timer 5 have four capture pairs.

A single event capture latches only a single snapshot, in the first capture register, with the first trigger (after programming) of the capture event. Subsequent edges of the triggering GPIO do not change the value of the first capture register and are ignored by the capturing circuitry.

8.9.2.4 One Shot

Upon an input trigger from a GPIO toggle or a register write, a separate GPIO serves as an output, delivering a pulse of configurable width. This feature implements a PWM reply in hardware. There are four different selections for the input trigger:

- External GPIO (TIMx_1SHOT assigned through PPA)
- Software trigger (write to the timer trigger register)
- Either of the two triggers
- None of the two triggers.

Timer 1 and Timer 5 support automatic switching from one-shot to counter mode without the CPU involvement. If no start value is programmed, an interrupt is immediately issued to the CPU.

8.9.2.5 GPIO Pulse Counter

Every timer block supports counting pulses from a programmable GPIO. This operation is available when the system is in Sleep 4 and Sleep 5 modes. The frequency range of the edge train may never exceed 80 MHz for proper edge detection and counter update.

If the threshold is reached, an interrupt is generated, and the counter is automatically reset to zero.

8.9.3 Pin Configuration

Each timer has two I/Os which can be assigned to any available GPIO pin through the PPA function. The pin definitions for the Timer interfaces are shown in Table 26.

8.10 GPIOs and Programmable Pin Assignment

The functions assigned to the GPIO pins are fully configurable and are controlled by the Programmable Pin Assignment (PPA).

- **Features**
- Fully Programmable Pin Assignment
- Selectable 25 kΩ pull-up/pull-down resistors per pin up to selected voltage rail
- Programmable open-drain functionality
- Selectable drive strength: 2 mA, 4 mA, 8 mA, 14 mA
- Fixed assignment for ADCx, QSPI, OQSPI and SWD pins
- Pin state is maintained when the system enters low-power sleep 4 and 5 modes.

The PPA provides a multiplexing function for the I/O pins of the on-chip peripherals. Any of the peripheral input or output signals can be freely mapped to any available GPIO port.

The list of peripheral I/Os that can be mapped through the PPA are shown in Table 54.

Table 54: Pin function list

Function ID	Function name	Pin type (Note 1)
0	GPIO	(I/O)
1	UART_RX	(IA) (Note 2)
2	UART_TX	(OA) (Note 3)
3	UART_CTSN	(IA)
4	UART_RTSN	(OA)
5	UART_TXDOE	(OA)
6	UART1_RX	(IA)
7	UART1_TX	(OA)
8	UART1_CTSN	(IA)
9	UART1_RTSN	(OA)
10	UART1_TXDOE	(OA)
11	UART2_RX	(IA)
12	UART2_TX	(OA)
13	UART2_CTSN	(IA)

Function ID	Function name	Pin type (Note 1)
14	UART2_RTSN	(OA)
15	UART2_TXDOE	(OA)
16	SPI_DI	(IA)
17	SPI_DO	(OA)
18	SPI_CLK	(I/O)
19	SPI_CSN0	(I/O)
20	SPI_CSN1	(OA)
21	SPI2_DI	(IA)
22	SPI2_DO	(OA)
23	SPI2_CLK	(I/O)
24	SPI2_CSN0	(I/O)
25	SPI2_CSN1	(OA)
26	I2C_SCL	(I/O)
27	I2C_SDA	(IO-OD)
28	I2C2_SCL	(IO-OD)
29	I2C2_SDA	(I/O-OD)
30	Analog (Note 4)	(ADC)
31	PCM_DI	(I)
32	PCM_DO	
33	PCM_FSC (Note 5)	(I/O)
34	PCM_CLK (Note 6)	(I/O)
35	DMICA_DI	(I)
36	DMIC_CLK	(I/O)
37	MCLK	
38	TIM_PWM	(OA)
39	TIM2_PWM	(OA)
40	TIM3_PWM	(OA)
41	TIM4_PWM	(OA)
42	TIM5_PWM	(OA)
43	TIM6_PWM	(OA)
44	TIM7_PWM	(OA)
45	TIM8_PWM	(OA)
46	TIM_1SHOT	(OA)
47	TIM2_1SHOT	(OA)
48	TIM3_1SHOT	(OA)
49	TIM4_1SHOT	(OA)
50	TIM5_1SHOT	(OA)
51	TIM6_1SHOT	(OA)
52	TIM7_1SHOT	(OA)
53	TIM8_1SHOT	(OA)
54	CLOCK	Note 7

Function ID	Function name	Pin type (Note 1)
55	FEM_BS	(O)
56	FEM_CS	(O)
57	FEM_CTRL0	(O)
58	FEM_CTRL1	(O)
59	FEM_CTRL2	(O)
60	BT_COEX_CBT	(O)
61	BT_WLAN_ACT	(O)
62	BT_ACT	(I)
63	BT_PRI	(I)
64	RF_SW1	(O)
65	RF_SW2	(O)
66	EXT_INTR	(O)
99	SWCLK	(I)
100	SWDIO	(I/O)
101	WPROTECT	(I)
102	CDETECT	(I)
103	ZB_WLAN_ACT	(O)
104	ZB_ACT	(I)
105	ZB_PRI	(I)
106	BTCOEX_ASC0	(O)
107	BTCOEX_ASC1	(O)
108	BTCOEX_ASC2	(O)

Note 1 (I): Input, (I/O): Input/Output), if nothing mentioned: Output (default).

Note 2 IA: Pad direction is automatically set to input when the function is selected.

Note 3 OA: Pad direction is automatically set to output when the function is selected.

Note 4 For XTAL32K monitoring. See pinout.

Note 5 Also defined as PCM_WCLK.

Note 6 Also defined as PCM_BCLK.

Note 7 See GPIO_CLK_SEL_REG.

Note 8 In other cases, the pad direction can be set through the pin mode register.

Note 9 In output mode and analog mode, the pull-up/down resistors are automatically disabled.

When a pin is configured to function as a GPIO, it has the following configurable features:

- Direction (input/output)
- Push pull/Open drain
- Pull-up/Pull-down
- Drive strength (2 mA, 4 mA, 8 mA, 14 mA)
- Slew rate (Fast/Slow)
- Input selection (CMOS/Schmitt Trigger).

After a power on reset (POR), the default state of the pins is shown in Table 55.

Table 55: Pin configuration

Pin	Support wake-up	Retention group	Power domain	Alternate function 0	Alternate function 1	Alternate function 2	POR default
RST_N			VBAT	RST_N			RST_N
P0_00	Yes	VBAT	VBAT	RTC_WAKE_UP			GPIO
P0_01		VBAT	VBAT	sen_out			sen_out
P0_02		VBAT	VBAT	xtal32k_m			xtal32k_m
P0_03		VBAT	VBAT	xtal32k_p			xtal32k_p
P0_04	ana wake	DIO1_1	VDDIO_DIO1	ADC0		eMMC_DIO4	GPIO
P0_05	ana wake	DIO1_1	VDDIO_DIO1	ADC1		eMMC_DIO5	GPIO
P0_06	ana wake	DIO1_1	VDDIO_DIO1	ADC2		eMMC_DIO6	GPIO
P0_07	ana wake	DIO1_1	VDDIO_DIO1	ADC3	MCLK	eMMC_DIO7	GPIO
P0_08	Yes	DIO1_2	VDDIO_DIO1	QSPIR_CLK	SDIO0_CLK	eMMC_CLK	GPIO
P0_09	Yes	DIO1_2	VDDIO_DIO1	QSPIR_CS	SDIO0_CMD	eMMC_CMD	GPIO
P0_10	Yes	DIO1_2	VDDIO_DIO1	QSPIR_D0	SDIO0_D0	eMMC_DIO0	GPIO
P0_11	Yes	DIO1_2	VDDIO_DIO1	QSPIR_D1	SDIO0_D1	eMMC_DIO1	GPIO
P0_12	Yes	DIO1_2	VDDIO_DIO1	QSPIR_D2	SDIO0_D2	eMMC_DIO2	GPIO
P0_13	Yes	DIO1_2	VDDIO_DIO1	QSPIR_D3	SDIO0_D3	eMMC_DIO3	GPIO
P1_00		FDIO	VDDIO_FDIO	OQSPI_D4	eMMC_DIO4		GPIO
P1_01		FDIO	VDDIO_FDIO	OQSPI_D5	eMMC_DIO5		GPIO
P1_02		FDIO	VDDIO_FDIO	OQSPI_D6	eMMC_DIO6		GPIO
P1_03		FDIO	VDDIO_FDIO	OQSPI_D7	eMMC_DIO7		GPIO
P1_04		FDIO	VDDIO_FDIO	OQSPI_D0			GPIO
P1_05		FDIO	VDDIO_FDIO	OQSPI_D1			GPIO
P1_06		FDIO	VDDIO_FDIO	OQSPI_D2			GPIO
P1_07		FDIO	VDDIO_FDIO	OQSPI_D3			GPIO
P1_08		FDIO	VDDIO_FDIO	OQSPI_CLK			GPIO
P1_09		FDIO	VDDIO_FDIO	OQSPI_CS			GPIO
P1_10	Yes	DIO2	VDDIO_DIO2	eMMC_CMD	SDIO1_CMD		GPIO
P1_11	Yes	DIO2	VDDIO_DIO2	eMMC_CLK	SDIO1_CLK		GPIO
P1_12	Yes	DIO2	VDDIO_DIO2	eMMC_DIO0	SDIO1_D0		GPIO
P1_13	Yes	DIO2	VDDIO_DIO2	eMMC_DIO1	SDIO1_D1		GPIO
P1_14		DIO2	VDDIO_DIO2	eMMC_DIO2	SDIO1_D2		GPIO
P1_15		DIO2	VDDIO_DIO2	eMMC_DIO3	SDIO1_D3		GPIO
P1_16		DIO2		SWCLK			SWCLK
P1_17		DIO2		SWDIO			SWDIO

8.11 ADC/Analog or ADC (Aux 12-bit)

8.11.1 Introduction

The RA6W2 includes high precision, ultra-low-power, and wide dynamic range SAR ADC with a 12-bit resolution. It has a 4-channel single-end ADC. Analog input is measured by four pins from P0_04 to P0_07, and pin selection is changed through the register setting.

Figure 59 shows the control block diagram.

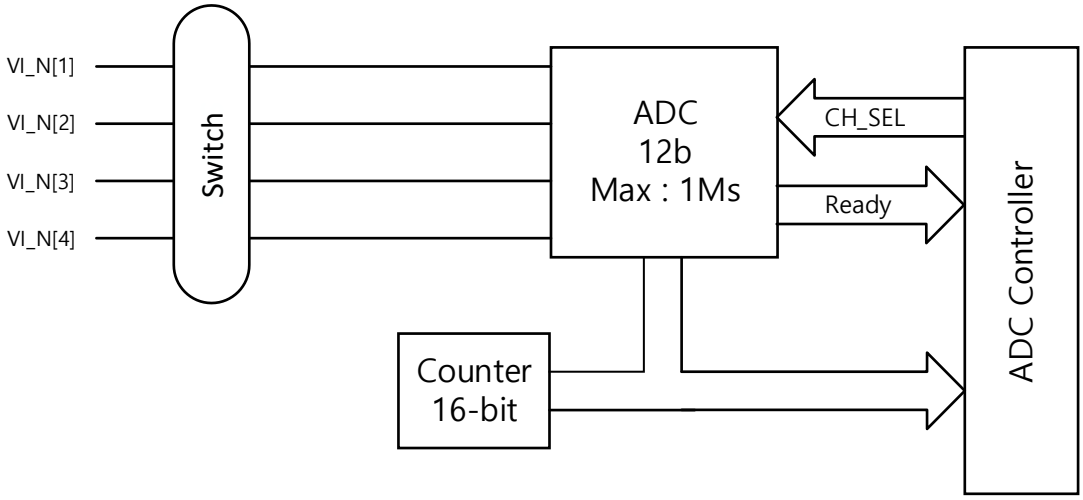


Figure 59. ADC control block diagram

8.11.2 Timing Diagram

The input is digitized at a maximum of 1.0 Msps throughput rate. And the maximum input clock rate is 15 MHz. Figure 60 shows the conversion timing, and Table 56 describes the DC specifications.

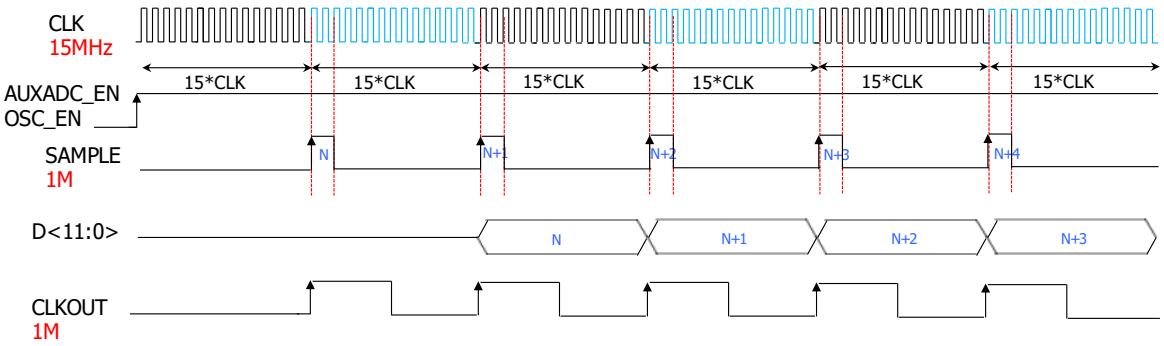


Figure 60. 12-bit ADC timing diagram

Table 56: ADC DC specification

Description	Min	Typ	Max	Unit
Resolution	4	12	12	Bits
Max clock input			15	MHz
Conversion frequency			1	MHz
Accuracy:				
SNR SNDR		67.2 61.7		dB dB
Analog input voltage	0		1.4	V
Reference voltage		0.7		V

8.11.3 DMA Transfer

There are four ADC channel settings available. When the input data of each channel reaches the FIFO level, it is possible to read the data through the DMA path.

8.11.4 Sensor Wake-up

The RA6W2 has an external sensor wake-up function that uses the analog input signal through an Aux ADC. Even in Sleep mode, when a change of an external analog signal is detected, a wake-up event occurs, and normal operation is resumed. This function can be used in up to four channels. Also, when multiple external sensors are used, analog signals are detected while the channel is automatically changed. For example, if all four channels are set as input sources which have their threshold register respectively, the channels are measured sequentially from 0 to 3.

If one of the four ADC channels exceeds the allowed range of values set by the threshold register, the RA6W2 awakes from the Sleep mode. The value setting of the input change can be either over threshold or under threshold.

8.11.5 ADC Pin Configuration

Table 57 shows the pin definition of the ADC.

Table 57: ADC pin configuration

Pin name	Default pin assignment	I/O	Description
ADC0	P0_04	ADC	ADC0 Analog input
ADC1	P0_05	ADC	ADC1 Analog input
ADC2	P0_06	ADC	ADC2 Analog input
ADC3	P0_07	ADC	ADC3 Analog input

9. Bluetooth LE Subsystem

9.1 Overview

The RA6W2 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® LE protocol implementation, for providing processing power for calculations or data fetches required by applications, and for housekeeping, including controlling the power scheme of the system.

Bluetooth LE Core. This is the baseband hardware accelerator for the Bluetooth® LE protocol.

ROM. This 144 kB ROM contains the Bluetooth® LE protocol stack and the boot code sequence.

OTP. This 32 kB OTP memory array is used to store application code and Bluetooth® LE profiles. It also contains the system configuration and calibration data.

System SRAM (SysRAM). This 48 kB SysRAM is primarily used to mirror the program code from the OTP when the system wakes/powers up. It also serves as a Data RAM for intermediate variables and various data that the protocol requires. It can be used as extra memory space for the Bluetooth® LE TX and RX data structures (Exchange RAM). The SysRAM cells can not only be retained during sleep modes but also be completely switched off during active mode if not needed.

UART and UART2. The serial interface of UART implements hardware flow control while UART2 does not. Both UARTs feature FIFOs with depths of 16 bytes each.

SPI. This is the serial peripheral interface (SPI) with master/slave capability, and it has separate FIFOs for RX and TX of two 16-bit words each.

I2C. This Master/Slave I2C interface is used for sensors and/or host Micro-Controllers Units (MCUs) communication. It comprises a 32-place 9-bit deep FIFO.

General Purpose (GP) ADC. This 10-bit analog-to-digital converter (ADC) has four external input channels (GPIOs) and internal channels for reading die temperature, battery voltage, and other internal analog nodes.

Radio Transceiver. This block implements the RF part of the Bluetooth LE protocol.

Clock Generator. This block is responsible for clocking the system. It contains two XTAL oscillators, one running at 32 MHz (XTAL32M) and used for the active mode of the system and the other running at 32.768 kHz (XTAL32K) and used for the sleep modes of the system. There are also three RC oscillators available: a 32 MHz oscillator (RC32M) with low precision (> 500 ppm), a 32 kHz oscillator (RC32K) with low precision (> 500 ppm), and a ~15 kHz oscillator (RCX) with high precision (< 500 ppm). The RCX oscillator can be used as a sleep clock to replace the XTAL32K oscillator to further improve the power dissipation of the system while reducing the bill of materials. The RC32M oscillator is used to provide a clock to mirror the OTP code into the SysRAM while the XTAL32M oscillator is settling directly after power/wake up. This clock is also used to run the Booter at power-up. An external digital clock can be used as a sleep clock to replace the XTAL32K or the RCX oscillator.

Timers. This block contains three timers:

- A 16-bit general purpose timer (Timer0) with two pulse width modulation (PWM) signals (PWM1 is inverted to PWM0).
- An 11-bit timer (Timer1) with two capture channels.
- A 14-bit timer (Timer2) that controls six PWM signals that all have the same frequency, but each has a configurable duty cycle.

Real Time Clock (RTC). This hardware controller supports the complete time of day clock: 12/24 hours, minutes, seconds, milliseconds, and hundredths of a millisecond. It includes a configurable alarm function and can be programmed to generate an interrupt on any event, like a rollover of the month, day, hour, minute, second, or hundredths of a millisecond.

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

Quadrature Decoder. This block decodes the pulse trains from a rotary encoder to provide the step and the direction of a movement of an external device. Three axes (X, Y, and Z) are supported. The block also supports an edge counting mode which enables counting positive or negative edges on the selected GPIOs.

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. This block implements the AMBA Lite version of the AHB and APB specifications.

Power Management. This sophisticated power management circuit is equipped with a Buck DC-DC converter and several low-dropout regulators (LDOs) that can be turned on/off via software.

9.2 Power Management Unit

9.2.1 Introduction

The RA6W2 has an integrated power management unit (PMU) which comprises a VDD_Clamp, a POR circuitry, a DC-DC converter, and various LDOs. The system diagram of the integrated PMU is shown in Figure 61.

Features

- Buck, and DC-DC bypass configurations
- Single inductance DC-DC converter configured for Buck configuration
- Programmable DC-DC converter outputs
- Active and sleep mode LDOs
- Low BOM and use of small external components.

9.2.2 Architecture

The PMU integrates two externally decoupled power rails: V_{BAT_HIGH} and V_{BAT_LOW}, and one internal V_{DD} power rail. There are three main power configurations: Buck and Bypass. The integrated PMU configures itself automatically to the appropriate mode depending on how the battery is initially connected to the application.

- V_{BAT_HIGH}: voltage range of 1.8 V - 3.6 V. This power rail is used for the blocks which require a higher supply voltage. The OTP and the GPIOs are connected to this power rail. V_{BAT_HIGH} is protected by the POR circuit POR_HIGH, which generates a POR when the voltage drops below the threshold voltage.
- V_{BAT_LOW}: the main system supply and most internal blocks are powered from this rail. Its functional range is from 1.1 V to 3.3 V. When used in Boost configuration, its default voltage range is from 1.1 V to 1.65 V, and within this range, the DC-DC converter can provide a supply at V_{BAT_HIGH} in the range of 1.8 V to 3.0 V. V_{BAT_LOW} is protected with the POR circuit POR_LOW which generates a hardware reset when the voltage drops below the threshold voltage.
- The internal V_{DD} power rail supplies the digital power domains (see Section 9.2.3 for the details).

The VDD_Clamp and RC32k blocks are supplied by the highest V_{BAT_HIGH} and V_{BAT_LOW}.

In the Buck configuration (Figure 61), the battery is connected to V_{BAT_HIGH}. The voltage on V_{BAT_LOW} is generated from V_{BAT_HIGH}. The different power modes of the system are explained in Section 9.2.4.

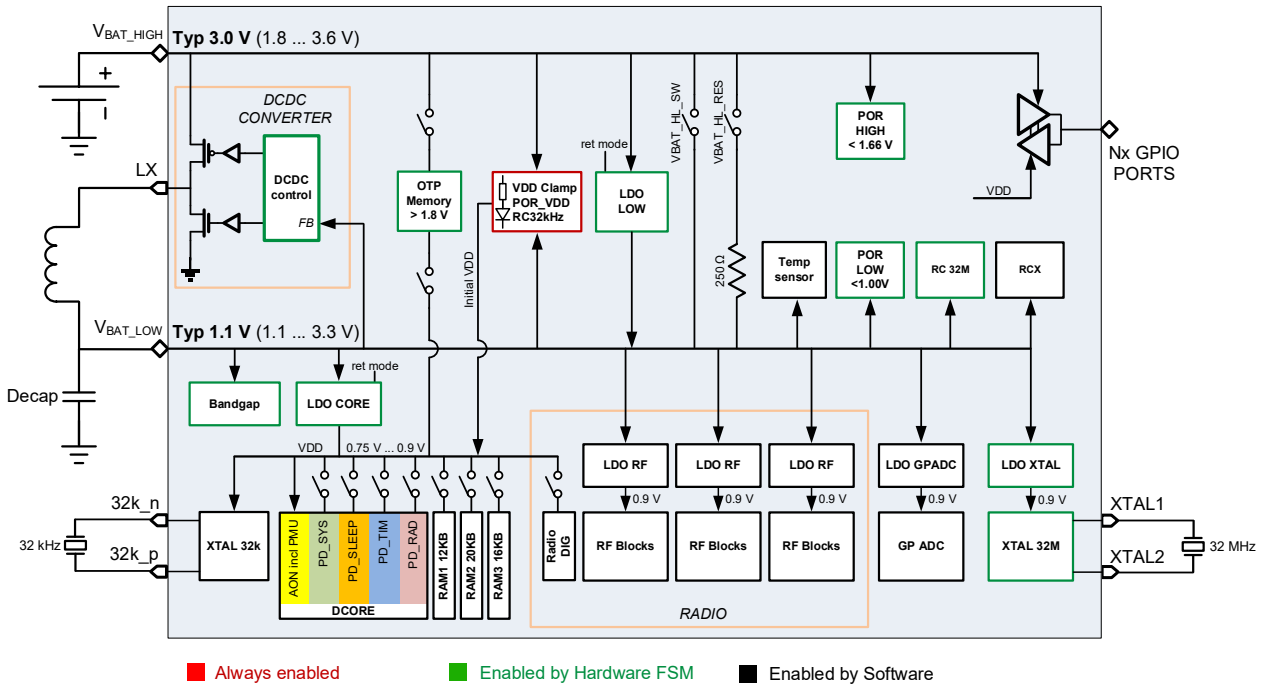


Figure 61. Power management unit: buck configuration

In bypass configuration (Figure 62), the DC-DC is bypassed, and the battery is connected to both V_{BAT_LOW} and V_{BAT_HIGH}. In this mode, an external inductor is omitted resulting in lower BOM. GPIOs supply follows the battery voltage in this configuration. The minimum cold boot voltage is 1.8 V. After cold boot and providing that no OTP or GPIO (at a specific voltage level) is needed, POR HIGH can be disabled and then V_{BAT_BYPASS} can go down to 1.1 V.

VBAT_LOW is limited to a max of 3.3 V, the battery/supply voltage shall not exceed that level in the bypass configuration. If an application requires supply levels up to 3.6 V and the choice is made to omit the DC-DC converter (reduced BOM at the cost of higher current consumption), it is recommended to use the "Buck Configuration" and enable LDO_LOW to power the VBAT_LOW rail (instead of via DC-DC converter).

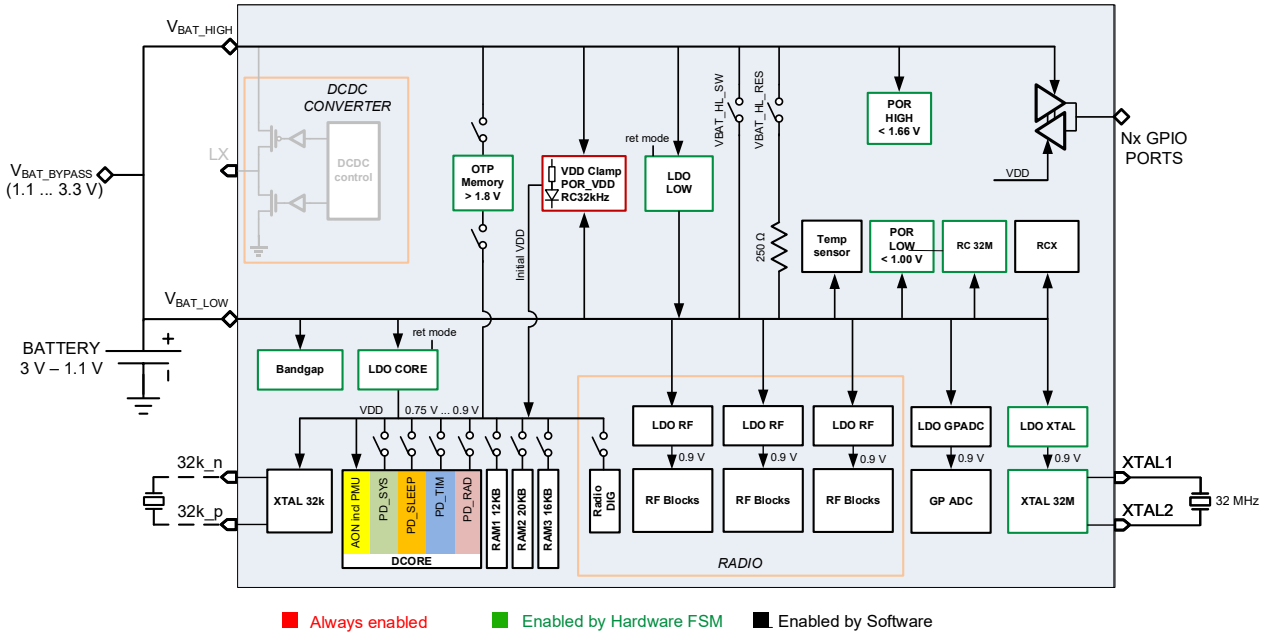


Figure 62. Power management unit: bypass configuration

9.2.3 Digital Power Domains

The RA6W2 supports several digital power domains that can be turned on and off by software (Figure 63). Some of the blocks contain registers that can retain their values even if the digital power domain where they reside is powered off. RAM cells can retain their contents independently from the digital power domains state.

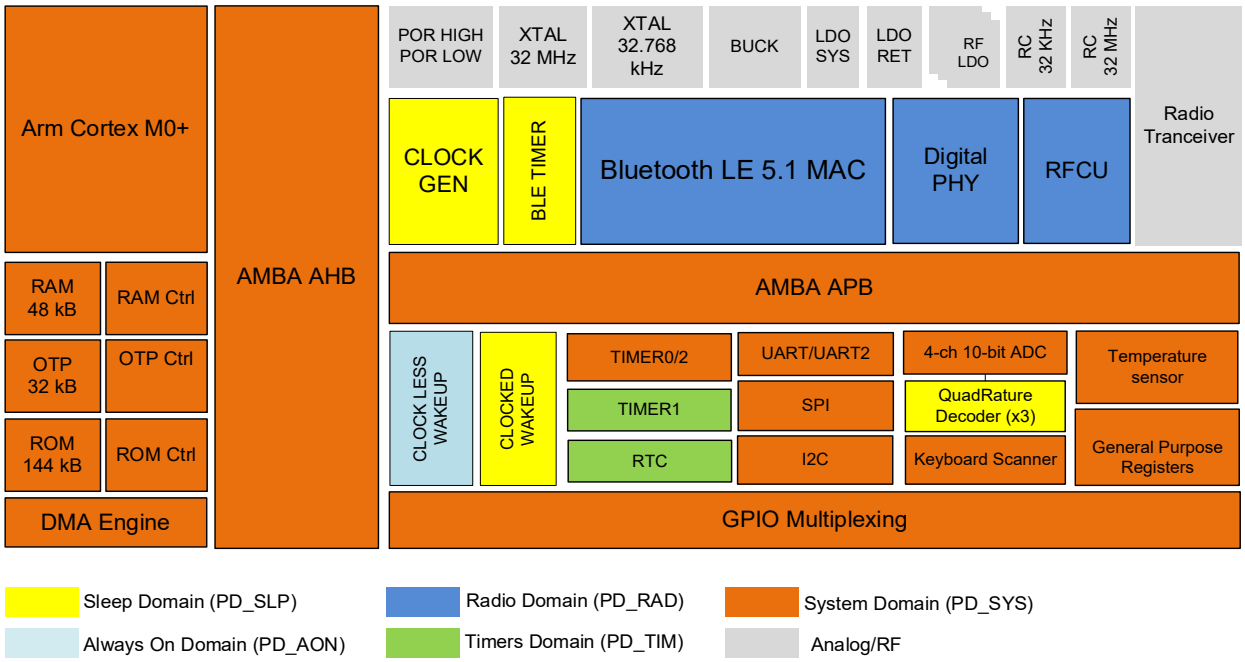


Figure 63. Digital power domains

The list of blocks residing in each one of the digital power domains is presented in Table 58.

Table 58: Power domains description

Domain name	Description
PD_AON	Always powered domain. It contains a Clock-less Wake-Up controller and the pad-ring.
PD_SLP	Sleep power domain. It comprises the Arm/WIC, the Bluetooth LE Timer, the PMU/Clock Generation, the Clocked Wake-Up Controller, the Quadrature Decoder, and various registers required for the Wake-Up sequence.
PD_SYS	System Power Domain. It comprises the AHB bus, the OTP cell and controllers, the ROM, the System RAM, the Watchdog, the Software Timer, and the GPIO port multiplexing.
PD_TIM	Timer Power Domain. It comprises the RTC and the Timer1. These two blocks can be active during sleep modes.
PD_RAD	Radio Power Domain. It comprises the Bluetooth LE Core and the digital PHY of the Radio.

9.2.4 Power Modes

There are five different power modes in the RA6W2:

- **Active mode**: System is active and operates at full speed.
- **Sleep mode**: No power gating has been programmed. The Cortex CPU is idle, waiting for an interrupt. PD_SYS is on. PD_TIM and PD_RAD depend on the programmed enabled value.
- **Extended Sleep mode**: PD_AON, PD_SLP, and conditionally PD_TIM are active. RAM is expected to be retained for:
  - Keeping a Bluetooth LE connection alive (stack variables or Bluetooth LE data).
  - Potentially keep the application code and it can be omitted if the OTP is instructed to automatically get mirrored into RAM upon every wake-up.
- **Deep Sleep mode**: Shipping clocked mode with all domains disabled. RAM may or may not be retained. RTC operation is programmable.

- **Hibernation mode**: Shipping clock-less mode with all domains disabled. RAM may or may not be retained. No clock is running.

A summary of the power modes, the digital power domains, as well as the clocks and wake-up capabilities are explained in Table 59.

Table 59: Power modes, Digital Power domains, Clocks, and Wake-Up Triggers

Power mode	Digital Power domains	LDOs, DC-DC Converter, and VDD level	Clock availability	RAM	Wake up from
Active or Sleep (WFI)	PD_AON = ON PD_SLP = ON PD_SYS = ON PD_TIM = OPTIONAL PD_RAD = OPT	VDD = 0.9 V DC-DC = ON (Buck) LDO_LOW = OFF LDO_CORE = ON, Active (0.9 V) VDD_Clamp = OFF LDO_RADIO = Programmable	All	SysRAM1 = ON (Application) SysRAM2 = optionally retained SysRAM3 = ON (Stack Data)	
Extended Sleep (with or without OTP)	PD_AON = ON PD_SLP = ON PD_SYS = OFF PD_TIM = OPTIONAL PD_RAD = OFF	VDD = 0.75 V DC-DC = OFF LDO_LOW = ON, in Buck mode. LDO_CORE = ON, in retain mode (0.75 V) VDD_Clamp = OFF LDO_RADIO = OFF	RCX or XTAL32K	SysRAMx = optionally retained (typically only SysRAM1 is retained)	▪ any GPIOs ▪ RTC alarm ▪ Timer1 ▪ Bluetooth LE sleep timer
Deep Sleep	PD_AON = ON PD_SLP = ON PD_SYS = OFF PD_TIM = OPTIONAL PD_RAD = OFF	VDD = 0.75 V DC-DC = OFF LDO_LOW = ON, in Buck mode LDO_CORE = ON, in retain mode (0.75 V) VDD_Clamp = OFF LDO_RADIO = OFF	RCX or XTAL32K	SysRAMx = optionally retained (Typically OFF)	▪ any GPIOs ▪ RTC alarm ▪ Timer1

Table 60: Power modes, Digital Power Domains, Clocks, and Wake-Up Triggers

Power mode	Digital Power domains	LDOs, DC-DC Converter, and VDD level	Clock availability	RAM	Wake-Up from
Active or Sleep (WFI)	PD_AON = ON PD_SLP = ON PD_SYS = ON PD_TIM = OPTIONAL PD_RAD = OPT	VDD = 0.9 V DC-DC = ON (Buck) LDO_LOW = OFF LDO_CORE = ON, Active (0.9 V) VDD_Clamp = OFF LDO_RADIO = Programmable	All	SysRAM1 = ON (Application) SysRAM2 = optionally retained SysRAM3 = ON (Stack Data)	

Power mode	Digital Power domains	LDOs, DC-DC Converter, and VDD level	Clock availability	RAM	Wake-Up from
Extended Sleep (with or without OTP)	PD_AON = ON PD_SLP = ON PD_SYS = OFF PD_TIM = OPTIONAL PD_RAD = OFF	VDD = 0.75 V DC-DC = OFF LDO_LOW = ON, in Buck mode. LDO_CORE = ON, in retain mode (0.75 V) VDD_Clamp = OFF LDO_RADIO = OFF	RCX or XTAL32K	SysRAMx = optionally retained (typically only SysRAM1 is retained)	▪ any GPIOs ▪ RTC alarm ▪ Timer1 ▪ Bluetooth LE sleep timer
Deep Sleep	PD_AON = ON PD_SLP = ON PD_SYS = OFF PD_TIM = OPTIONAL PD_RAD = OFF	VDD = 0.75 V DC-DC = OFF LDO_LOW = ON, in Buck mode LDO_CORE = ON, in retain mode (0.75 V) VDD_Clamp = OFF LDO_RADIO = OFF	RCX or XTAL32K	SysRAMx = optionally retained (Typically OFF)	▪ any GPIOs ▪ RTC alarm ▪ Timer1
Hibernation	PD_AON = ON PD_SLP = OFF PD_SYS = OFF PD_TIM = OFF PD_RAD = OFF	VDD = ~0.75 V DC-DC = OFF LDO_LOW = OFF LDO_CORE = OFF VDD_Clamp = ~0.75 V LDO_RADIO = OFF	No Clocks	SysRAMx = optionally retained	P0_1, P0_2, P0_3, P0_4, P0_5

Table 61 shows the typical rail voltages and their drivers present during various PMU modes.

Table 61: Power rails drivers and voltages

Configurations	Mode	V_{BAT_HIGH}	V_{BAT_LOW}	V_{DD}
Buck	Active	Battery (3.6 V – 1.8 V)	DC-DC out (1.1 V)	LDO_CORE (0.9 V)
	Deep or Extended Sleep		LDO_LOW (1.1 V)	LDO_CORE in retain mode (0.75 V)
	Hibernation		0 V (none)	VDD_Clamp (~0.75 V)
Bypass	Active	Battery (3.3 V – 1.1 V)	Battery (3.3 V – 1.1 V)	LDO_CORE (0.9 V)
	Deep or Extended Sleep			LDO_CORE in retain mode (0.75 V)
	Hibernation			VDD_Clamp (~0.75 V)

Note 1 Parasitic diodes in DC-DC and LDO_LOW circuits prevent V_{BAT_HIGH} from dropping to 0 V. V_{BAT_HIGH} can also be programmed to be clamped to V_{BAT_LOW}.

9.2.5 VDD Level in Hibernation

While in Hibernation, the Always On domain (PD_AON) is supplied by a clamp. Since the reference is not enabled, the actual voltage supplied by the clamp depends on the load and temperature. To ensure proper operation of the PD_AON across the application operating temperature range and load, it is recommended to configure the voltage level of the VDD_Clamp using POWER_AON_CTRL_REG[LDO_RET_TRIM] according to Table 62.

Table 62: VDD_Clamp recommended settings over temperature and load

Temperature range	0 kB retained RAM	48 kB retained RAM
-40 °C to +40 °C	0xE	0xD

Temperature range	0 kB retained RAM	48 kB retained RAM
-40 °C to +60 °C	0xD	0xC
-40 °C to +85 °C	0xB	0xA

9.2.6 Retainable Registers

When the system enters one of the sleep modes, some registers need to retain their values even though their power domain might be shut down. These special retainable registers and their power domains are described in Table 63.

Table 63: Retainable registers

Power domains	Retainable registers
PD_SYS	OTPC_MODE_REG
	OTPC_TIM1_REG
	OTPC_TIM2_REG
	OTPC_AHBADR_REG
	OTPC_CELADR_REG
	OTPC_NWORDS_REG
	DEBUG_REG
PD_RAD	BLE_CNTL2_REG
	RF_ADCI_DC_OFFSET_REG
	RF_ADCQ_DC_OFFSET_REG
	RF_DC_OFFSET_RESULT_REG
	RF_DC_OFFSET_FULL_RES_REG
	RF_DC_OFFSET_MPAR_RES0/1/2/3_REG

9.2.7 Programming

9.2.7.1 Buck Configuration

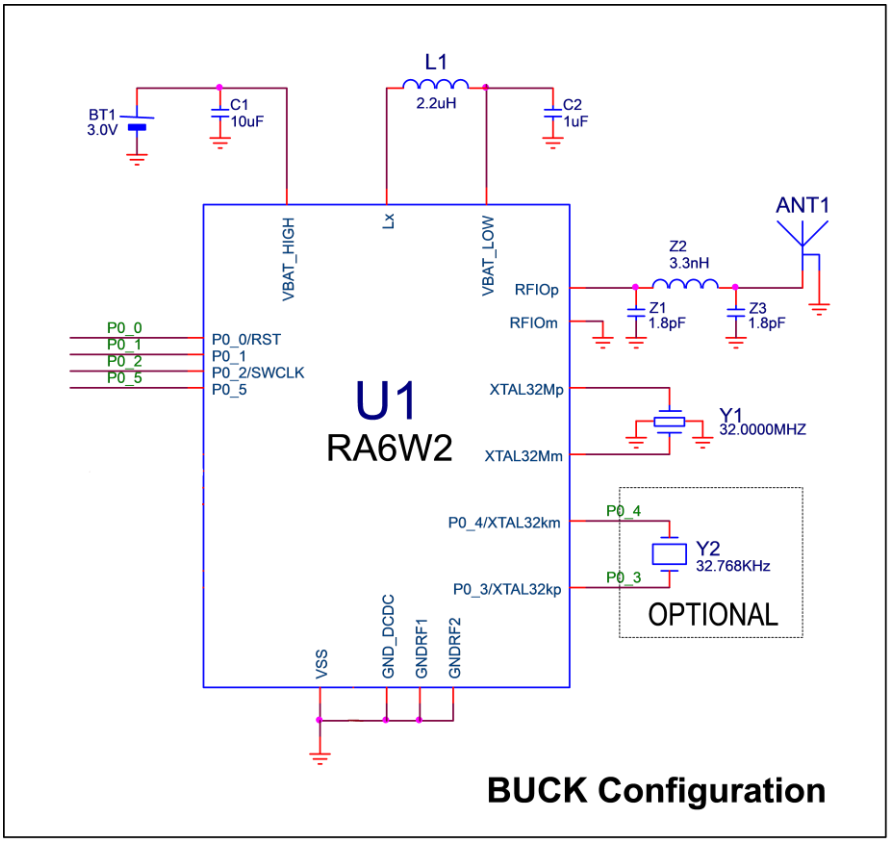


Figure 64. Buck configuration system diagram

In Buck configuration (Figure 61), the voltage on V_{BAT_LOW} and V_{DD} is generated from V_{BAT_HIGH} in the following ways:

- **Hibernation mode**

  In Hibernation mode, the V_{BAT_LOW} rail is not powered and the digital core V_{DD} is supplied by the VDD clamp.

  The VDD clamp is supplied automatically by selecting the highest of V_{BAT_HIGH} and V_{BAT_LOW}. Since in Buck configuration the V_{BAT_HIGH} rail is always the highest supply on the chip, it is safe to disable this automatic supply selection. Setting POWER_AON_CTRL_REG[CMP_VCONT_SLP_DISABLE] = 0x1 forces the clamp to use V_{BAT_HIGH} as supply and reduces the hibernation current by approximately 40 nA.

- **Extended Sleep and Deep Sleep modes**

  In the Extended Sleep mode or Deep Sleep mode, the V_{BAT_LOW} rail is supplied from LDO_LOW which is in a retention low power mode. The digital core V_{DD} is powered from LDO_CORE, retention mode. To configure this mode the following settings must be applied:

  - POWER_CTRL_REG[LDO_LOW_CTRL_REG] = 0x3
  - POWER_CTRL_REG[LDO_CORE_RET_ENABLE] = 0x1

- **Active and Sleep modes**

  The V_{BAT_LOW} rail is powered by LDO_LOW or by the DC-DC converter. To enable the DC-DC converter the following settings must be applied:

  - POWER_LEVEL_REG[DCDC_LEVEL] = 0x0
  - DCDC_CTRL_REG[DCDC_ENABLE] = 0x1

  The DC-DC converter is automatically disabled and re-enabled when PD_SYS is powered down and powered up again. Therefore, to use the DC-DC converter to power V_{BAT_LOW} rail, LDO_LOW has to be turned off again after the system wakes up by setting the register:

  - POWER_CTRL_REG[LDO_LOW_CTRL_REG] = 0x1

Register settings when the V_{BAT_LOW} rail is powered by LDO_LOW:

- POWER_CTRL_REG[LDO_LOW_CTRL_REG] = 0x3

9.2.7.2 Bypass Configuration

In the bypass configuration, the V_{BAT_HIGH} and V_{BAT_LOW} rails are shorted on the PCB. This configuration is detected by the chip as a boost configuration, but since the boost converter is not able to generate a voltage on $V_{BAT\_HIGH,}$ the initial voltage must be above 1.75 V to allow the OTP to be read and mirrored.

The software can disable the DC-DC converter and LDO_LOW to reduce quiescent current and avoid unnecessary switching of the DC-DC converter. The following register settings are required to accomplish this:

- DCDC_CTRL_REG[DCDC_ENABLE] = 0x0
- POWER_CTRL_REG[LDO_LOW_CTRL_REG] = 0x1

If the voltage drops below 1.75 V, POR_HIGH must be masked to prevent unnecessary resets. The OTP reads cannot be performed after this point. Masking POR_HIGH is done by the following setting:

- POWER_AON_CTRL_REG[POR_VBAT_HIGH_RST_MASK] = 0x1

When POR_HIGH is masked, its status remains available, but it does not generate a reset. POR_HIGH can be also disabled by the following setting:

- POWER_CTRL_REG[POR_VBAT_HIGH_DISABLE] = 0x1

When POR_ HIGH is disabled, its status becomes unavailable.

9.3 Hardware FSM (Power-Up, Wake-Up, and Go-to-Sleep)

The Hardware Finite State Machine (FSM) responsible for the power-up, wake-up, and go-to-sleep processes of the system is presented in Figure 65.

NOTE
Boost Power mode is not included in the system configuration. Any mention or explanation of Boost Power mode is intended for debug purposes only.

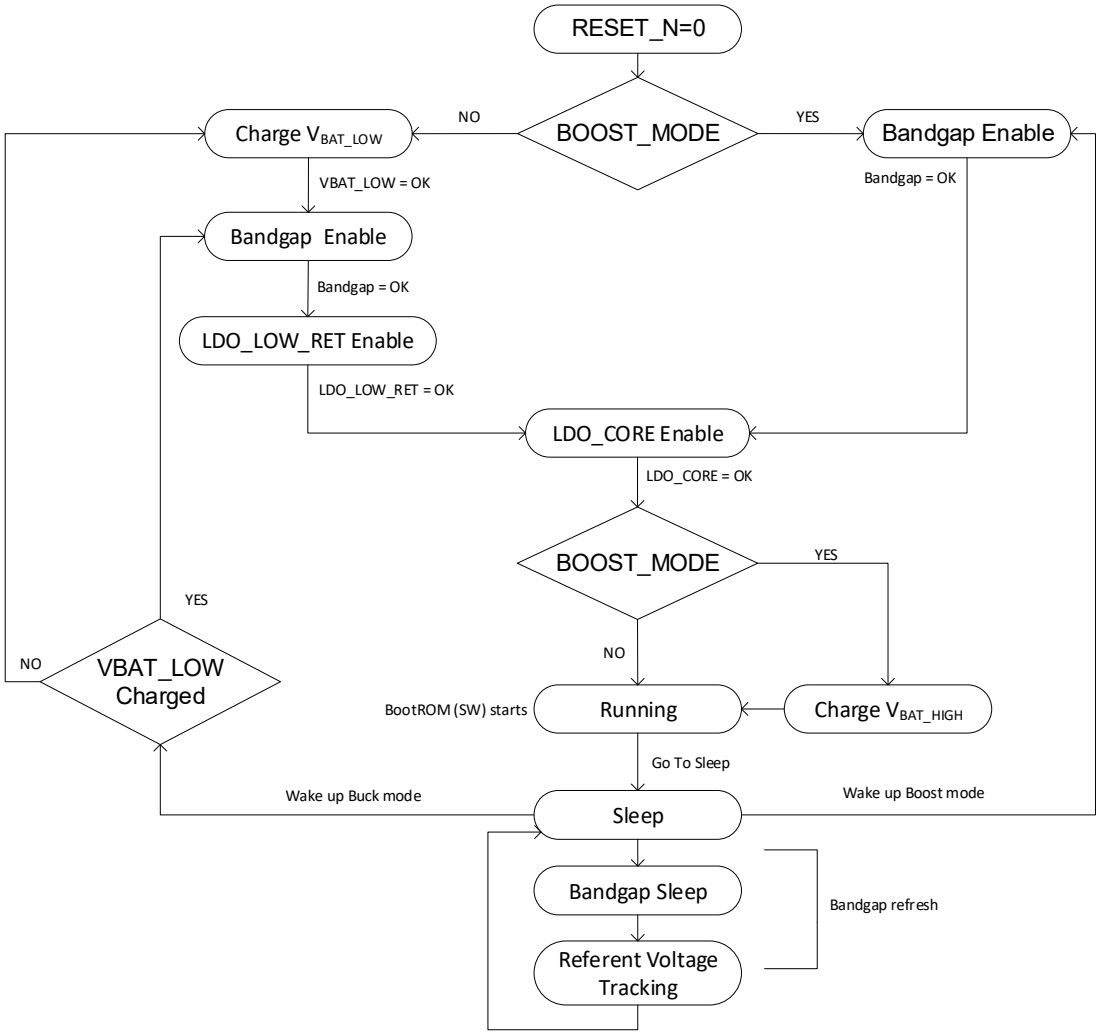


Figure 65. Power-Up/Wake-Up/Sleep FSM diagram

The details of the power-up, wake-up, and sleep sequences of the FSM for the different modes are described in the following sections.

9.3.1 Power-Up/Wake-Up in Buck Configuration

At the beginning of a power-up (cold boot), the PMU detects whether the system is in a Buck or a Boost configuration, and this decision is retained from that point on. The power-up (cold boot) sequence of the Buck configuration is shown in Figure 66. When the system is at the start of the Buck configuration path, then V_{BAT_HIGH} rail has a stable supply already and "`boost_mode` = 0" (which means buck is identified) has been set.

To start the system, it is required that the V_{BAT_LOW} rail is also brought up to an acceptable level, which is done by hardware, enabling the resistive switch VBAT_HL_RES and monitoring POR_LOW. The rising voltage of the V_{BAT_LOW} rail eventually triggers POR_LOW to "ok" after that, the bandgap is enabled. The hardware FSM runs at 32 kHz from the RC32K oscillator at this time, and it dynamically changes to 512 kHz in one clock cycle after the switch is enabled. The hardware FSM continues working at 512 kHz. When the reference voltage from the bandgap is stable, LDO_LOW is enabled. After a stable 1.1 V is generated, LDO_CORE is enabled to generate a stable V_{DD} of 0.9 V. After this, the main system clock, RC32MHz, is enabled. All conditions are now in place to release the system reset so that the Booter can start running. An indicative time needed to power up the system in Buck configuration until the application software starts running, is around 2.5 ms. The time required to charge V_{BAT_LOW} and V_{BAT_HIGH} depends on the external capacitor values on these rails.

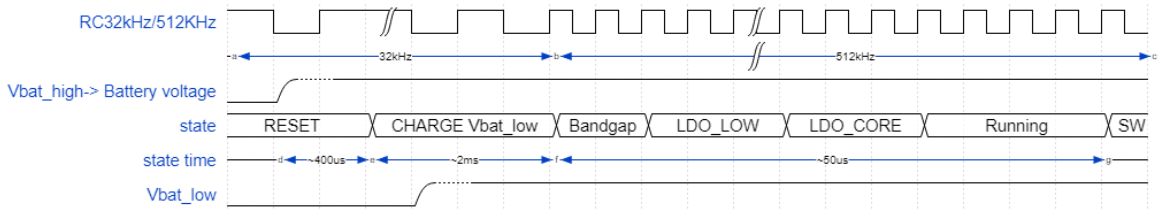


Figure 66. Power-Up (Buck)

In hibernation, the resistive switch VBAT_HL_RES can be closed to pre-charge V_{BAT_LOW} to the level of V_{BAT_HIGH} by programming POWER_AON_CTRL_REG[VBAT_HL_CONNECT_RES_CTRL]. Therefore, during the wake-up sequence from the hibernation mode, step "Charge V_{BAT_LOW}" can be omitted (Figure 67) via POWER_AON_CTRL_REG[CHARGE_VBAT_DISABLE]. All other steps are the same as in the power-up cold-boot sequence. The total time needed to wake up the system from hibernation up until booter software starts running is around 185 µs.

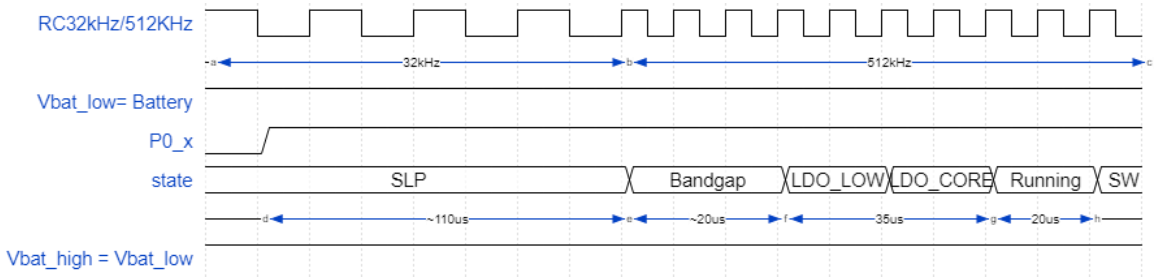


Figure 67. Wake-Up from hibernation (Buck)

The wake-up sequence from the clocked, extended sleep, or deep sleep mode using an external GPIO toggle is shown in Figure 68. The difference between the power-up and wake-up sequence in the Buck configuration is that the V_{BAT_LOW} rail is already charged via the switch VBAT_HL_RES in the wake-up sequence as shown in Figure 65. Therefore, when the system wakes up from a deep sleep or an extended deep sleep, the bandgap is enabled within one 32 kHz clock cycle after the wake-up signal is triggered. After the bandgap is enabled, LDO_LOW and LDO_CORE are enabled one after the other. The total time which is needed to wake up the system until the software starts running is around 800 µs. If RAM has been retained, running software means application, if not then the Booter is started.

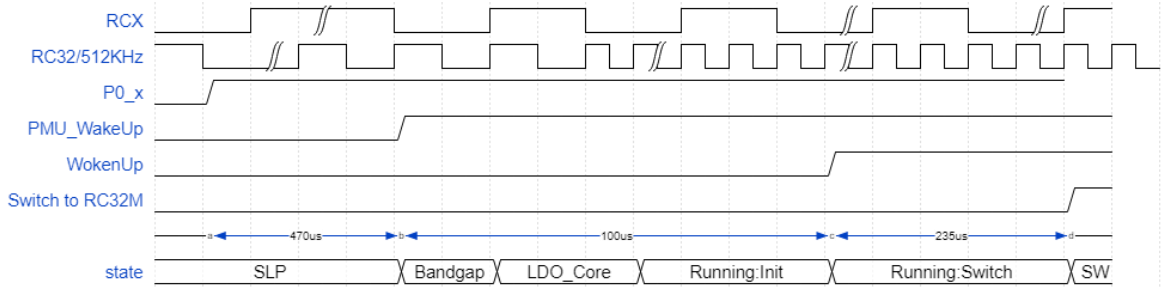


Figure 68. Wake-Up (Buck)

A GPIO trigger should go through the wake-up controller first, which requires 7 RCX clock cycles before it is allowed to trigger the PMU and have the state machine running. Even after all power rails are done, switching the system clock from RCX into RC32M (so software starts running) takes 3.5 RCX clock cycles (indicated in state = RUNNING:SWITCH) in Figure 68.

If the system wakes up from an internal timer running at the RCX clock, then the WokenUp signal is asserted 6 RCX clock cycles after the timer generates the interrupt (for example, ~400 µs).

9.3.2 Go-to-Sleep and Refresh Bandgap

The sleep state disables the power-consuming blocks and triggers the "hold mode" for the bandgap referenced voltages. After a certain amount of time (sleep refresh counter), these "hold" voltages need to be refreshed. The lower loop of the FSM in Figure 65 enables the bandgap, refreshes the voltages, checks for BOD events via the POR circuits and if ok, resets the refresh timer and goes back to sleep. This is an autonomous cycle led by hardware until the system is woken up by a wake-up event. The refresh timer can be configured by setting the

PMU_SLEEP_REG [BG_REFRESH_INTERVAL] bit field (1 LSB = 64 × 32 kHz clock cycles). The go-to-sleep and the bandgap refresh sequence are shown in Figure 69.

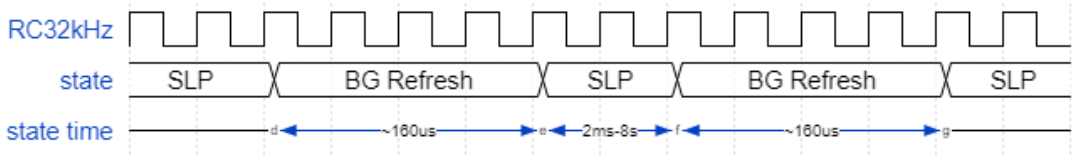


Figure 69. Go-to-sleep and bandgap refresh

9.4 OTP Memory Layout

The OTP memory must 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 Figure 70.

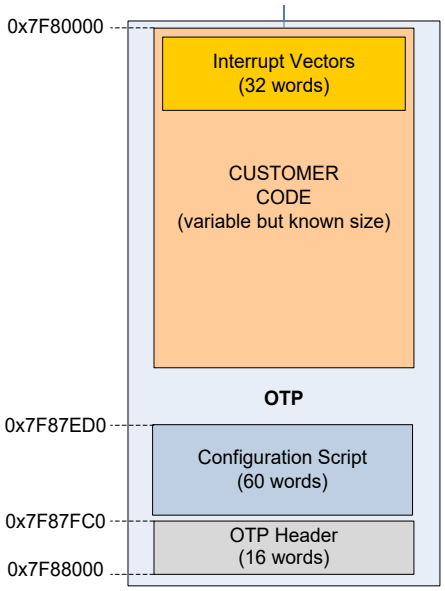


Figure 70. OTP layout scheme

The OTP memory is a matrix of 8Kx32-bit words. The contents are described below:

- **Interrupt Vectors:** they are the vectors of the interrupt service routines and always reside at the address 0x0. This is part of the application (customer) code. The size of this vector list is 32 words
- **Customer Code:** it contains the applications and the profiles that a customer has developed. The size is known and fixed before the mass production and the programming of the OTP
- **Configuration Script (**Section 9.4.2**):** it is used to program registers with values that are defined during production testing, to store a trim value for the application software, and to define the UART time-out timer during booting. It is executed by the Booter to prepare and initialize the system before the CPU starts running the application code. Available size is 60 words
- **OTP Header:** it contains various information about the configuration of the system and the Bluetooth LE-specific data. The size of the header is 16 words

9.4.1 OTP Header

The OTP header breakdown is presented in Table 64.

Table 64: OTP header

Address	Words (32-bit)	Description	Programmed during	
			Chip test	Product manufacturing
7F87FC0	1	Application Programmed Flag #1 0x1234A5A5 = Application is in OTP		Yes
7F87FC4	1	Application Programmed Flag #2 0xA5A51234 = Application is in OTP		Yes

7F87FC8	1	Boot-specific configuration:
		▪ Bits[7:0] :
		  ▪ 0xAA = Boot from SPI port at a specific location
		  ▪ 0xFF = Normal sequence
		▪ Bits[15:8] = Wake up Command opcode
		▪ Bits[23:16] = SPI_DIV
		▪ Bits[31:24]:
		  ▪ 0x00 = Two-wire UART (P0_0/P0_1)
		  ▪ 0x01 = One-wire UART (P0_3)
		  ▪ 0x02 = One-wire UART (P0_5)
		  ▪ Default (all other values) = Two-wire UART (P0_0/P0_1)
				Yes

7F87FCC	1	Boot-specific port mapping: Bits[7:4] = SPI_CLK, Port number Bits[3:0] = SPI_CLK, Pin number Bits[15:12] = SPI_EN, Port number Bits[11:8] = SPI_EN, Pin number Bits[23:20] = SPI_DO, Port number Bits[19:16] = SPI_DO, Pin number Bits[31:28] = SPI_DI, Port number Bits[27:24] = SPI_DI, Pin number		Yes
7F87FD0	1	Reserved	Yes	
7F87FD4	2	Bluetooth Device Address (64-bit word). It is handled as a string of bytes.		Yes
7F87FDC	1	OTP DMA length (number of 32-bit words).		Yes
7F87FE0	1	Position: Bits[7:0] = X coord Bits[15:8] = Y coord Bits[23:16]= Wafer # Bits[31:24] = LOT #	Yes	
7F87FE4	1	Tester: Bits[7:0] = Tester_Site Bits[15:8] = Tester_ID (LSB) Bits[23:16] = Tester_ID (MSB) Bits[31:24] = Reserved	Yes	
7F87FE8	1	TimeStamp: Bits[7:0] = TS_Byte0 Bits[15:8] = TS_Byte1 Bits[23:16] = TS_Byte2 Bits[31:24] =TS_Byte3	Yes	

Address	Words (32-bit)	Description	Programmed during	
			Chip test	Product manufacturing
7F87FEC	5	Reserved for Future Needs		

The Device and Package Flag reflects what the current device (RA6W2) is and which package is used. Default (unprogrammed) values are 0xFFFFFFFF.

Boot-specific mapping value is used to define a specific configuration for the SPI interface when used for booting from an external device (either an MCU or a FLASH). Byte0 is the flag to instruct the BootROM to use the specific SPI pin mapping and skip the rest of the serial peripheral interfaces. The BootROM takes care of waking up an external flash when the flash memory is in a deep power-down state.

Byte1 is used for the Wake-Up Command opcode that the flash memory responds to. If Byte0 is left unprogrammed, BootROM sends the "0xAB" opcode by default. Furthermore, the BootROM can wake up the external flash by toggling the CS pin.

Two more flags indicate whether the application code has indeed been programmed (burned) into the OTP. Both flags are read by the BootROM software designating that the system is in the Normal mode and not in the Development mode (Section 9.5).

9.4.2 Configuration Script

The Configuration Script (CS) is a table of 32-bit entries and is 60 words deep, so in total, the CS can utilize 240 bytes of space.

The CS is used to program registers with values that are defined during production testing, to store a trim value for the application software, and to define the UART time-out timer during booting. It is executed by the Booter to prepare and initialize the system before the CPU starts running the application code.

The format of the commands in the CS is presented in Table 65.

Table 65: CS commands and description

#	Command type	Description
1	Start Command	One 32-bit word containing 0xA5A5A5A5 to signal a valid CS is in place.
2	Register Configuration	▪ One 32-bit word containing an address of an existing register ▪ One 32-bit word containing the data value of the register These are always in pairs with the address sitting in even memory addresses.
3	SDK Value	One 32-bit word which is equal to 0x9000YYXX indicating that the next word is a value stored during production testing. More specifically: ▪ 9: it indicates that the following word(s) are not to be stored to registers but are used by the SDK software ▪ YY: it indicates that YY amount of words follow ▪ XX: it is an increasing value and can be used for indexing by the software application. If YY > 1, XX is not increased for the words that belong to the same value. One or more 32-bit words can represent one value.
4	SWD mode	One 32-bit word which is equal to 0x70000000. It prevents the JTAG from being enabled at the end of the Booter and the Booter does not enter the endless while (1) loop. Instead, it continues to rescan all peripherals in the development mode path.
5	UART STX timeout value	One 32-bit word which is equal to 0x8XXXXXXX. The XXXXXXX is used to program the selected STX timeout in multiples of 100 µs. So, for example, 0x80000028 is 40 × 100 µS = 4 ms.
6	SPI Clock value	0xA0000000 This value overwrites the default 2-MHz clock speed of the SPI boot path and sets it to 32 MHz.

The Booter stops processing the CS once it encounters an empty OTP value (0xFFFFFFFF). This way, no more processing time is spent checking the rest and it is possible to add new entries later, for example, to patch/update previous entries.

An example describing the format of the configuration script is presented in Table 66.

Table 66: CS example

Words	Even Words	Odd Words	Description
0-1	0xA5A5A5A5	0x80000028	Start command of the CS Script, followed by STX timeout value of 4 ms (40 × 100 µs)
2-3	<Address>	<Value>	Booter automatically writes the <Value> to the <Address>
4-5	0x90000301	<Value>	Three calibration values stored during production testing. SDK should know what this is for.
6-7	<Value>	<Value>	
8-9	<Address>	<Value>	Booter automatically write the <Value> to the <Address>
10-11	<Address>	<Value>	Booter automatically write the <Value> to <Address>
12-13	0x90000402	<Value1>	Four calibration values stored during production testing. SDK should know what this is for.
14-15	<Value2>	<Value3>	Calibration value stored during production testing. SDK should know what this is for.
16-17	<Value4>	0x70000000	Disable SWD
18-19	0xFFFFFFFF	(don’t care)	Booter stops running the CS after an empty entry, so anything after this is "don’t care".

9.5 BootROM Sequence

The booting process of RA6W2 is presented in Figure 71. The Booter is always executed when a POR or an hardware reset occurs, or the RESET_ON_WAKEUP feature is configured.

The booter starts executing with the RC32M clock, to speed up its execution. Then the Booter checks whether the system is in the Boost configuration or not. The configuration (Buck or Boost) has been already decided by the hardware and should be readable by software at the ANA_STATUS_REG[BOOST_SELECTED] bit field. To access the OTP, V_{BAT_HIGH} needs to be set at ≥ 1.8 V. In the Boost configuration the DC-DC is enabled to boost V_{BAT_HIGH} at 1.8 V and BOOST_VBAT_OK = 1 confirms that the voltage at the DC-DC is stable. After the OTP is operational, the Booter initializes the UART baud rate at 115.2 kHz, and the CS (Section 9.4.2) is enabled to be executed.

After the CS has been executed, the Booter must decide whether the device is in Development or Normal mode by reading the two words indicated as application flags in the OTP. The OTP image is copied into RAM starting at address 0x0 by the Booter.

In Development mode, the "Boot from Specific" flag is evaluated. If the flag is programmed, new pin locations for booting from an external SPI slave to make RA6W2 an SPI Master is set. The "Boot from Specific" flag addresses mostly the QFN package allowing for booting from a different pin configuration than the default one, so that the system can boot from an external FLASH using the development mode. The details of the configuration are presented in Section 9.4.1. If this path is entered, the system always tries to boot from UART so that the SPI Flash can be updated if needed. Any of the three UART configurations specified in Table 67 can be selected by writing bits [31:24] at the "Boot specific config" field in the OTP header. If booting from SPI Flash fails, the Booter jumps back to the normal scan sequence of the peripheral devices.

If the "Boot from Specific" flag is not programmed, the system should continue with scanning the different serial interfaces to identify whether a device is connected to it. After OTP is disabled, six steps as described in Table 67 are performed. Before using the UART, the XTAL32M clock needs to be enabled. All the boot steps are protected by a timeout.

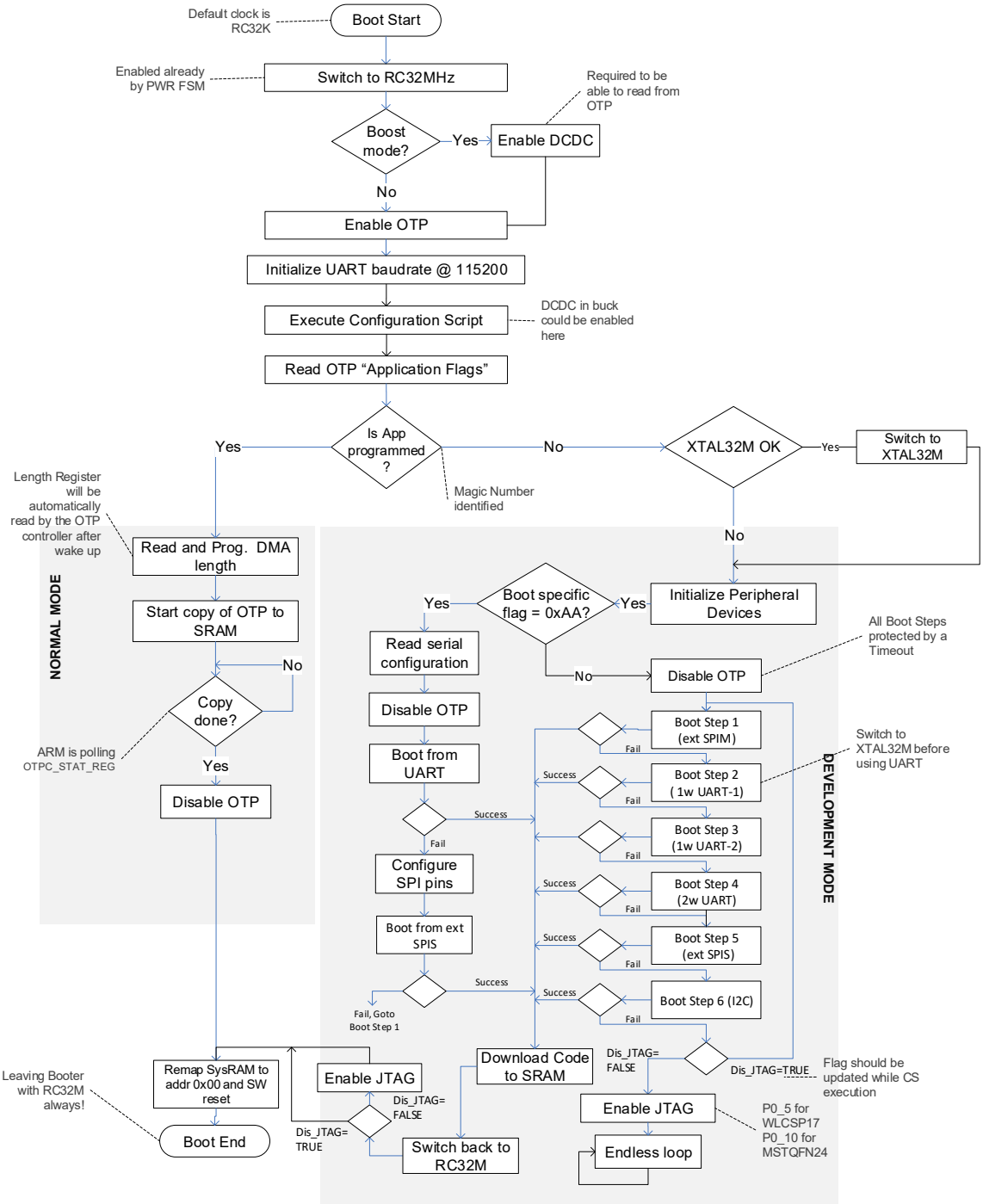


Figure 71. BootROM sequence

Since the booting from the UART protocol is a half-duplex, a single GPIO is used in RA6W2 for the external UART. The protocol is the same as for a two-wire UART booting except that the Booter software needs to change the pin direction before sending or receiving information.

Table 67: Booting sequence steps

	Step 1: Boot from external SPI master	Step 2: Boot from 1-wire UART (First option)	Step 3: Boot from 1-wire UART (Second option)	Step 4: Boot from 2-wire UART	Step 5: Boot from external SPI Slave	Step 6: Boot from I2C
P0_0/RST	MISO			Tx	MOSI	
P0_1	MOSI			Rx	SCS	

	Step 1: Boot from external SPI master	Step 2: Boot from 1-wire UART (First option)	Step 3: Boot from 1-wire UART (Second option)	Step 4: Boot from 2-wire UART	Step 5: Boot from external SPI Slave	Step 6: Boot from I2C
P0_2						
P0_3	SCS		RxTx		MISO	SDA
P0_4	SCK				SCK	SCL
P0_5		RxTx (Default)				
P0_6						
P0_7						
P0_8						
P0_9						
P0_10						
P0_11						

If no bootable devices are found on any of the serial interfaces, the Booter can do two things, depending on what is stored in the CS. If the "Debugger disable" (0x70000000) command is stored there, the Booter starts scanning for peripherals again. Otherwise, it enters the endless loop with the debugger (JTAG) being enabled. The debugger can be connected to P0_5 pin.

After the BootROM sequence has been completed, the default system clock is RC32M, regardless of which boot path has been chosen and all GPIOs are set back to their default reset values.

9.6 Reset

9.6.1 Introduction

The RA6W2 comprises a reset (RST) pad which is active high. It contains an RC filter with a resistor of 465 kΩ and a capacitor of 3.5 pF to suppress spikes. It also contains a 25 kΩ pull-down resistor. This pad should be driven externally by a field-effect transistor (FET) or a single button connected to VBAT. The typical latency of the RST pad is in the range of 2 µs.

Features

- RC spike filter on RST to suppress external spikes (465 kΩ, 3.5 pF)
- Three different reset lines (software, hardware, and POR)
- Latching the cause of a reset operation (RESET_STAT_REG)
- Configurable POR circuitry.

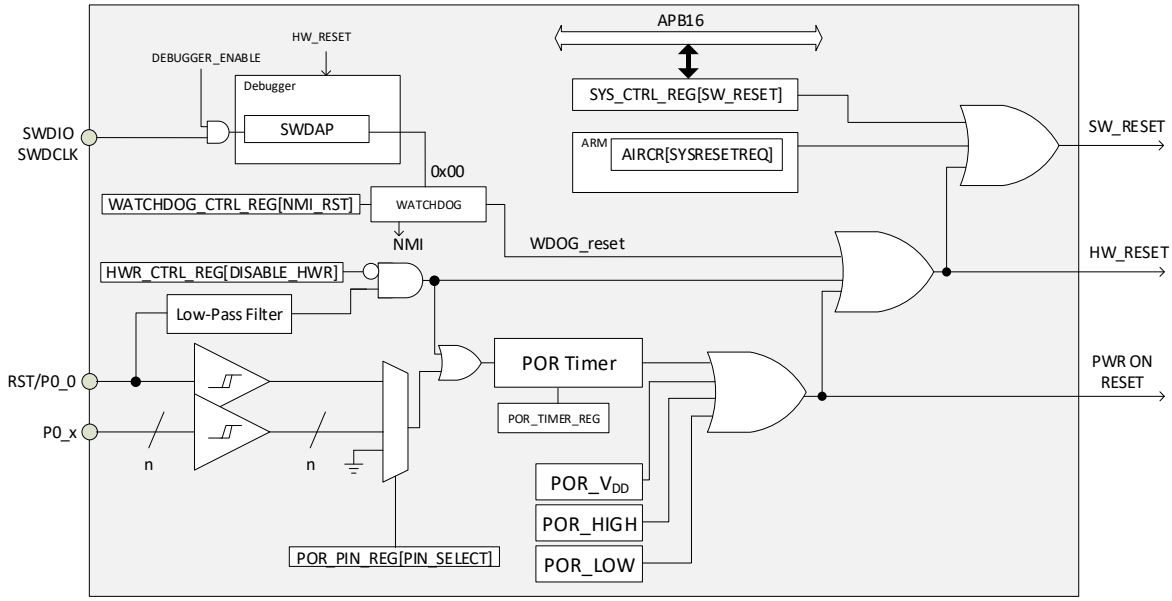


Figure 72. Reset block diagram

9.6.2 Architecture

9.6.2.1 POR, Hardware, and Software Reset

There are three main reset signals in the RA6W2:

- The Power-On Reset (POR): it is optionally triggered by a GPIO set as the POR source with a selectable polarity and/or the RST pad (P0_0) after a programmable time delay.
- The hardware reset: it is optionally triggered by the RST pad (P0_0) when it becomes active for a short period of time (less than the programmable delay for POR).
- The software reset: it is triggered by writing the SYS_CTRL_REG[SW_RESET] bit.

The POR signal is generated:

- Internally and releases the system's flip flops as soon as the VDD, V_{BAT_HIGH}, and V_{BAT_LOW} voltages crossed the specified thresholds.
- Externally by a POR source (RST pad multiplexed on a GPIO or P0_0 configured as RST pin).

The hardware reset can also be automatically activated when the system wakes up from the Extended or Deep Sleep mode by programming the bit PMU_CTRL_REG[RESET_ON_WAKEUP]. The POR and the hardware reset basically run the cold start-up sequence and the BootROM code is executed.

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

The block diagram of the reset block is shown in Figure 72.

Certain registers are reset by POR only, or by POR and the hardware reset signal but not by the software reset. These registers are listed in Table 68.

Table 68: Reset signals and registers

Reset by POR only	Reset by POR or hardware reset	Reset by POR, hardware reset, or software reset
BANDGAP_REG	BLE_CNTL2_REG	The rest of the Register File
POR_PIN_REG	CLK_AMBA_REG[OTP_ENABLE]	
POR_TIMER_REG	CLK_FREQ_TRIM_REG	
HWR_CTRL_REG	CLK_RADIO_REG	
RESET_STAT_REG[PORESET_STAT]	CLK_CTRL_REG	

Reset by POR only	Reset by POR or hardware reset	Reset by POR, hardware reset, or software reset
PAD_LATCH_REG	PMU_CTRL_REG	
POWER_AON_CTARL_REG	SYS_CTRL_REG	
GP_DATA_REG	TRIM_CTRL_REG	
TEST_VDD_REG	RAM_PWR_CTRL_REG	
	CLK_RC32K_REG	
	CLK_XTAL32K_REG	
	CLK_RC32M_REG	
	CLK_RCX_REG	
	XTALRDY_CTRL_REG	
	XTAL32M_CTRL0_REG	
	PMU_SLEEP_REG	
	POWER_CTRL_REG	
	POWER_LEVEL_REG	
	DCDC_CTRL_REG	
	RAM_LPMX_REG	
	HIBERN_CTRL_REG	
	CLK_RTCDIV_REG	
	RTC_CONTROL_REG	
	RTC_KEEP_RTC_REG	
	OTPC_*_REG	
	QDEC_*_REG	
	All RF calibration registers	

9.6.2.2 POR Functionality

The POR functionality is available on two sources:

- RST Pad: the RST pad is always capable of producing a POR.
- GPIO Pin: a GPIO can be selected by the user application to act as a POR source.

The time needed for a GPIO pin selected for the POR to be active is stored in the POR_TIMER_REG. The register field POR_TIME is a 7-bit field that holds the time factor by which the total time for POR is calculated. The maximum value of the field is 0x7F. The total time for POR is calculated by the following formula:

$$\text{Total time} = \text{POR\_TIME} \times 4096 \times \text{RC32k clock period} \quad (1)$$

where RC32k clock period = 31.25 µs at 25 °C.

The maximum time for which a POR can be performed is ~16.2 s at 25 °C.

The RC32k clock frequency depends on temperature, so based on the temperature span of -10 °C to 50 °C, the clock frequency range is calculated to be 25 kHz to 39 kHz. Then,

$T_{PORcold}$ = 13 s

T_{PORhot} = 20.8 s

9.6.2.2.1 POR Timer Clock

The POR timer is clocked by the RC32k clock. If a software application disables the RC32k, the hardware takes care of enabling the RC32k clock when a POR source (the RST pad or a selected GPIO pin) is asserted. It should be noted that if the POR is generated from the RST pad, the RC32k operates with the reset (default) trimming value. If a GPIO pin is used as the POR source, the RC32k clock is trimmed. The timing difference between both cases is expected to be minor.

9.6.2.2.2 RST Pad

The RST pad produces a hardware reset if the pin active time is less than the programmed value in the POR_TIMER_REG register or a POR if the pin active time is greater than or equal to that value. Reset pad is always Active High.

9.6.2.2.3 POR from GPIO

When a GPIO is used as a POR source, the selected pin retains its capability to act as GPIO. The POR_PIN_REG[PIN_SELECT] field holds the required GPIO pin number. If the value of the PIN_SELECT field equals 0, the POR triggered by GPIO functionality is disabled. The polarity of the pin can be configured by the POR_PIN_REG [POR_POLARITY] bit, where 0 means Active Low and 1 means Active High.

9.6.2.3 POR Timing Diagram

The operation of the POR triggered by both the RST pad and a selected GPIO pin is shown in Figure 73.

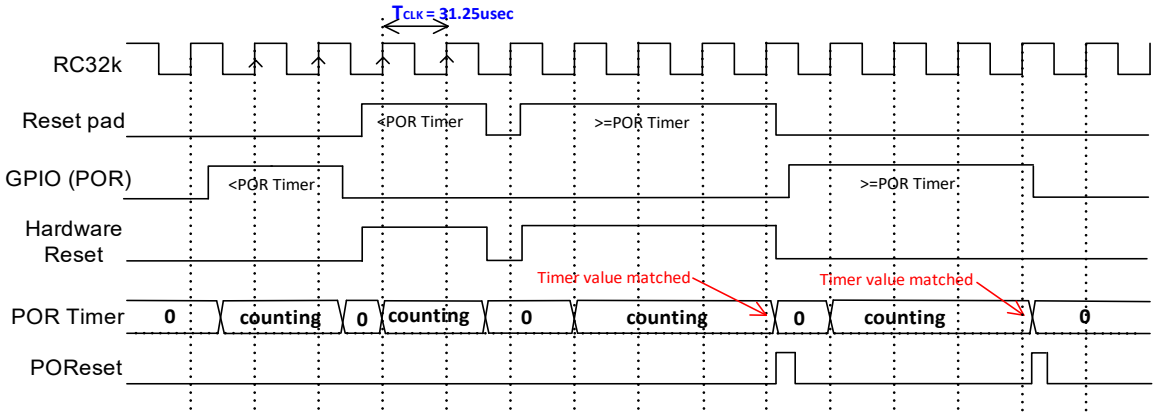


Figure 73. POR timing diagram

9.6.2.4 POR Considerations

When a POR source (the RST pad or a selected GPIO pin) is asserted, the POR timer starts to count. When the POR source is released before the timer has expired, the POR timer is reset to 0. If a POR source is asserted while there is already an asserted POR, and the first POR is released after the second POR is asserted, and the total time of the two asserted sources is larger than or equal to the POR_TIME, POR occurs.

It should also be noted that the POR timer triggered by the RST pad can only expire once. After the POR timer has expired, the RST pad must be released so the timer can be reloaded. There is no such limitation when a GPIO is used as the POR source.

The POR_PIN_REG[PIN_SELECT] field cannot survive any reset (POR, hardware reset, or software reset), therefore, you must take special care on setting up the GPIO POR source right after a reset. This also applies to the POR_TIMER_REG[POR_TIME] field after a POR.

Be aware that, if a GPIO is used as a POR source, the dynamic current of the system increases due to the dynamic current consumed by the RC32k oscillator. This increase is calculated to be from 100 nA to 120 nA and it is also present during the sleep time period. POR from the RST pad does not add this dynamic current consumption.

9.6.3 Programming

To configure the functionality of triggering a POR by a GPIO pin, complete the following steps:

1. Select a GPIO to be set as the POR source by programming POR_PIN_REG[POR_PIN_SELECT].

2. Set up the input polarity of the GPIO that causes POR by programming POR_PIN_REG[POR_PIN_POLARITY].
3. Configure the time for the POR to happen by programming POR_TIMER_REG[POR_TIME]. The default time is around three seconds.

NOTE
To set up the time when the RST pad produces a POR, just set the POR_TIMER_REG register.

9.7 Arm Cortex-M0+

9.7.1 Introduction

The Arm 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, but it also uses several newer instructions from the Armv6 architecture and a few instructions from the Thumb-2 technology. Thumb-2 technology extends 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. It has a two-stage pipeline (fetch+predecode and decode+execute) as opposed to its predecessor (Cortex-M0) that has a three-stage pipeline (fetch, decode, and execute).

A simplified block diagram of the Cortex-M0+ is shown in Figure 74.

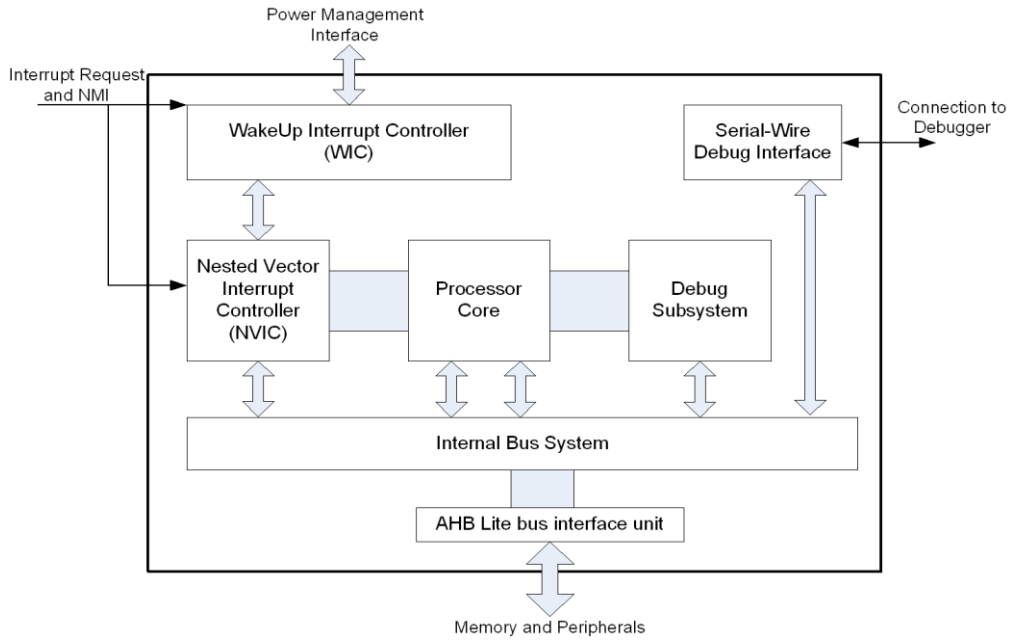


Figure 74. Arm Cortex-M0+ block diagram

Features

- Thumb instruction set: highly efficient, of high code density, and able to execute all Thumb and Thumb-2 instructions
- 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 made, 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 and the NVIC automatically handles nested interrupts
- The design is configured to respond to exceptions (for example, interrupts) as soon as possible (minimum 15 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 assemblers
- 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 Pendable SuperVisor service (PendSV) 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 the sleep modes are entered by specific instructions rather than implementation defined control registers
- Fault handling exception to catch various sources of errors in the system
- Support for 21 interrupts
- Little endian memory support
- Wake-Up Interrupt Controller (WIC) to allow the processor to be powered down during sleep, while interrupt sources are still allowed 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 SWD 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 supports 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.

9.7.2 Architecture

9.7.2.1 Interrupts

This section lists all 21 interrupt lines, except the NMI interrupt, and describes their sources and functionality. The overview of the interrupts is shown in Table 69.

Table 69: Interrupt List

IRQ Number (Inherent priority)	IRQ name	Description
0	BLE_WAKEUP_LP_IRQn	Wake up the system from Low Power (Extended Sleep) interrupt from Bluetooth LE.
1	BLE_GEN_IRQn	Bluetooth LE Interrupt. Sources: ▪ BLE_FINETGTIM_IRQn: Fine Target Timer interrupt generated when Fine Target timer expires. The timer resolution is 625 µs base time reference

IRQ Number (Inherent priority)	IRQ name	Description
		▪ BLE_GROSSTGTIM_IRQn: Gross Target Timer interrupt generated when Gross Target timer expired. The timer resolution is 16 times 625 µs base time reference ▪ BLE_CSCNT_IRQn: 625 µs base time reference interrupt, available in active modes ▪ BLE_SLP_IRQn: End of Sleep mode interrupt ▪ BLE_ERROR_IRQn: Error interrupt, generated when undesired behavior or bad programming occurs in the Bluetooth LE Core ▪ BLE_RX_IRQn: Receipt interrupt at the end of each received packets ▪ BLE_EVENT_IRQn: End of Advertising/Scanning/Connection events interrupt ▪ BLE_CRYPT_IRQn: Encryption/Decryption interrupt, generated when AES and/or CCM processing is finished ▪ BLE_SW_IRQn: Software triggered interrupt, generated on software request.
2	UART_IRQn	UART interrupt.
3	UART2_IRQn	UART2 interrupt.
4	I2C_IRQn	I2C interrupt.
5	SPI_IRQn	SPI interrupt.
6	ADC_IRQn	Analog-Digital Converter interrupt.
7	KEYBRD_IRQn	Keyboard interrupt.
8	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.
9	RF_CAL_IRQn	RF Calibration Interrupt.
10	GPIO0_IRQn	GPIO interrupt through debounce.
11	GPIO1_IRQn	GPIO interrupt through debounce.
12	GPIO2_IRQn	GPIO interrupt through debounce.
13	GPIO3_IRQn	GPIO interrupt through debounce.
14	GPIO4_IRQn	GPIO interrupt through debounce.
15	SWTIM_IRQn	Timer0/2 interrupt.
16	WKUP_QUADEC_IRQn	Combines the Wake-Up Capture Timer interrupt, the GPIO interrupt, and the QuadDecoder interrupt.
17	TIM1_IRQn	Timer1 interrupt.
18	RTC_IRQn	Real Time Clock interrupt.
19	DMA_IRQn	DMA interrupt.
20	XTAL32RDY_IRQn	XTAL32M settling ready interrupt.

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 69 (a lower interrupt number has a higher priority level).

To access the Cortex-M0+ NVIC registers, the Cortex Microcontroller Software Interface Standard (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 conveniently) the corresponding IRQ name listed in Table 69. For example, the corresponding interrupt handler name in the vector table for IRQ#15 is SPI_Handler. For more information on the

Arm Cortex-M0+ interrupts and the corresponding CMSIS functions, *see* section 4.2 Nested Vectored Interrupt Controller in the Cortex-M0+ Devices Generic User Guide.

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

9.7.2.2 System Timer (SysTick)

The Cortex-M0+ System Timer (SysTick) can be configured for using two 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 (for example, 2, 4, 8, or 16 MHz).

The default SysTick Timer configuration uses 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 about the timing.

9.7.2.3 Wake-Up Interrupt Controller

The Wake-Up Interrupt Controller (WIC) is a peripheral that can detect an interrupt and wake the processor from Extended Sleep mode. The WIC is enabled only when the SLEEPDEEP bit in the system control register is set to 1 (see System Control Register in the Cortex-M0+ Technical Reference Manual).

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 Extended 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 the interrupt latency is increased in Extended Sleep mode.

9.7.3 Programming

For more information on the Arm Cortex-M0+, see the documents listed in Table 70.

Table 70: Arm documents list

	Document title	Arm document number
1	Cortex-M0+ Devices Generic User Guide	Arm DUI 0662B (available on the website)
2	Cortex-M0+ Technical Reference Manual, r0p1	Arm DDI 0484C (available on the website)
3	Armv6-M Architecture Reference Manual	Arm DDI 0419C (can be downloaded by registered customers)

9.8 AMBA Bus

9.8.1 Introduction

The RA6W2 is based on the AMBA 2.0 AHB and APB components. The AHB is an AMBA Lite version which requires a single master on the system, but there is arbitration between the Arm Cortex-M0+ CPU and the Direct Memory Access (DMA) engine. There are two APB bridges, one for APB16 and the other for APB32, implementing three different decoded slaves which are grouped according to the power domain structure of the chip.

The AMBA bus organization is shown in Figure 75.

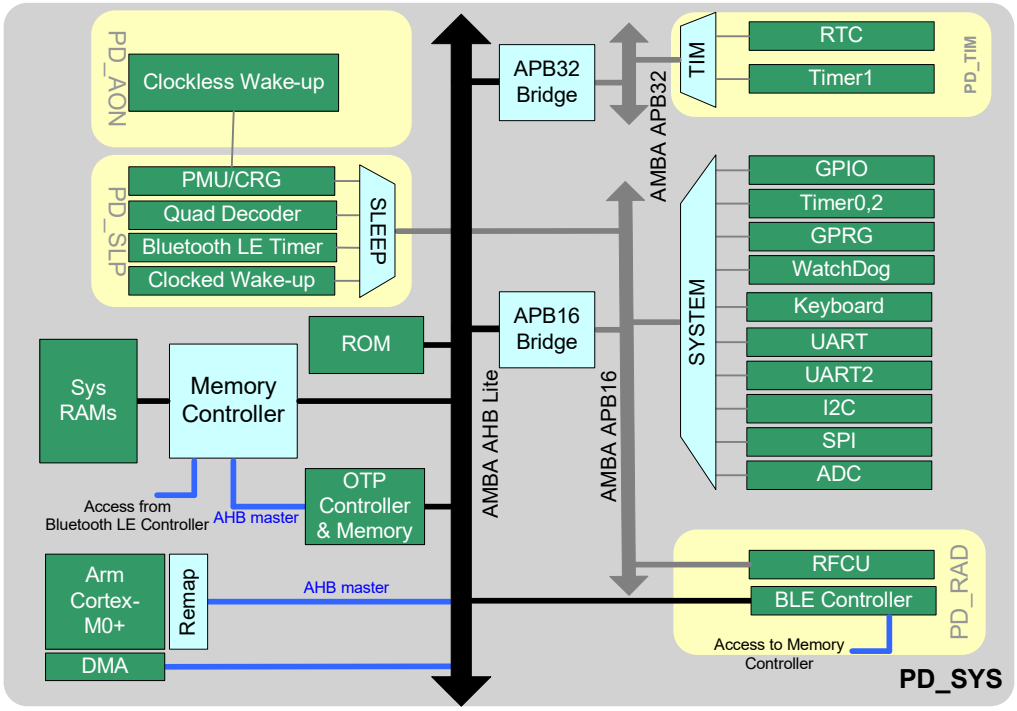


Figure 75. AMBA bus architecture and power domains

9.8.2 Architecture

Since the RA6W2 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 or the DMA can be the master, while OTP, Bluetooth LE Core, Memory and ROM controllers are slaves.

The Always On power domain (PD_AON) contains only the clock-less wake-up controller and the start-up hardware FSM responsible for the activation of the power devices within the system.

The sleep power domain (PD_SLP) contains the clock tree, the Bluetooth LE Timer, the Clocked Wake-Up Controller, and the Quadrature Decoder. These blocks are supposed to trigger or to capture wake-up events while the system is in any of the clocked sleep modes.

The timers power domain (PD_TIM) contains special purpose timers that might or might not be crucial for an application: a full featured Real Time Clock (RTC) engine and Timer1. The registers of these blocks are 32-bit wide, therefore, they are connected to the APB32 bus.

The APB16 bus connects to the radio power domain (PD_RAD), which consists of the Radio control unit and the Bluetooth LE controller, and to the peripheral blocks which are all part of the same power domain as the CPU (PD_SYS).

9.8.3 Programming

Since the AMBA Bus only acknowledges a single master at a time, a programmable arbitration is implemented to decide whether the Arm Cortex-M0+ or the DMA is the master. The priority can be configured in the GP_CONTROL_REG[CPU_DMA_BUS_PRIO] with the CPU having the highest priority by default.

9.9 Memory Map

Table 71: Memory map

Address	Description	Power Domain
0x00000000 0x04000000	**Boot/Bluetooth LE ROM/OTP/RAM** Remapped address space based on SYS_CTRL_REG[REMAP_ADR0].	
0x04000000 0x07F00000	**RESERVED**	

Address	Description	Power Domain
0x07F00000 0x07F24000	**Boot/Bluetooth LE ROM** Contains Boot ROM code and Bluetooth LE protocol related code.	
0x07F24000 0x07F40000	**RESERVED**	
0x07F40000 0x07F40100	**OTP-Regs** Contains the control registers of the OTP Subsystem.	PD_SYS
0x07F40100 0x07F80000	**RESERVED**	
0x07F80000 0x07F88000	**OTP** Contains the OTP cell actual memory space.	
0x07F88000 0x07F C0000	**RESERVED**	
0x07FC0000 0x07FCC000	**System RAM** 48 kB. Contains application code, data for the application, stack, and heap. SysRAM1 (16 kB): 0x07FC0000 to 0x07FC3FFF SysRAM2 (12 kB): 0x07FC4000 to 0x07FC6FFF SysRAM3 (20 kB): 0x07FC7000 to 0x07FCBFFF	
0x07FD8000 0x40000000	**RESERVED**	
0x40000000 0x40001000	**AHB/Bluetooth LE-Regs** Contains the control registers of the Bluetooth LE Link Layer Processor.	PD_RAD
0x40001000 0x40004000	**AHB/Radio**	PD_RAD
0x40004000 0x50000000	**RESERVED**	
0x50000000 0x50000100	**APB16/PMU-CRG** Contains the control registers of the Power Management Unit and the Clock Generator.	PD_SLP
0x50000100 0x50000200	**APB16/Wake-Up** Contains the registers of the clocked and clock-less wake up controllers.	PD_SLP
0x50000200 0x50000300	**APB16/Quadrature Decoder** Contains Logic that implements a step counter for X and Y axis from a rotary encoder.	PD_SLP
0x50000300 0x50001000	**RESERVED**	
0x50001000 0x50001100	**APB16/UART** Contains the control registers of the UART.	PD_SYS
0x50001100 0x50001200	**APB16/UART2** Contain the control registers of the UART2.	PD_SYS
0x50001200 0x50001300	**APB16/SPI** Contains the control registers of the SPI interface.	PD_SYS
0x50001300 0x50001400	**APB16/I2C** Contains the control registers of the I2C interface.	PD_SYS
0x50001400 0x50001500	**APB16/Kbrd** Contains the registers of the Keyboard controller.	PD_SYS

Address	Description	Power Domain
0x50001500 0x50001600	**APB16/ADC** Contains the registers of the 4-channel ADC.	PD_SYS
0x50001600 0x50001700	**APB16/AnaMisc** Contains registers for various analog blocks.	PD_SYS
0x50001700 0x50003000	**RESERVED**	
0x50003000 0x50003100	**APB16/Ports** Contains the mode and direction registers of the GPIOs.	PD_SYS
0x50003100 0x50003200	**APB16/Watchdog** Contains the control registers of the Watchdog timer.	PD_SYS
0x50003200 0x50003300	**APB16/Version** Contains the version/revision of the chip.	PD_SYS
0x50003300 0x50003400	**APB16/Gen Purpose** Contains general purpose control registers.	PD_SYS
0x50003400 0x50003500	**APB16/Timer** Contains the control registers of Timer0 and Timer2.	PD_SYS
0x50003500 0x50003600	**APB16/RF Monitor** Contain the control registers of the RFMON.	PD_SYS
0x50003600 0x50003700	**APB16/DMA** Contains the control registers of the DMA.	PD_SYS
0x50003700 0x50004000	**RESERVED**	
0x50004000 0x50004100	**APB32/Timer1** Contains the control registers of Timer1.	PD_TIM
0x50004100 0x50004200	**APB32/RTC** Contains the control registers of the Real Time Clock.	PD_TIM
0x50004200 0xE0000000	**RESERVED**	
0xE0000000 0xE0100000	**Internal Private Bus** Contains various registers of the Arm Cortex-M0+.	PD_SYS

9.10 Memory Controller

9.10.1 Introduction

The Memory Controller of RA6W2 Bluetooth LE Subsystem is responsible for the interface between the memory cells and the masters of the system that request access. It comprises two arbiters which use a fixed priority level scheme to allow parallelization between the three main masters of RAM. The memory controller also provides the actual physical sequence of the RAM cells in a continuous memory space to enable the activation of the required amount of SysRAM only to save power.

The block diagram is presented in Figure 76.

Features

- Three different capacities for the RAM cells with retention capability (12 kB, 16 kB, and 20 kB).
- Arbitration among the AHB masters (CPU or DMAs), OTP, and the Bluetooth LE core.
- Transparently interfaces the AHB busses to memory signaling.
- Fixed arbitration algorithm with time sharing.

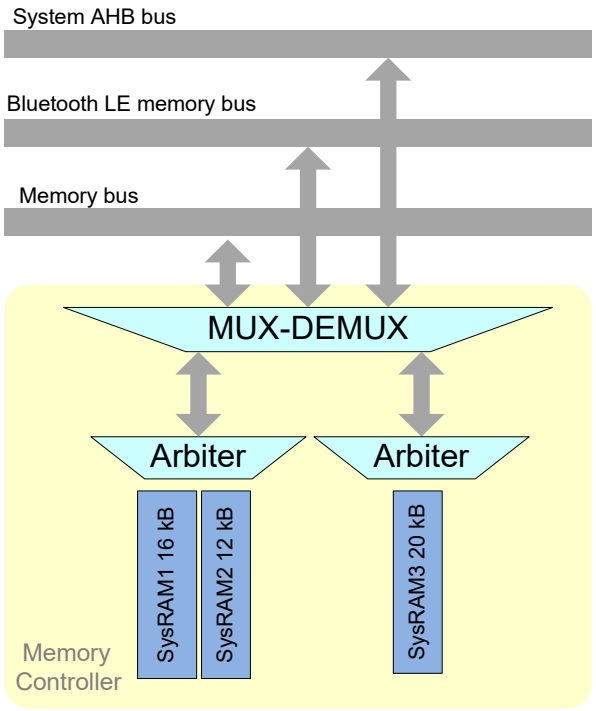


Figure 76. Memory controller block diagram

9.10.2 Architecture

The Memory Controller contains two Arbiters which connect to the following busses via a Mux-Demux:

- Bluetooth LE Mem I/F: this is a memory interface directly from the Bluetooth 5.1 Core to the RAM used as an exchange memory (TX/RX descriptors and others). This interface always operates at 16 MHz.
- System Mem I/F: this is a memory interface directly from the OTP memory to the RAM used for copying data after power-up/wake-up.

Arbitration

The arbitration is a mixture of the highest priority and a fair use policy. If more than one master request access to cells which reside under the same arbiter, time division is employed. This is to make sure none of the busses can stall the others for a long period. The OTP and Bluetooth LE accesses are handled as very critical and therefore they have the highest priority.

9.11 Clock Generation

9.11.1 Clock Tree

The generation of the system's clocks is described in detail in Figure 77.

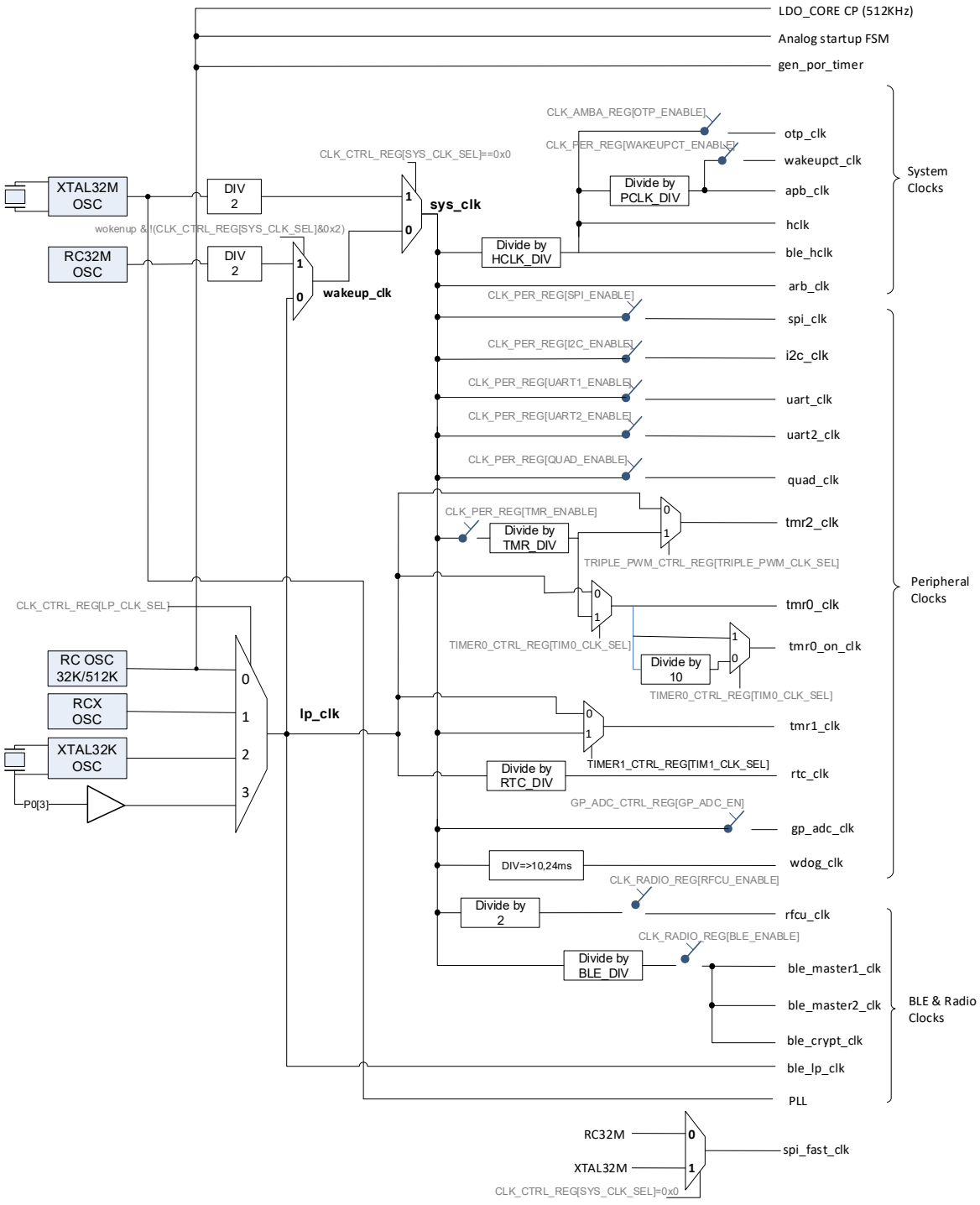


Figure 77. Clock tree diagram

Figure 77 shows the possible clock sources as well as all different divisions and multiplexing paths towards the generation of each block's clock. Furthermore, the required registers that must be programmed are also shown in Figure 77.

Internal clock sources of RA6W2 Bluetooth LE Subsystem are the RC32M, RC32K/512K, and RCX oscillators. External clock sources of RA6W2 Bluetooth LE Subsystem are the 32 MHz crystal oscillator (the pins

XTAL32Mp and XTAL32Mm), the 32.768 kHz crystal oscillator (the pins XTAL32kp and XTAL32km mapped on P0_3 and P0_4, respectively), or an external digital clock (the pin P0_3).

There are two main clock lines which are of interest:

- lp_clk: this is the low power clock used for the sleep modes and can only be either the RCX, the RC32K, the XTAL32K, or an externally supplied digital clock.
- sys_clk: this is the system clock used for the AMBA clock (hclk), which runs the CPU, the memory, and the bus. This clock source can be one of the oscillators or an externally supplied digital clock.

The clock names shown in Figure 77 are explained in Table 72.

Table 72: Generated clocks description

Clock name	Description
wakeupct_clk	Clocked wake-up controller clock.
apb_clk	AMBA APB interface clock.
otp_clk	OTP controller clock.
hclk	AMBA AHB interface clock.
ble_hclk	AMBA AHB clock for the Bluetooth LE core.
wdog_clk	Watchdog clock.
pmu_rc_clk	Clock for the PMU and analog start-up FSM.
icp_clk_512_c (512KHz)	Clock for the Charge pump in the LDO_CORE.
gen_por_timer	Clock for the POR_FORCE Timer.
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].
spi_fast_clk	Fast 32 MHz clock for the SPI controller.
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.
tmr1_clk	Timer1 clock.
tmr0_on_clk	Timer0 ON counter clock.
rtc_clk	Clock for Real Time Clock (RTC).
gp_adc_clk	General Purpose ADC conversion clock.
ble_crypt_clk	Clock for the Crypto block of the Bluetooth LE core.
ble_master1_clk	Internal clock for the Bluetooth LE core.
ble_master2_clk	Internal clock for the Bluetooth LE core.
arb_clk	Clock for the memory controller arbiter.
ble_lp_clk	Bluetooth LE core low power clock.
pll	PLL clock.

General Clock Constraints

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

- The minimum of the AMBA clock (hclk) should be 8 MHz when Bluetooth LE is utilized. This is also the clock of the Cortex CPU and ensures the required MIPS for handling the Bluetooth LE Protocol.

- The AMBA clock (hclk) should always be greater or equal to the ble_*_clks. This is required for the proper operation of the Bluetooth LE protocol. For example, hclk at 16 MHz and Bluetooth LE clocks at 8 MHz are an acceptable combination but not the other way around.

9.11.2 Crystal Oscillators

The Digital Controlled XTAL Oscillators (DXCO) are designed for low power consumption and high stability. There are two such crystal oscillators in the system, one at 32 MHz (XTAL32M) and the other at 32.768 kHz (XTAL32K). The XTAL32K has no trimming capabilities and is used as the clock of the Deep Sleep/Extended Sleep modes. The XTAL32M can be trimmed.

The principal schematic of the two oscillators is shown in Figure 78. No external components to the RA6W2 are required other than the crystal itself. If the crystal has a case connection, it is advised to connect the case to ground.

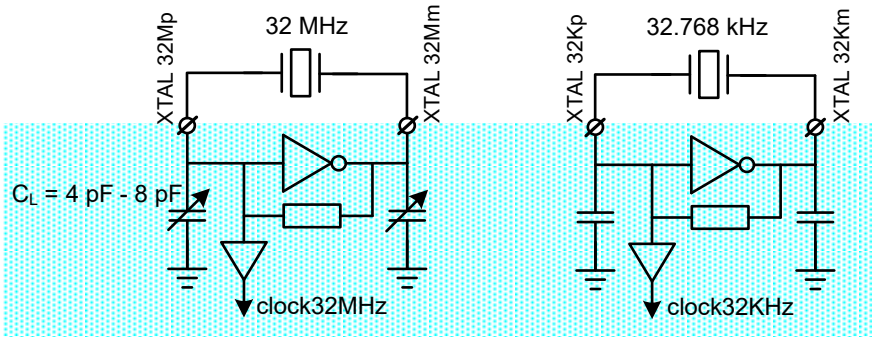


Figure 78. Crystal oscillator circuits

9.11.2.1 Frequency Control (32 MHz Crystal)

The 8-bit register CLK_FREQ_TRIM_REG controls the trimming of the 32 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 the minimum to the maximum value in 256 equal steps. With CLK_FREQ_TRIM_REG[XTAL32M_TRIM] = 0x00, the minimum capacitance and thus the maximum frequency are selected. With CLK_FREQ_TRIM_REG[XTAL32M_TRIM] = 0xFF, the maximum capacitance and thus the minimum frequency are selected.

The five least significant bits of CLK_FREQ_TRIM_REG register (XTAL32M_TRIM<4:0>) directly control five binary weighted capacitors (Figure 79). The three most significant bits of CLK_FREQ_TRIM_REG register (XTAL32M_TRIM<7:5>) are binary to the thermometer decoded. Each of the seven outputs of the decoder controls a capacitor, of which the value is 32 times the value of the smallest capacitor.

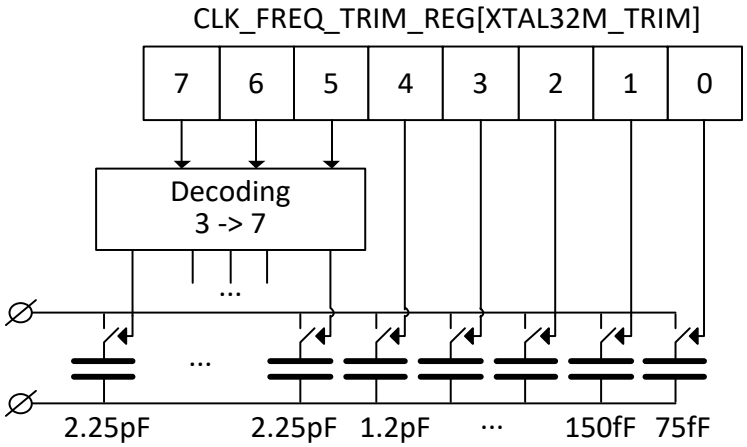


Figure 79. XTAL32 MHz oscillator frequency trimming

9.11.2.2 Automated Trimming and Settling Notification

There is provision in the RA6W2 for automating the actual trimming of the 32 MHz crystal oscillator. This is a special hardware block that realizes the XTAL trimming in a single step. Notification about the XTAL oscillator being settled after applying the trim value is also provided in form of an interrupt, namely, the XTAL32RDY_IRQn line. The automated mechanism for applying the trim value and signaling that the oscillator is settled is shown in Figure 80.

The XTAL32RDY_IRQn is always triggered as soon as an internal counter reaches the value programmed at XTALRDY_CTRL_REG. This counter runs on the RC32M clock if the system is powering up, or on a selected low power clock if the system is waking up. The enabling of the XTAL32M is always done by hardware. There are two sections until the interrupt notifies the software that the XTAL32M can be used:

- The start-up section, where the XTAL32M oscillator is slowly converging towards the initial frequency of the crystal. This section ends with the application of the trim value to achieve a <50 ppm, 32 MHz clock.
- The settling section, where the XTAL32M oscillator settles to the preferred frequency after the application of the trim value which is done automatically by hardware.

There are two ways of deciding when the start-up section ends and when the trim values are supposed to be applied. This decision is controlled by TRIM_CTRL_REG[XTAL_TRIM_SELECT] bit field:

- **Counter Mode**: trim value stored in the CLK_FREQ_REG is applied as soon as an internal counter reaches the value XTAL_COUNT_N-1. This is the default mode.
- **Current Mode**: trim value is applied as soon as the current drops.

The different modes are shown in Figure 80.

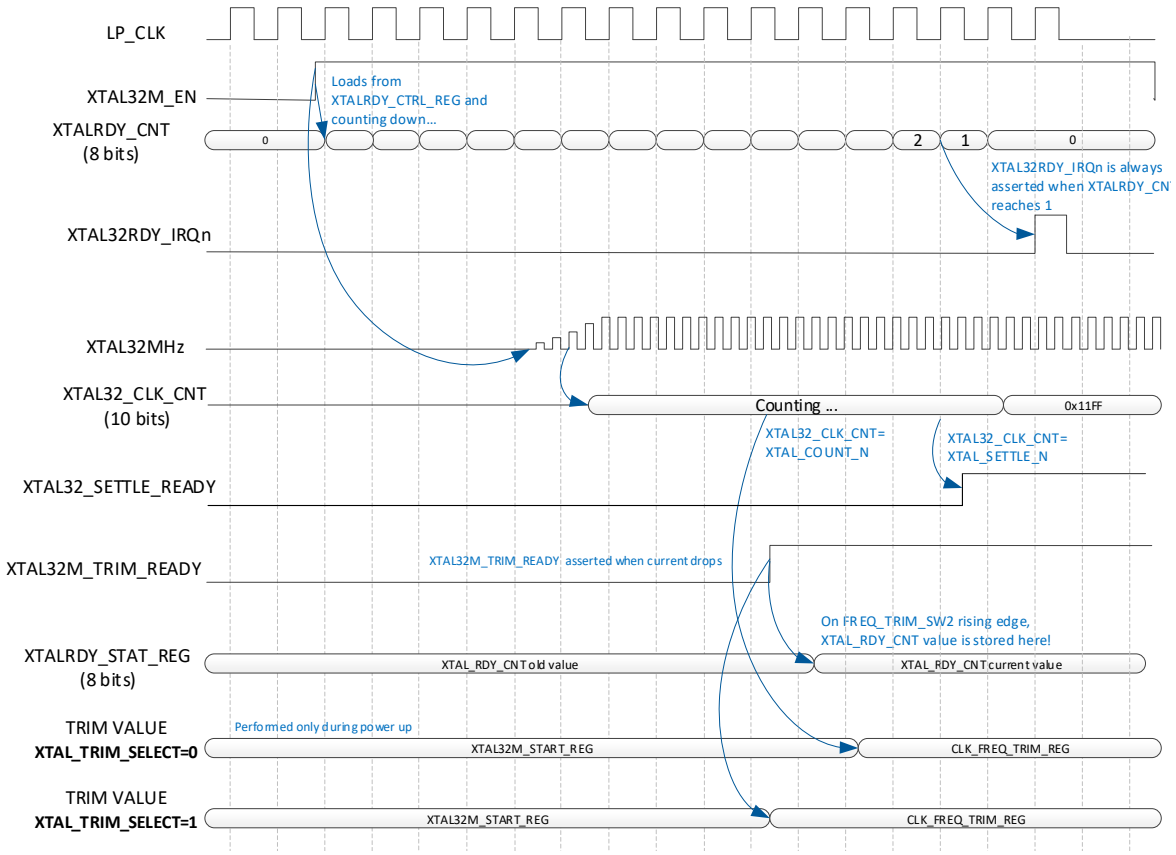


Figure 80. Automated mechanism for XTAL32M trim and settling

In both modes mentioned above, trimming is done by hardware. In the current mode, upon assertion of XTAL32M_TRIM_READY, the interrupt counter value is stored in a shadow register XTALRDY_STAT_REG to enable software understanding when the start-up section is finished.

The settling section usually takes no more than five to 10 clock cycles. Using the explanation above, fine tuning and reducing the XTAL32M latency is feasible. One feature of the XTAL32_CLK_CNT is that it asserts an observable signal (SYS_STAT_REG[XTAL32_SETTLE_READY]) as soon as the counter reaches a pre-defined

threshold programmed at TRIM_CTRL_REG[XTAL_SETTLE_N]. This allows the software to have an indication of the status of the clock by adjusting the threshold accordingly.

9.11.3 RC Oscillators

There are three RC oscillators in the RA6W2:

- One providing 32 MHz (RC32M)
- One providing 32 kHz and 512 kHz (RC32K/512K)
- One providing a frequency of 15 kHz (RCX).

The RC32M is powered by V_{BAT_LOW} which is available during Active or Sleep mode. The output clock is slower than 32 MHz if untrimmed and it is used to clock the CPU and the digital part of the chip during power-up or wake-up, while the XTAL32M oscillator is settling.

The simple RC oscillator (RC32K/512K) operates on VDD and provides 32 kHz or 512 kHz. The main usage of the RC32K/512K oscillator is for internal clocking during power-up or start-up. It clocks the hardware FSM which brings up the power management system of the chip. In the power-up or start-up sequence, the clock dynamically changes from 32 kHz to 512 kHz to speed up the sequence. The enhanced RC oscillator (RCX) provides a stable 15 kHz frequency and operates on V_{BAT_LOW} which is available during Active or Sleep mode.

The RCX oscillator can be used to replace the 32.768 kHz crystal, since it has a precision of < 500 ppm, while its output frequency needs to be recalibrated over temperature.

Using the RCX requires the following registers to be set:

- Set GP_DATA_REG = 0x20 after the system wakes up
- RCX calibration (the calibration is optional)
- Go to sleep: set GP_DATA_REG = 0x40 after the sleep procedure is handled.

The procedure is also implemented as a part of the SDK.

Frequency Calibration

The output frequency of the 32 kHz crystal oscillator and the three RC-oscillators can be measured relative to the 32/2 (16) MHz crystal oscillator using the on-chip reference counter.

The measurement procedure is as follows:

1. REF_CNT_VAL = N (the larger number N is, the more accurate and longer the calibration is).
2. CLK_REF_SEL_REG = 0x0000 (RC32K) or
   CLK_REF_SEL_REG = 0x0001 (RC32M) or
   CLK_REF_SEL_REG = 0x0002 (XTAL32K) or
   CLK_REF_SEL_REG = 0x0003 (RCX)
3. Start the calibration: CLK_REF_SEL_REG[REF_CAL_START] = 1.
4. Wait until CLK_REF_SEL_REG[REF_CAL_START] = 0.
5. Read CLK_REF_VAL_H_REG and CLK_REF_VAL_L_REG = M (32-bit values).
6. Frequency = (N/M) × 32/2 MHz.

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

9.12 OTP Controller

9.12.1 Introduction

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), comprises a DMA engine which connects to the AHB bus as a master, and has the highest priority to copy code from OTP into SysRAM in mirrored mode. Figure 81 shows the block diagram.

Features

- Implements all timing constraints for any access to the physical OTP cell
- Automatic single Error Code Correction (ECC) - 6 bits (implemented in the OTP cell)
- 32-bit read in a single read access from the OTP cell
- Single word buffer for programming. No burst programming supported
- Empty words are 0xFFFFFFFF. Zeros are programmed per 32-bit word
- Embedded DMA engine for fast mirroring of the OTP contents into the SysRAM
- Embedded DMA supports reading in bursts of 4 × 32-bit words
- Transparent random address access to the OTP memory cells via the AHB slave memory interface
- Hardwired handshaking with the PMU to realize the mirroring procedure.

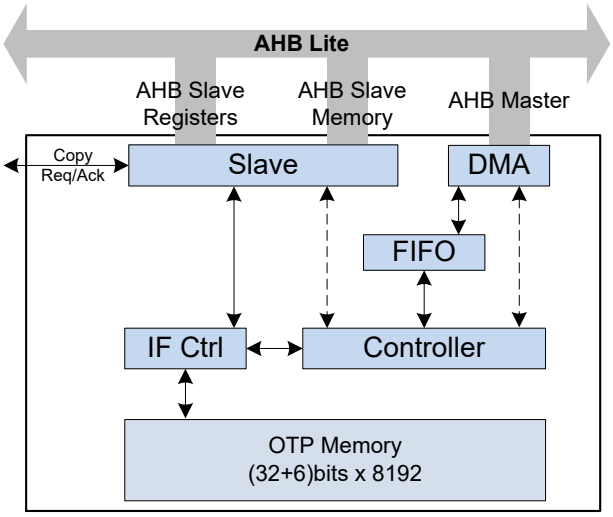


Figure 81. OTP controller block diagram

9.12.2 Architecture

The OTP controller block includes the OTP macro cell and pure digital logic implementing the controlling functions. The OTP memory communicates with the controller through a proprietary interface.

The internal organization of the OTP cell is 32-bit data and 6-bit ECC for each of the 8192 addressable positions. The six bits of the ECC are only accessible within the OTP cell. The ECC is generated by the OTP cell during the programming and is used again by the OTP cell in a transparent way during reading.

The AHB master interface is controlled by a DMA engine with an internal FIFO of eight 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 the 32-bit words of a transfer minus 1 must be specified in OTPC_NWORDS_REG[OTP_NWORDS].

The DMA engine internally supports the following burst types:

- Eight words incremental burst (INCR8)
- Four words incremental burst (INCR4)
- Unspecified incremental burst (INCR) with a length different from 1, 4, or 8
- Single word access (SINGLE).

The slave block combines two AHB slave interfaces: one is for the registers and can be read from/written to, and the other is for the contents of the OTP memory and is read-only.

The OTP controller configures the OTP cell to be in one of the following modes:

- **Deep Stand-by Mode (DSTBY)**. In this mode, the required power supplies are applied to the OTP cell, while the internal LDO of the OTP cell is inactive.
- **Stand-by Mode (STBY)**. In this mode, the OTP cell is disabled by deactivating the chip select signal. The OTP cell is powered and the internal LDO is enabled. The power consumption of the OTP cell in this mode is not the minimum possible but is less than in an active mode (RD, PROG, PVFY, RINI, AREAD). This is the state from which any active mode of operation can be transitioned into with the least delay.
- **Read Mode (RD)**. When this mode is used, the contents of the OTP cell can be read at the respective AHB address space. This mode can also be used to execute software in place (XIP). A read request is translated by the OTP controller into the corresponding control sequence for the OTP cell to retrieve the requested data.
- **Programming Mode (PROG)**. The PROG mode provides the functionality to program a 32-bit word into an OTP position. The OTP cell expands the 32-bit word by calculating and automatically appending a 6-bit checksum (ECC). Please note that there is no way to access these extra six bits of the ECC information. Programming is performed only for bits equal to 0. Bits equal to 1 are bypassed to save programming time. Because the ECC value is unknown to the controller, there are always six extra programming pulses applied to the ECC bits. Programming is done by issuing a programming request stored in the Programming Buffer (PBUF). PBUF consists of two configuration registers storing the 32-bit data value and the 13-bit address in the OTP cell where the value should be programmed. A new request can only be stored in PBUF when the previous one is served. A status bit indicates whether this has already been done, therefore programming should be monitored by software before a new programming request is issued.
- **Programming Verification Mode (PVFY)**. The PVFY mode forces the OTP cell to enter a special margin read mode. This mode is used to verify the content of the OTP positions that have been programmed using the PROG mode and to verify that the programmed data is retrieved correctly under all corner cases. When this mode is used, the contents of the OTP cell can be read at the respective AHB address space. The CPU must read all OTP positions that have been programmed by accessing the corresponding addresses and verify that all the retrieved words are equal to the expected values.
- **Read Initial State Mode (RINI)**. The RINI mode implements a production test of the initial margin read, which should be performed in the OTP cell, before the first programming is applied. This test verifies that the OTP cell is empty (all the bits are equal to 1). The OTP controller sends the required control sequence to the OTP cell to enable the test mode. Then the CPU should read all the content of the OTP cell at the respective AHB address space and verify that all the retrieved words are equal to 0xFFFFFFFF. The RINI mode should be used after the PROG mode to verify the content of the OTP positions that have been programmed and specify the bits that remain un-programmed. This verification is required to ensure that the programming process has not affected the un-programmed bits. This specific read mode is a margin read, which means that it is not an equivalent to the normal read and should only be used for verification.
- **Automatic Read Mode (AREAD)**. This mode is used to mirror large parts of the OTP cell into RAM through the AHB master interface and the integrated DMA controller.

Transitioning from one mode to another automatically steps through the STBY mode.

OTP Accessing Considerations

Access to the OTP memory (read/write) can only be performed at certain voltage ranges. You are responsible for meeting these conditions while accessing the OTP. The recommended operation conditions of the OTP memory can be found under Recommended Operating Conditions section.

9.12.3 Programming

To configure the OTP controller:

1. Enable clock for the OTP controller by setting the CLK_AMBA_REG[OTP_ENABLE] bit.
2. Put the OTP in STBY mode (OTPC_MODE_REG[OTPC_MODE_MODE] = 0x1).
3. Wait for OTP mode to change (OTPC_STAT_REG[OTPC_STAT_MRDY] = 1).
4. Set OTP speed by writing OTPC_TIM1_REG and OTPC_TIM2_REG if the system clock speed is to be reduced. These numbers basically generate asynchronous timing signals towards the OTP cell that comply to the default internal 16 MHz bus speed.
5. Perform an OTP access:

a. Programming:
   i. Set up OTP write mode (OTPC_MODE_REG[OTPC_MODE_MODE] = 0x3).
   ii. Wait for OTP mode to change (OTPC_STAT_REG[OTPC_STAT_MRDY] = 1).
   iii. Check OTPC_STAT_REG[OTPC_STAT_PBUF_EMPTY] = 1
   iv. Write the data to be programmed to OTPC_PWORD_REG.
   v. Write the address to which the data is to be programmed to OTPC_PADDR_REG.
   vi. Wait until the programming is finished (OTPC_STAT_REG[OTPC_STAT_PRDY] = 1).
   vii. Switch to OTP verify mode (OTPC_MODE_REG[OTPC_MODE_MODE] = 0x4).
   viii. Wait for OTP mode to change (OTPC_STAT_REG[OTPC_STAT_MRDY] = 1).
   ix. Read back and compare the data written.
   x. Put the OTP in STBY mode (OTPC_MODE_REG[OTPC_MODE_MODE] = 0x1).
   xi. Wait for OTP mode to change (OTPC_STAT_REG[OTPC_STAT_MRDY] = 1).
b. Reading:
   i. Set up OTP read mode (OTPC_MODE_REG[OTPC_MODE_MODE] = 0x2).
   ii. Wait for OTP mode to change (OTPC_STAT_REG[OTPC_STAT_MRDY] = 1).
   iii. Read the OTP word.
   iv. Put the OTP in STBY mode (OTPC_MODE_REG[OTPC_MODE_MODE] = 0x1).
   v. Wait for OTP mode to change (OTPC_STAT_REG[OTPC_STAT_MRDY] = 1).

9.13 DMA Controller

9.13.1 Introduction

The 4-channel direct memory access (DMA) controller transfers data of eight bits, 16 bits, or 32 bits between the on-chip supported peripherals (SPI, UART, UART2, I2C, and ADC) and the on-chip RAM and supports regular memory-to-memory transfers. The DMA also supports a programmable interrupt generation to generate an interrupt after a certain number of transfers to off load the Cortex interrupt rate. The on-chip peripheral requests are multiplexed on the two available channel pairs to increase the DMA utilization. A block diagram of the controller is shown in Figure 78.

Features

- Four channels with an optional peripheral trigger
- Full 32-bit source and destination pointers
- Flexible interrupt generation
- Programmable length
- Flexible peripheral request per channel
- Option to initialize memory (DMA_INIT)
- Programmable edge-sensitive request support (recommended when writing to UART/UART2 and I2C).

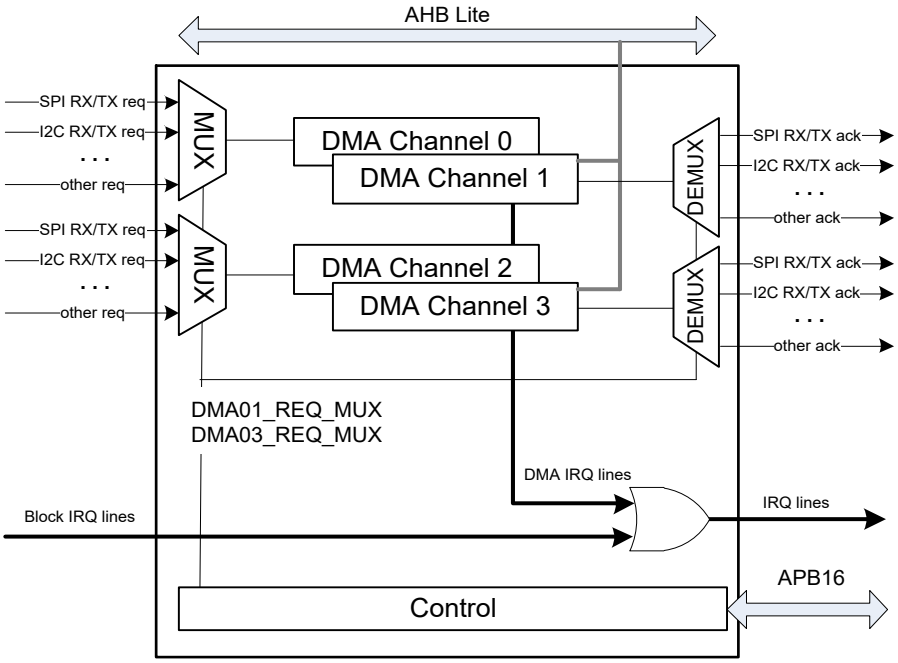


Figure 82. DMA controller block diagram

9.13.1.1 Architecture

9.13.1.2 DMA Peripherals

By default, the DMA assumes memory-to-memory transactions. Each DMA channel can also be connected with the hand-shaking signals or other request signals of the corresponding peripherals (Table 73).

Table 73: DMA served peripherals

Name	Direction
SPI	RX/TX
UART	RX/TX
UART2	RX/TX
I2C	RX/TX

Name	Direction
GP-ADC	RX

9.13.1.3 Input/Output Multiplexer

The multiplexing of peripheral requests is controlled by DMA_REQ_MUX_REG. Thus, if DMA_REQ_MUX_REG[DMAxy_SEL] is set to a certain (non-reserved) value, the TX/RX request from the corresponding peripheral is routed to DMA channels y (TX request) and x (RX request), respectively. Similarly, an acknowledging de-multiplexing mechanism is applied.

When two or more bit-fields (peripheral selectors) of DMA_REQ_MUX_REG have the same value, the lesser significant selector is given priority (see also the register's description).

9.13.1.4 DMA Channel Operation

A DMA channel is switched on with bit DMA_ON. This bit is automatically reset if the DMA channel's transfer is finished. The DMA channels can either be triggered by software or by a peripheral DMA request. If DREQ_MODE is 0, a DMA channel is immediately triggered.

If DREQ_MODE is 1, a DMA channel can be triggered by a hardware request coming from a selected peripheral. All DMA channels support either level (default) or edge-sensitive requests via the bit-field REQ_SENSE of DMAx_CTRL_REG (x = 0, 1, 2, 3). If this bit-field is set (recommended for Memory-to-UART/UART2 and Memory-to-I2C transfers), the channel detects a positive edge on the request signal of the selected peripheral to start up a new transfer cycle. The edge-sensitive requests can be used globally, if desired, for all the peripherals interfacing with the DMA.

When DMA starts, data is transferred from address DMAx_A_START_REG to address DMAx_B_START_REG for a length of DMAx_LEN_REG, which can be eight, 16, or 32 bits wide. The address increment is realized with an internal 16-bit counter DMAx_IDX_REG, which is set to 0 when the DMA transfer starts and is compared with the DMAx_LEN_REG after each transfer. The register value is multiplied by the values of the automatic increment of source address (AINC), the automatic increment of destination address (BINC), and bus transfer width (BW) before it is added to DMAx_A_START_REG and DMAx_B_START_REG. AINC or BINC must be 0 for register access.

If at the end of a DMA cycle, the DMA start condition is still true, the DMA continues. The DMA stops if DREQ_MODE is low or if DMAx_LEN_REG is equal to the internal index register. This condition also clears the DMA_ON bit if DREQ_MODE is 0 or if DREQ_MODE is set to 1 and CIRCULAR bit is not set.

If a hand shaking is attached to the specific DMA channel at the end of a DMA cycle, the channel is blocked for as long as the peripheral is not ready for the next transaction.

If the bit CIRCULAR is set to 1, the DMA controller automatically resets the internal index registers and continues from its starting address without intervention of the Arm Cortex-M0+. If the DMA controller is started with DREQ_MODE = 0, the DMA always stops, regardless of the state of CIRCULAR.

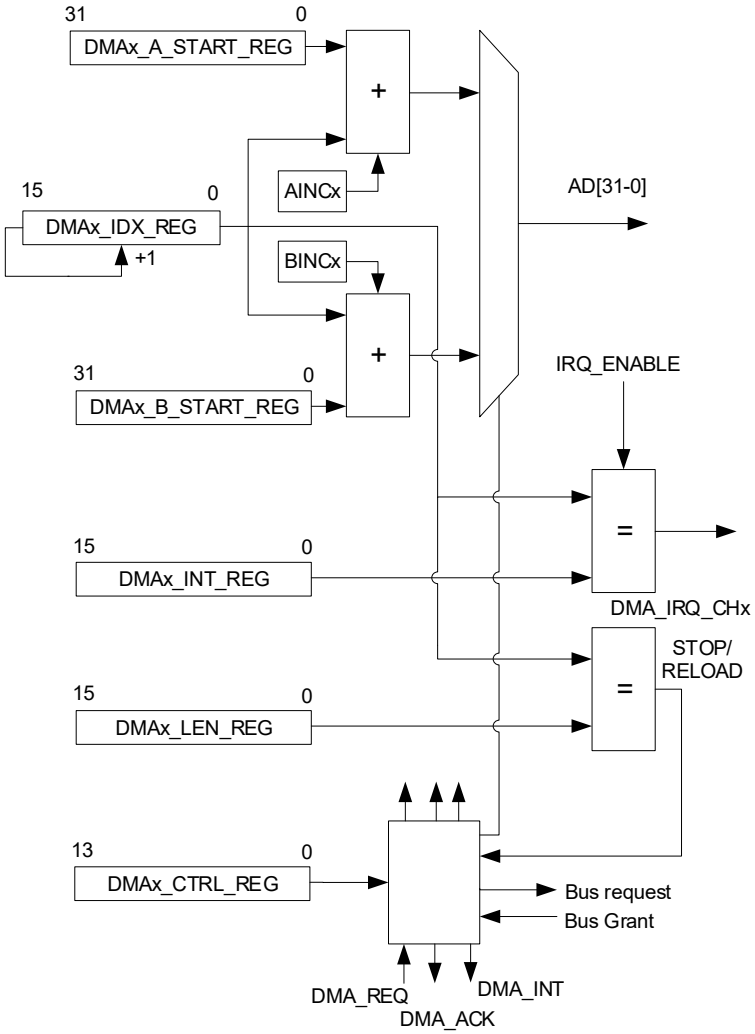


Figure 83. DMA channel diagram

Each DMA channel can generate an interrupt if the index counter DMAx_IDX_REG reaches the value of the channel's interrupt transfer length register, DMAx_INT_REG. After the transfer and before DMAx_IDX_REG is incremented, the interrupt is generated.

For example, if DMA_x_INT_REG = 0 and DMA_x_LEN_REG = 0, there is one transfer and an interrupt.

9.13.1.5 DMA Arbitration

The priority level of a DMA channel can be set with bits DMA_PRIO[2-0]. These bits determine which DMA channel is activated if more than one DMA channel requests DMA. If two or more channels have the same priority, an inherent priority applies (see register description).

With DREQ_MODE = 0, a DMA can be interrupted by a channel with a higher priority if the DMA_IDLE bit is set.

When DMA_INIT is set, however, the DMA channel currently performing the transfer locks the bus and cannot be interrupted by any other channels until the transfer is completed, regardless of whether DMA_IDLE is set. The purpose of DMA_INIT is to initialize a specific memory block with a certain value without any interruption from other active DMA channels that may request the bus at the same time. Consequently, DMA_INIT should be used only for memory initialization. When the DMA transfers data to/from peripherals, DMA_INIT should be set to 0.

NOTE
When DMA_INIT is enabled, AINC must be set to 0 and BINC to 1. Memory initialization could also be performed by simply setting AINC to 0 and BINC to 1 without enabling the DMA_INIT, provided that the source address of the memory does not change during the transfer. However, it is not guaranteed that

NOTE
the DMA transfer is not interrupted by other channels of a higher priority, when they request access to the bus at the same time.

9.13.1.6 Freezing DMA Channels

Each channel of the DMA controller can be temporarily disabled by writing a 1 to bit 4 SET_FREEZE_REG[FRZ_DMA] to freeze all channels.

To enable a frozen channel again, write a 1 to bit 4 RESET_FREEZE_REG[FRZ_DMA].

There is no hardware protection from erroneous programming of the DMA registers.

The on-going Memory-to-Memory transfers (DREQ_MODE = 0) cannot be interrupted, therefore the corresponding DMA channels are frozen after a Memory-to-Memory transfer is completed.

9.13.2 Programming

9.13.2.1 Memory to Memory Transfers

1. Set the length of data to be transferred (DMAx_LEN_REG).
2. Set the source address (DMAx_A_START_REG).
3. Set the destination address (DMAx_B_START_REG).
4. Configure the number of transfers until an interrupt is generated (DMAx_INT_REG).
5. Configure transfer options:
   a. DMAx_CTRL_REG[AINC]: Automatic increment of source address.
   b. DMAx_CTRL_REG[BINC]: Automatic increment of destination address.
   c. DMAx_CTRL_REG[BW]: Bus transfer width.
   d. DMAx_CTRL_REG[IRQ_ENABLE]: Enable the DMA interrupt generation for this channel.
6. Start the DMA transfer by setting the DMAx_CTRL_REG[DMA_ON] bit.
7. Wait until the transfer is finished (DMAx_CTRL_REG[DMA_ON] = 0).
8. Clear the IRQ status bit for channel x in DMA_INT_STATUS_REG.

9.13.2.2 Peripheral to Memory Transfers

1. Set the length of data to be transferred (DMAx_LEN_REG).
2. Set the source address (DMAx_A_START_REG) to the peripheral Rx register (for example, I2C_DATA_CMD_REG).
3. Set the destination address (DMAx_B_START_REG). This should point to a buffer in memory (for example, SYSRAM).
4. Configure the number of transfers until an interrupt is generated (DMAx_INT_REG).
5. Map the peripheral to the selected channels pair (DMA_REQ_MUX_REG[DMAxy_SEL]).
6. Configure transfer options:
   a. DMAx_CTRL_REG[AINC]: Disable automatic increment of source address.
   b. DMAx_CTRL_REG[BINC]: Automatic increment of destination address.
   c. DMAx_CTRL_REG[BW]: Bus transfer width.
   d. DMAx_CTRL_REG[DREQ_MODE]: Enable triggering by peripheral DMA request.
   e. DMAx_CTRL_REG[DMA_PRIO]: Set the channel's priority.
   f. DMAx_CTRL_REG[IRQ_ENABLE]: Enable the DMA interrupt generation for this channel.
   g. DMAx_CTRL_REG[REQ_SENSE]: Enable edge-sensitive requests for this channel. This is recommended for Memory-to-UART/UART2/I2C transfers but can also be used globally for all the supported peripherals and for both directions (TX/RX).
7. Start the DMA transfer by setting the DMAx_CTRL_REG[DMA_ON] bit.
8. Enable peripheral’s DMA request (for example, I2C_DMA_CR_REG[TDMAE]).

9. Clear the IRQ status bit for channel x in DMA_INT_STATUS_REG.

9.14 I2C Interface

9.14.1 Introduction

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, and A/D and D/A converters.

Features

- Two-wire I2C serial interface consisting 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 locations deep transmit/receive FIFOs (32× 8-bit Rx and 32× 10-bit Tx)
- Master transmit and Master receive operation
- Slave transmit and Slave receive operation
- 7-bit or 10-bit addressing
- 7-bit 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
- DMA support.

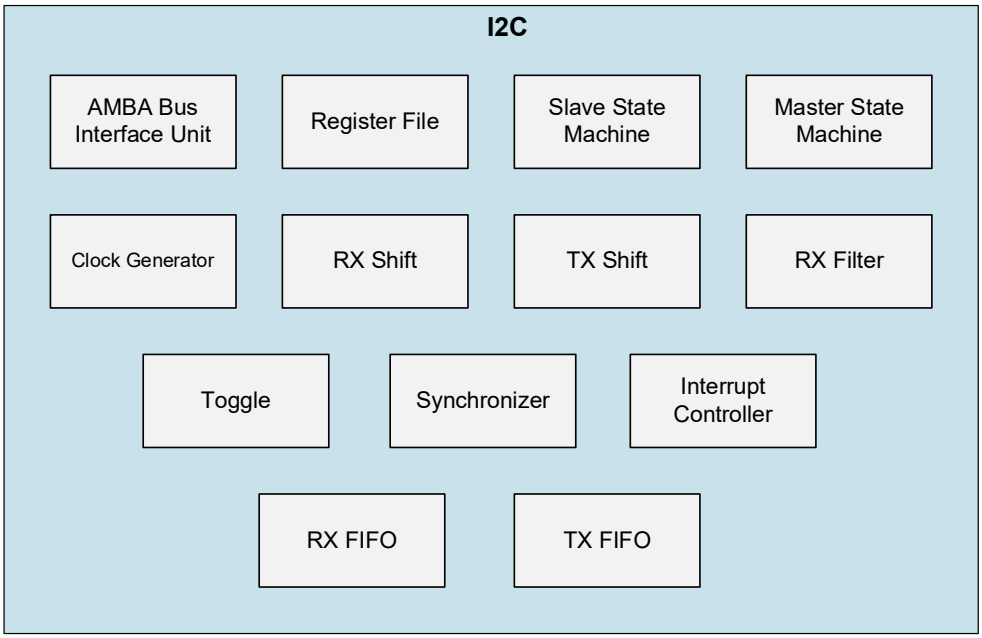


Figure 84. I2C Controller block diagram

9.14.2 Architecture

The I2C Controller block diagram is shown in Figure 84. It contains the following subblocks:

- **AMBA Bus Interface Unit**: it accesses the register file via the APB interface
- **Register File**: it contains configuration registers and the interface with software

- **Master State Machine**: it generates the I2C protocol for the master transfers
- **Slave State Machine**: it generates the I2C protocol for the slave transfers
- **Clock Generator**: it calculates the required time to do the following:
  - Generate the SCL clock when configured as a master
  - Check for bus idle
  - Generate a START and a STOP
  - Set up the data and hold the data
- **RX Shift**: it takes data into the design and extracts it in byte format
- **TX Shift**: it presents data supplied by CPU for transfer on the I2C bus
- **RX Filter**: it detects the events in the bus, for example, start, stop, and arbitration lost
- **Toggle**: it generates pulses on both sides and toggles to transfer signals across clock domains
- **Synchronizer**: it transfers signals from one clock domain to another
- **Interrupt Controller**: it generates the raw interrupt and interrupt flags, allowing them to be set and cleared.
- **RX FIFO/TX FIFO**: it holds the RX FIFO and TX FIFO register banks and controllers along with their status levels.

9.14.2.1 I2C Bus Terms

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

- **Transmitter** is the device that sends data to the bus. A transmitter can either initiate the data transmission to the bus (a master-transmitter) or respond to a request from the master to send data back (a slave-transmitter).
- **Receiver** is the device that receives data from the bus. A receiver can either receive data on its own request (a master-receiver) or respond to a request from the master to receive data (a slave-receiver).
- **Master** is 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** is the device addressed by the master. A slave can be either a receiver or a transmitter.

These concepts are shown in Figure 85.

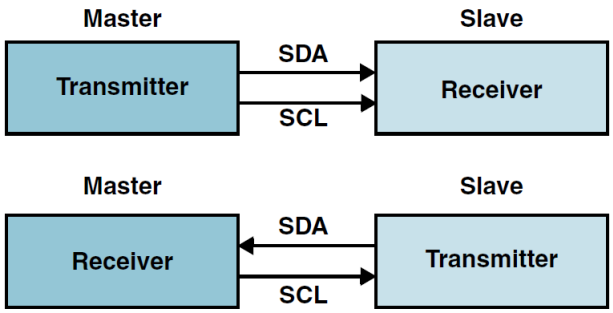


Figure 85. Master/Slave and Transmitter/Receiver relationships

- Multi-master means the ability for more than one master to co-exist on the bus at the same time without collision or data loss.
- Arbitration is the predefined procedure that authorizes only one master at a time to take control of the bus. For more information, see Section 9.14.2.4.
- Synchronization is the predefined procedure that synchronizes the clock signals provided by two or more masters. For more information, see Section 9.14.2.5.
- SDA is the data signal line (Serial Data).
- SCL is the clock signal line (Serial Clock).

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 of the SDA data line changes from low to high, while the SCL clock line remains high. When the data transfer has been terminated, the bus is free or idle again. The bus stays busy if a RESTART is generated instead of a STOP condition.

NOTE
START and RESTART conditions are functionally identical.

9.14.2.2 I2C Behavior

The I2C can only be controlled via software to be an I2C master, communicating with other I2C slaves. The master is responsible for generating the clock and controlling the transfer of data. The I2C protocol also allows multiple masters to reside on the I2C bus and uses an arbitration procedure to determine the bus ownership. A slave is responsible for either transmitting or receiving data to/from the master. The acknowledgment of data is sent by the device that is receiving data, which can be either a master or a slave.

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 whether the master wants to transmit data or receive data from the slave. The slave then sends an acknowledge pulse (ACK) after the address.

If a master-transmitter writes to a slave-receiver, the receiver gets one byte of data. This transaction continues until the master terminates the transmission with a STOP condition. If a master-receiver reads from a slave-transmitter, the slave transmits a byte of data to the master, and the master then acknowledges the transaction with an 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 shown in Figure 86.

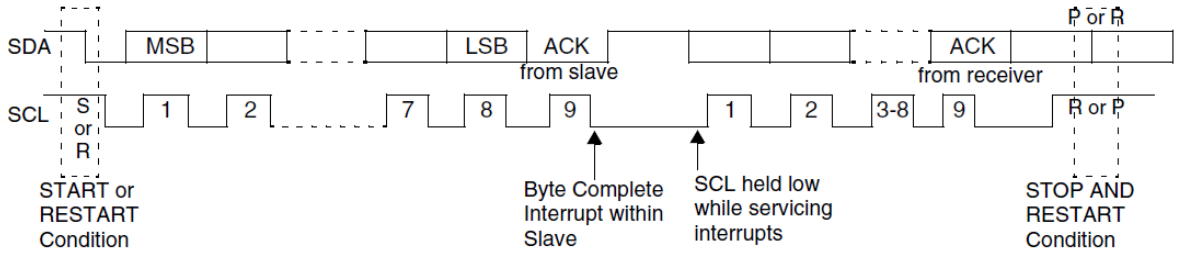


Figure 86. Data transfer on the I2C bus

The I2C is a synchronous serial interface. The SDA line is a bidirectional signal and changes only when 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 only by the maximum capacitance specification of 400 pF. Data is transmitted in byte packages.

9.14.2.2.1 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 or 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.

9.14.2.2.2 Combined Formats

The I2C Controller supports transactions in a read and write combined format 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.

To initiate combined format transfers, I2C_CON_REG[I2C_RESTART_EN] should be set to 1. With this value set and the I2C Controller operating as a master, when an I2C transfer is completed, the I2C Controller checks the transmit FIFO and executes the next transfer. If the direction of the new transfer differs from the previous one, 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 after a START condition.

9.14.2.3 I2C Protocols

The I2C Controller has the following protocols:

- START and STOP Conditions
- Addressing Slave Protocol
- Transmitting and Receiving Protocols
- START Byte Transfer Protocol.

9.14.2.3.1 START and STOP Conditions

When the bus is idle, both SCL and SDA signals are pulled high through external pull-up resistors on the bus. When a master wants to start a transmission on the bus, it 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, it issues a STOP condition. This is defined to be a low-to-high transition of the SDA line while SCL is 1. Figure 87 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.

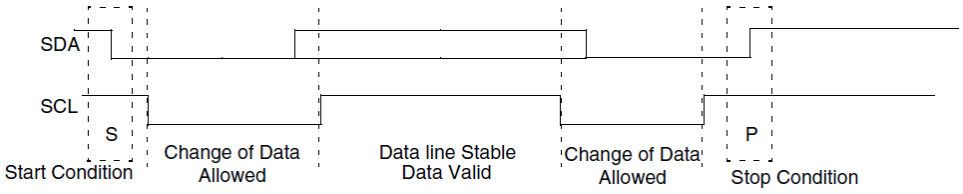


Figure 87. START and STOP conditions

NOTE
The signal transitions for the START/STOP conditions (Figure 87) reflect those observed at the output signals of the master driving the I2C bus. Be careful with 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.

9.14.2.3.2 Addressing Slave Protocol

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

9.14.2.3.3 7-bit Address Format

In 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 (Figure 88). 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.

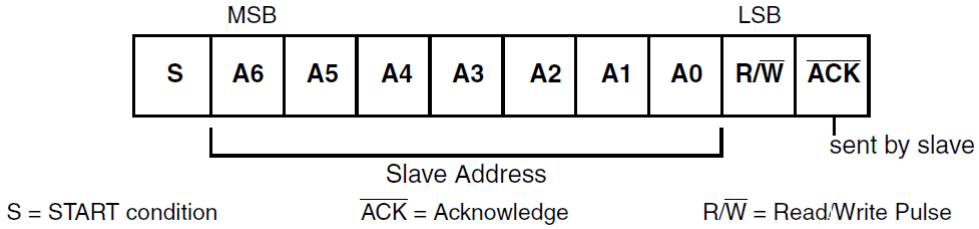


Figure 88. 7-bit address format

9.14.2.3.4 10-bit Address Format

In the 10-bit address format, 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, the next two bits (bits 2:1) 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 89 shows the 10-bit address format and Table 74 defines the special purpose and the reserved first byte addresses.

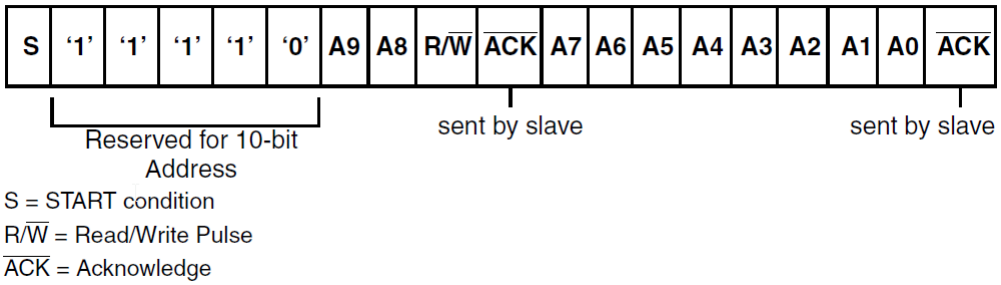


Figure 89. 10-bit address format

Table 74: I2C definition of bits in first byte in 10-bit address format

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, see "START Byte Transfer Protocol".
0000 001	X	CBUS address. I2C Controller ignores these accesses.
0000 010	X	Reserved.
0000 011	X	Reserved.
0000 1XX	X	Reserved
1111 1XX	X	Reserved.
1111 0XX	X	10-bit slave addressing.

The I2C Controller does not restrict you from using these reserved addresses. However, if these reserved addresses are used, incompatibility with other I2C components may occur.

9.14.2.3.5 Transmitting and Receiving Protocols

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

9.14.2.3.6 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 a master sends a slave address and an R/W bit or a master transmits a byte of data to a slave, the slave-receiver must respond with an acknowledge signal (ACK). When a slave-receiver does not respond with an ACK signal, 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 a master-transmitter is transmitting data as shown in Figure 90, the slave-receiver responds to the master-transmitter with an ACK signal after every byte of data is received.

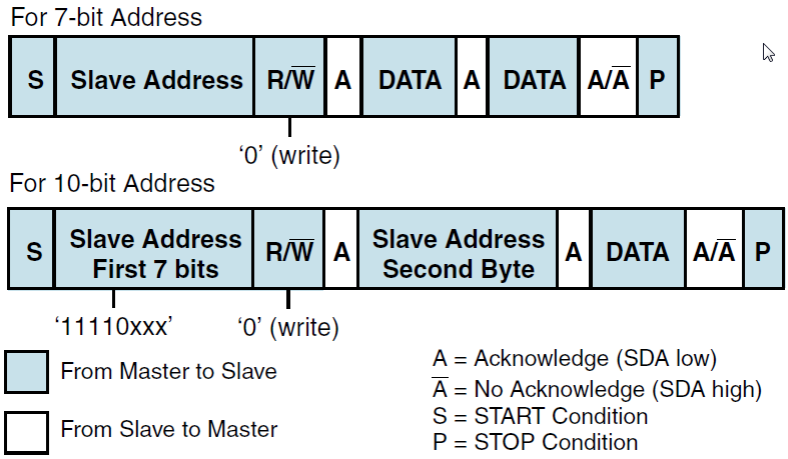


Figure 90. Master-transmitter protocol

9.14.2.3.7 Master-Receiver and Slave-Transmitter

If a master is receiving data as shown in Figure 91, the master responds to the slave-transmitter with an ACK signal 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 signal. The master can then communicate with the same slave or a different slave.

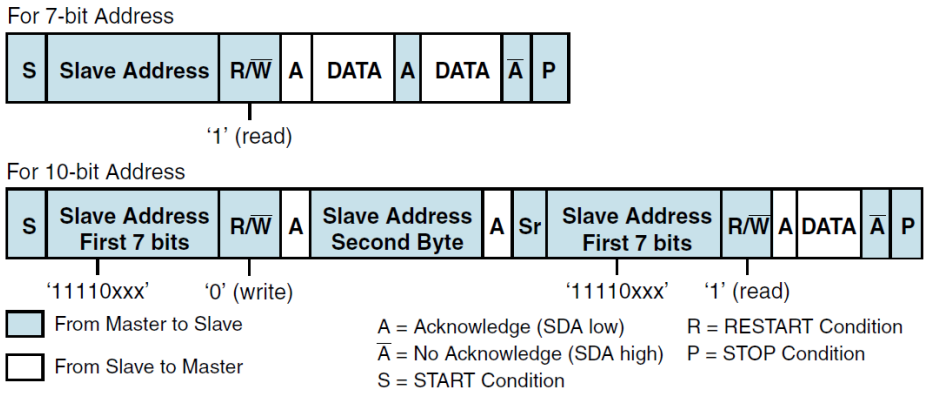


Figure 91. Master-receiver protocol

9.14.2.3.8 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 followed by a 1 being transmitted (Figure 92). 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.

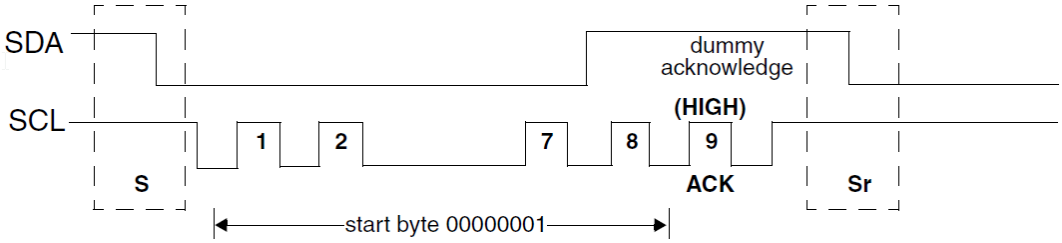


Figure 92. START byte transfer

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 gets reset after the RESTART condition is generated.

9.14.2.4 Multiple Master Arbitration

The I2C Controller allows multiple masters to reside on the same bus. There is an arbitration procedure if two masters on the same I2C bus try to control the bus at the same time by generating a START condition simultaneously. 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 a 0 loses the arbitration and turns off its data output stage. The master that has lost the 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 93 illustrates the timing of an arbitration between two masters on the bus.

Control of the bus is determined by the address or master code and data sent by the competing masters, so there is no central master or 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.

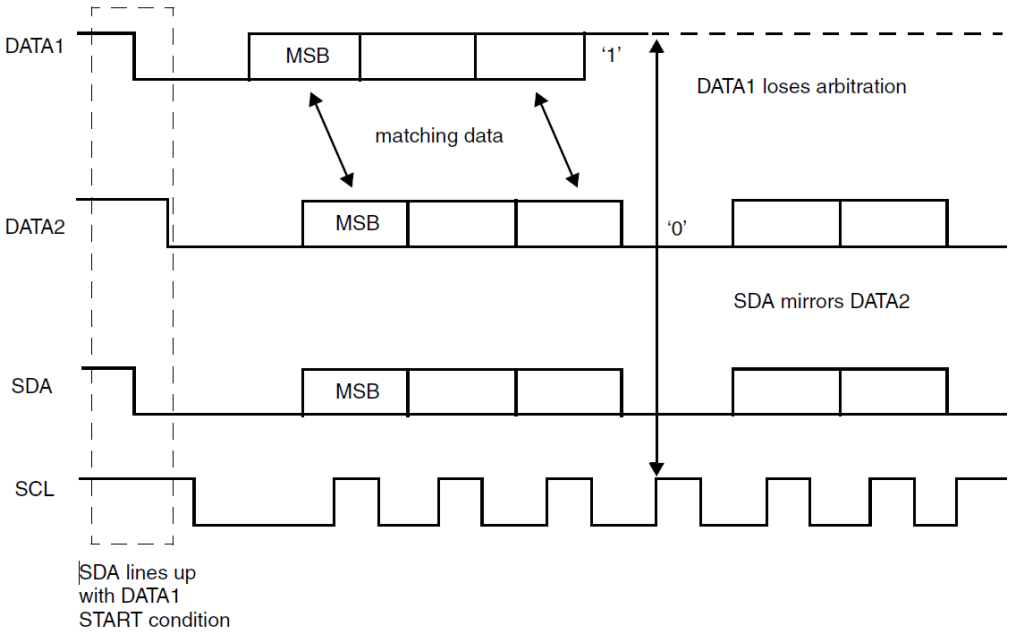


Figure 93. Multiple master arbitration

9.14.2.5 Clock Synchronization

All masters generate their own clock to transfer messages. Data is only valid during the HIGH period of the SCL clock. When two or more masters try to transfer information on the bus at the same time, they must synchronize the 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 period 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, the first master goes into a HIGH wait state until the SCL clock line transitions to 1.

All masters then count out their HIGH time and the master with the shortest HIGH time transitions the SCL line to 0. The masters then count 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 shown in Figure 94. Optionally, slaves may hold the SCL line LOW to slow down the timing on the I2C bus.

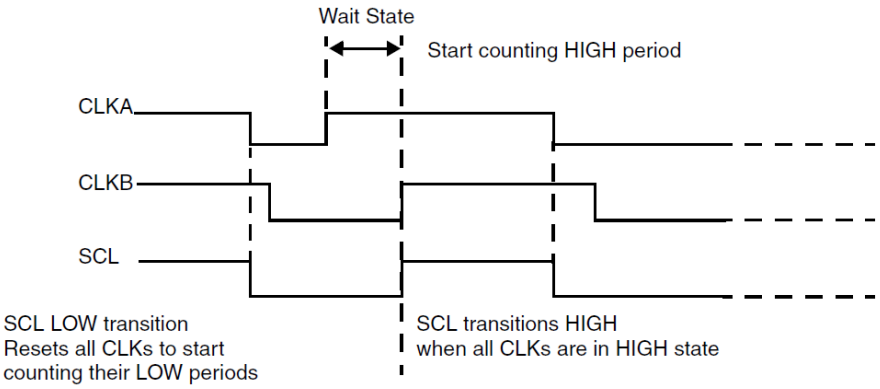


Figure 94. Multiple master clock synchronization

9.14.3 Programming

To configure and use the I2C Controller:

1. Set up the GPIOs to be used for the I2C interface (P0x_MODE_REG[PID] = 9 or 10).
2. Enable the clock for the I2C Controller (CLK_PER_REG[I2C_ENABLE] = 0x1).
3. Disable the I2C Controller (I2C_ENABLE_REG = 0).
4. Configure the I2C clock frequency:

a. Standard mode (100 kbit/s) : I2C_CON_REG[I2C_SPEED] = 1.
b. Full speed mode (400 kbit/s) : I2C_CON_REG[I2C_SPEED] = 2.

5. Setup the I2C Controller as:
   a. Master: I2C_CON_REG[I2C_MASTER_MODE] = 1 and I2C_CON_REG[I2C_SLAVE_DISABLE] = 1.
   b. Slave: I2C_CON_REG[I2C_MASTER_MODE] = 0 and I2C_CON_REG[I2C_SLAVE_DISABLE] = 0.
6. Choose whether the controller starts its transfers in the 7-bit or 10-bit addressing format when acting as a master (I2C_CON_REG[I2C_10BITADDR_MASTER]) or whether the controller responds to the 7-bit or 10-bit addresses when acting as a slave (I2C_CON_REG[I2C_10BITADDR_SLAVE]).
7. Set target slave address in:
   a. Master mode (I2C_TAR_REG[IC_TAR] = 0x55 (default)).
   b. Slave mode (I2C_SAR_REG[IC_SAR] = 0x55 (default)).
8. Set the threshold levels on RX and TX FIFO (I2C_RX_TL_REG and I2C_TX_TL_REG).
9. Enable the required interrupts (I2C_INTR_MASK_REG).
10. Enable the I2C Controller (I2C_ENABLE_REG = 0x1).
11. Read a byte:
    a. Prepare to transmit the read command byte (I2C_DATA_CMD_REG[I2C_CMD] = 1).
    b. Wait until TX FIFO is empty (I2C_STATUS_REG[TFE] = 1).
    c. Wait until master has finished reading the byte from the slave device (I2C_STATUS_REG[MST_ACTIVITY] = 0).
12. Write a byte:
    a. Prepare to transmit the write command byte (I2C_DATA_CMD_REG[I2C_CMD] = 0 and I2C_DATA_CMD_REG[I2C_DAT] = command byte).
    b. Wait until TX FIFO is empty (I2C_STATUS_REG[TFE] = 1).
    c. Wait until master has finished reading the response byte from the slave device (I2C_STATUS_REG[MST_ACTIVITY] = 0).

9.15 UART

9.15.1 Introduction

The RA6W2 contains two instances of the UART block, that is, UART and UART2.

The UART is compliant to the industry-standard 16550 and is used for serial communication with a peripheral. Data is written from a master (CPU) over the APB bus to the UART and it is converted to the 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 also DMA support on the UART block, thus the internal FIFOs can be used. Only UART supports the hardware flow control signals (RTS and CTS) and the 9-bit mode.

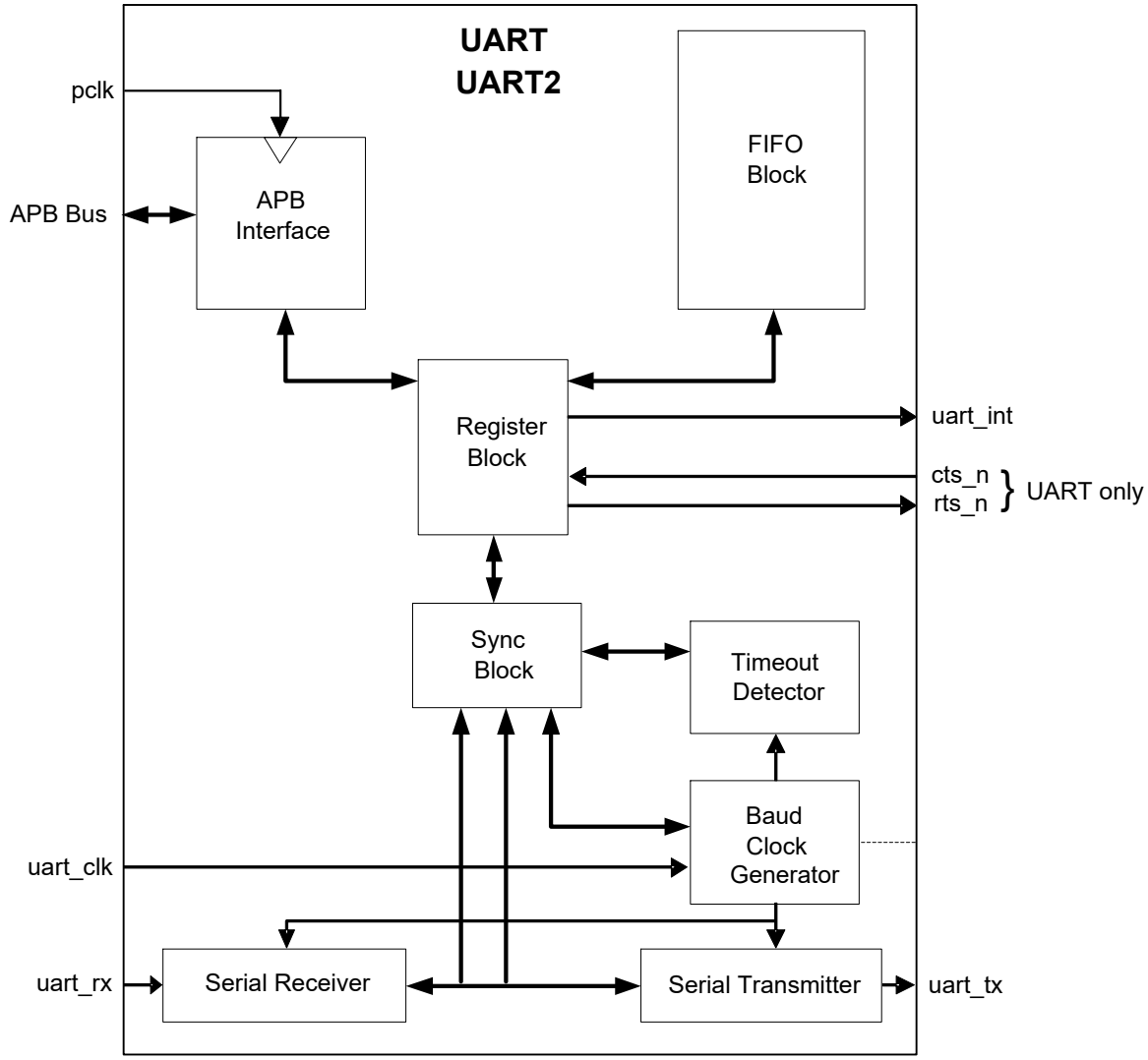


Figure 95. UART block diagram

Features

- 16-byte transmit and receive FIFOs
- Hardware flow control (CTS/RTS) (UART only)
- Shadow registers to reduce software overhead and a software programmable reset is included
- Transmitter Holding Register Empty (THRE) interrupt 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.

9.15.2 Architecture

9.15.2.1 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 the start and stop bits is referred to as a character (Figure 96).

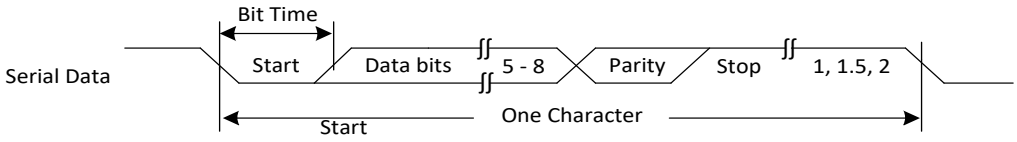


Figure 96. Serial data format

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 and then by the stop bit(s), which can be of 1, 1.5, or 2 bits.

All the bits in the transmission (except the half stop bit when the 1.5 stop bits are used) are transmitted for the same time duration. This is referred to as a Bit Period or Bit Time. One Bit Time equals to 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 has been transmitted for is known, the mid-point for sampling is every 16 baud clocks after the mid-point sample of the start bit. Figure 97 shows the sampling points of the first couple of bits in a serial character.

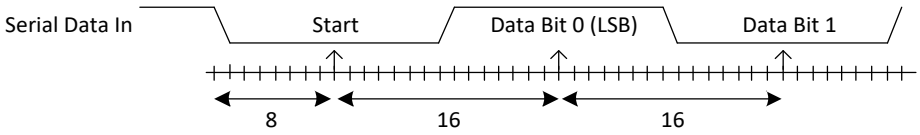


Figure 97. Receiver serial data sampling points

As part of the 16550 standards, an optional baud clock reference output signal (baudout_n) is supplied to provide timing information to the 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) in the following equation:

$$\text{baud rate} = (\text{serial clock frequency})/(16 \times \text{divisor}) \quad (2)$$

where the divisor is a 16-bit integer value plus 4-bit fractional value. The divisor range is 0 to 65535,9375 with steps of 1/16. Divisor High 8-bit integer part is in the DLH register. Divisor Low 8-bit integer part is in the DLL register. Divisor 4-bit fractional port is in the DLF register.

The registers settings for the common baud rate values are presented in Table 75.

Table 75: UART baud rate generation

Baud rate	Divider	Divisor latch	DLH Reg	DLL Reg	DLF Reg	Actual BR	Error (%)
1200	833,333	833,3125	3	65	5	1200,03	0,00
2400	416,667	416,6875	1	160	11	2399,88	0,00

Baud rate	Divider	Divisor latch	DLH Reg	DLL Reg	DLF Reg	Actual BR	Error (%)
4800	208,333	208,3125	0	208	5	4800,48	0,01
9600	104,167	104,1875	0	104	3	9598,08	0,02
19200	52,083	52,0625	0	52	1	19207,68	0,04
38400	26,042	26,0625	0	26	1	38369,30	0,08
57600	17,361	17,375	0	17	6	57553,96	0,08
115200	8,681	8,6875	0	8	11	115107,91	0,08
230400	4,340	4,3125	0	4	5	231884,06	0,64
460800	2,170	2,1875	0	2	3	457142,86	0,79
921600	1,085	1,0625	0	1	1	941176,47	2,12
1000000	1	1	0	1	0	1000000	0,00

9.15.2.2 Clock Support

The UART has two system clocks (`pclk` and `sclk`). Having a second asynchronous serial clock (`sclk`) allows for accurate serial baud rate settings and meeting APB bus interface requirements.

With the two-clock design, a synchronization module is implemented to synchronize all controls and data across the two system clock boundaries.

Although 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, 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 the initial configuration of the UART.

9.15.2.3 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 76.

Table 76: UART Interrupt Priorities

Interrupt ID bits [3-0]	Interrupt Set and Reset functions			
	Priority	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 four-	Reading the receiver buffer register

Interrupt ID bits [3-0]	Interrupt Set and Reset functions			
	Priority	Interrupt type	Interrupt source	Interrupt reset control
			character times and there is at least one character in it during this time.	
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 to check whether there is an interrupt and what its source is; 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	-	-

9.15.2.4 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 shown in the flowchart in Figure 98. Figure 99 shows the programmed transmitter FIFO empty threshold level, where THRE Interrupts are active when the FIFO is empty. In this case the programmable THRE interrupt mode is disabled.

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 its function from indicating transmitter FIFO empty to FIFO full. This allows software to fill the FIFO in each transmit sequence by polling LSR[5] before writing another character. Instead of waiting until the FIFO is completely empty, the flow becomes "fill transmitter FIFO whenever an interrupt occurs and there is data to transmit". 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 99.

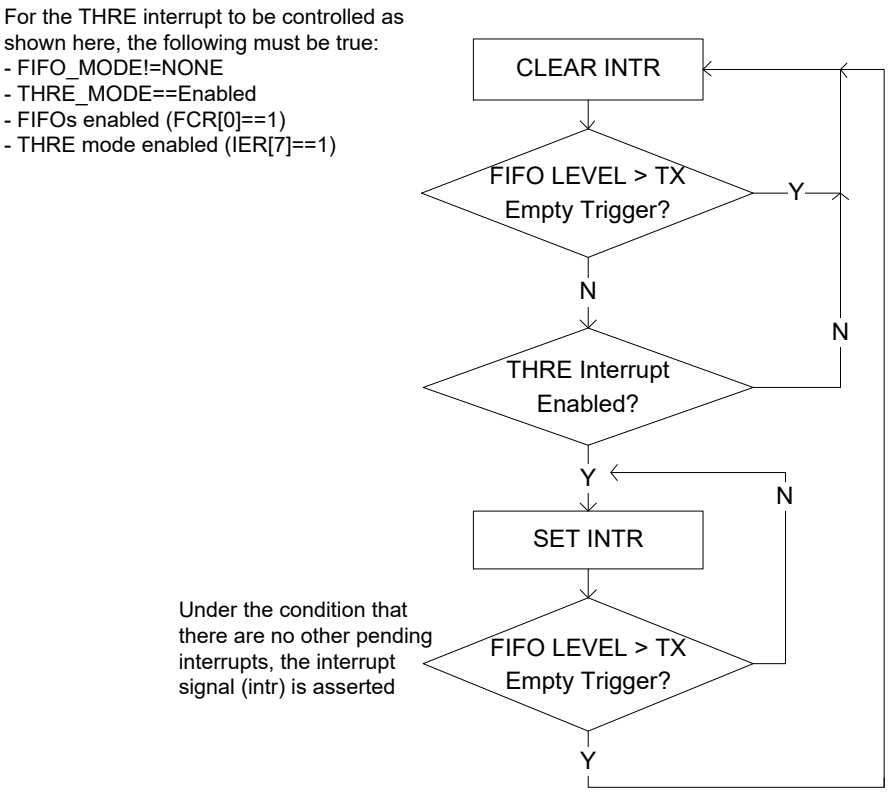


Figure 98. Flowchart of interrupt generation for programmable THRE interrupt mode

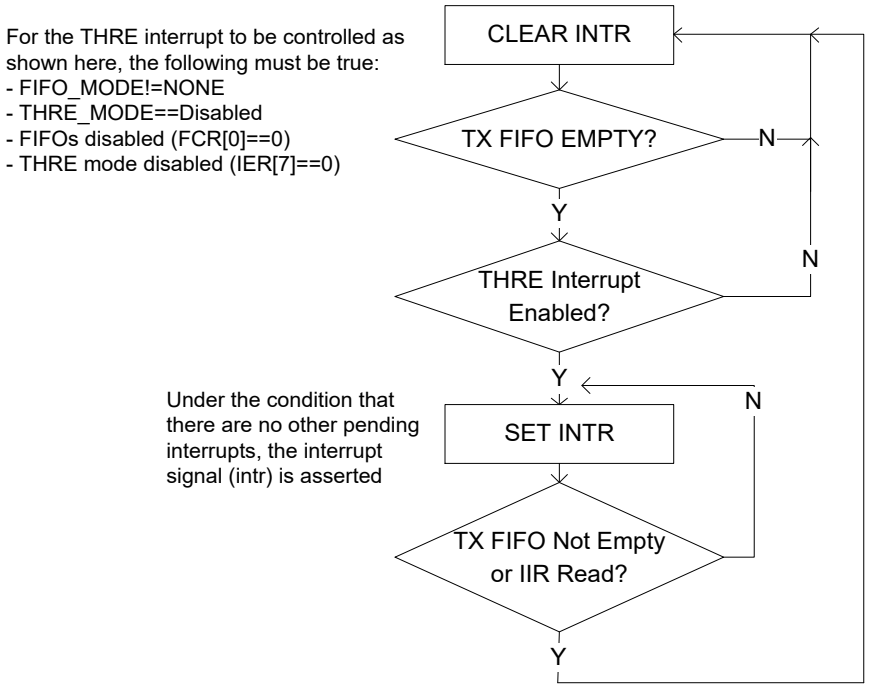


Figure 99. Flowchart of interrupt generation when not in programmable THRE interrupt mode

9.15.2.5 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 supports a host burst mode where the host increments its address but still accesses the same receive buffer register
- UART_STHR supports 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.

9.15.2.6 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 by using a 2-wire UART directly as described in the Bluetooth Low Energy Specification (Volume 6, Part F).

9.15.3 Programming

To configure and use the UART controllers:

1. Set up the GPIOs to be used for the UART interface (P0x_MODE_REG[PID] = 1 to 4 and/or 19-20).
2. Enable the selected UART by setting the CLK_PER_REG[UARTx_ENABLE] bit.
3. Enable access to Divisor Latch Registers (DLL and DLH) by setting the UARTx_LCR_REG[UART_DLAB] bit.
4. Set the desired baud rate. To calculate the registers values for the desired baud rate, use the following equation:

$$\text{Divisor} = \text{UART CLK}/(16 \times \text{Baud rate}) \quad (3)$$

   a. UARTx_IER_DLH_REG: High byte of the Divisor integer part.
   b. UARTx_RBR_THR_DLL_REG: Low byte of the Divisor integer part.
   c. UARTx_DLF_REG: The fractional part of the Divisor.
5. Configure the break control bit, parity, number of stop bits, and data length (UARTx_LCR_REG).
6. Enable and configure the FIFO (UARTx_IIR_FCR_REG).
7. Configure the generated interrupts, if needed (UARTx_IER_DLH_REG).
8. Send a byte:
   a. Check if Transmit Hold Register (THR) is empty (UARTx_LSR_REG[UART_THRE]).
   b. Load the byte to THR (UARTx_RBR_THR_DLL_REG).
   c. Check if the byte has been transmitted (UARTx_LSR_REG[UART_TEMT]).
9. Receive a byte:
   a. Wait until serial data is ready (UARTx_LSR_REG[UART_DR]).
   b. Read the incoming byte from the THR (UARTx_RBR_THR_DLL_REG).

9.16 Interface

9.16.1 Introduction

This controller implements the Serial Peripheral Interface (SPI™) for master and slave modes. The serial interface can transmit and receive from four bits to up to 32 bits in master/slave mode. The controller comprises separate TX and RX FIFOs and DMA handshake support. Slave mode clock speed is independent from the system clock speed. Moreover, master's clock speed can be as fast as the system's clock speed. The controller can generate an interrupt upon data threshold reached in the TX or RX FIFOs.

Features

- Slave and master mode
- From 4-bit to up to 32-bit operation
- SPI Master clock line speed up to 32 MHz, SPI Slave clock line speed up to 16 MHz
- SPI mode 0, 1, 2, and 3 support (clock edge and phase)
- Built-in separate 8-bit wide and 4-byte deep RX/TX FIFOs for continuous SPI bursts
- Maskable interrupt generation based on TX or RX FIFO thresholds
- DMA support.

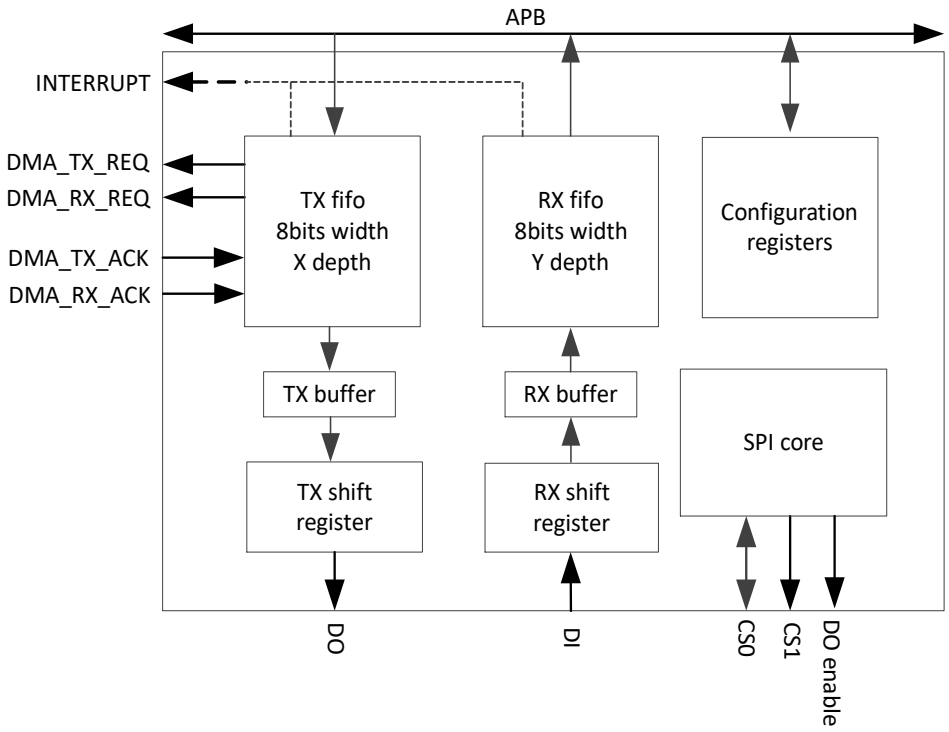


Figure 100. SPI block diagram

9.16.2 Architecture

The SPI controller is an APB peripheral operating on the apb_clk clock. It contains a front end which is clocked by the spi_clk clock and is responsible for the serialization/deserialization of the data in the RX and TX streams.

Two separate FIFOs, each of eight bits wide and four bytes deep, are used to store data for RX and TX streams. Since a SPI word can be configured to be from four bits to up to 32 bits, one to four FIFO positions can be written/read at the same time. FIFOs contain logic implementing programmable thresholds comparison.

The SPI controller supports DMA requests and interrupt generation based on the FIFO thresholds. If enabled, a DMA request and/or interrupt is asserted with whether TX_FIFO level is low or RX_FIFO level is high.

The SPI interface supports all four modes of operation and the corresponding polarity (CPOL) and phase (CPHA) of the SPI clock (SPI_CLK) are defined in Table 77.

Table 77: SPI modes configuration and SCK states

SPI mode	CPOL	CHPA	TX SPI_CLK	RX SPI_CLK	Idle SPI_CLK
0	0	0	Falling edge	Rising edge	Low
1	0	1	Rising edge	Falling edge	Low
2	1	0	Rising edge	Falling edge	High
3	1	1	Falling edge	Rising edge	High

To read from or to write to an external single byte Flash device in the SPI master mode, a byte swap mechanism is implemented to allow for a proper placement of the bytes in a 16-bit word for the DMA to write to/read from the internal RAM. More specifically, when the SPI controller is configured as a master with DMA support and a 16-bit word width so that the bus utilization is increased compared to reading from an 8-bit device, the byte swap mechanism brings the least significant byte read and place it in the most significant byte in the 16-bit word. The controller automatically swaps the bytes to allow for placing the first byte read in the least significant byte of the 16-bit word. This feature is programmable via SPI_CTRL_REG[SPI_SWAP_BYTES].

The SPI controller can operate at the highest speed (32 MHz on the SPI_CLK line) in a special master mode. The clock of the controller is then either the XTAL32M or the RC32M and can be used for fast booting from external FLASH devices that support this frequency.

SPI Timing

The timing of the SPI interface when the SPI controller is in slave mode is presented in Figure 101.

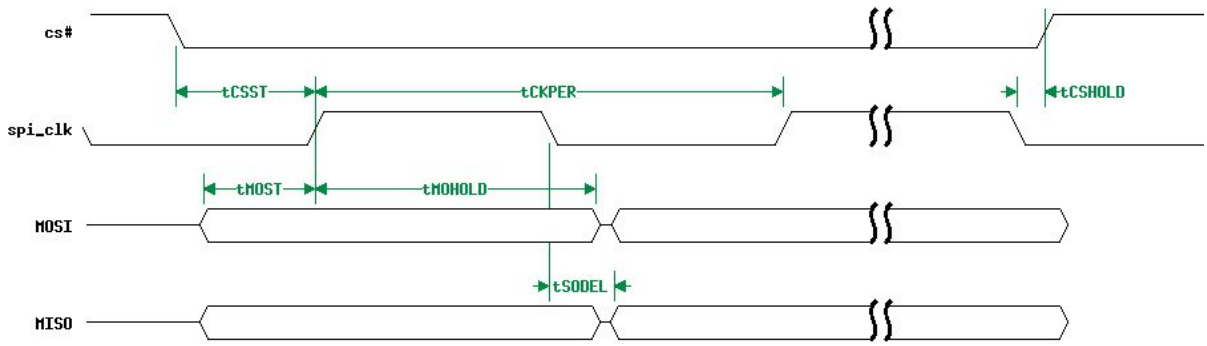


Figure 101. SPI Slave mode timing (CPOL = 0, CPHA = 0)

Table 78: SPI timing parameters

Parameter	Description	Typ	Unit
t_{CKPER}	spi_clk clock period	60	ns
t_{CSST}	CS active time before the first edge of spi_clk	1 spi_clk cycle	
t_{CSHOLD}	CS non-active time after the last edge of spi_clk	1 spi_clk cycle	
t_{MOST}	Master input data latching setup time	(spi_clk/2) - 5	ns
t_{MOHOLD}	Master input data hold time	0	ns
t_{SODEL}	Slave output data delay	25	ns

9.16.3 Programming

9.16.3.1 Master Mode

To configure the SPI controller in master mode, follow the steps below:

1. Set the appropriate GPIO ports in SPI clock mode (output), SPI Chip Select mode (output), SPI Data Out mode (output), and SPI Data In mode (input).
2. Enable the SPI clock by setting CLK_PER_REG[SPI_ENABLE] = 1.
3. Reset SPI FIFO by setting SPI_CTRL_REG[SPI_FIFO_RESET] = 1.

4. Set the SPI clock mode (synchronous or asynchronous with APB clock) by programming SPI_CLOCK_REG[SPI_MASTER_CLK_MODE].
5. Set the SPI clock frequency by programming SPI_CLOCK_REG[SPI_CLK_DIV]. If SPI_CLK_DIV is not 0x7F, SPI_CLK = module_clk/2 × (SPI_CLK_DIV + 1). If SPI_CLK_DIV = 0x7F, SPI_CLK = module_clk.
6. Set the SPI mode (CPOL or CPHA) by programming SPI_CONFIG_REG[SPI_MODE].
7. Set the SPI controller in master mode by setting SPI_CONFIG_REG[SPI_SLAVE_EN] = 0.
8. Define the SPI word length (from 4-bit to 32-bit) by programming SPI_CONFIG_REG[SPI_WORD_LENGTH]. SPI_WORD_LENGTH = word length - 1.

To read/write the following sequence should be performed:

1. If a slave device is slow and does not give the data at the correct clock edge, configure the SPI module to capture data at the next clock edge by setting SPI_CTRL_REG[SPI_CAPTURE_AT_NEXT_EDGE] = 1. Otherwise, set SPI_CTRL_REG[SPI_CAPTURE_AT_NEXT_EDGE] = 0.
2. Release the FIFO reset by setting SPI_CTRL_REG[SPI_FIFO_RESET] = 0.
3. Enable the SPI TX path by setting SPI_CTRL_REG[SPI_TX_EN] = 1.
4. Enable the SPI RX path by setting SPI_CTRL_REG[SPI_RX_EN] = 1.
5. Enable the SPI chip select by programming the SPI_CS_CONFIG_REG[SPI_CS_SELECT] = 1 or 2. This option allows the master to select the slave that is connected to the GPIO that has the function of SPI_CS0 or SPI_CS1.
6. Enable the SPI controller by setting SPI_CTRL_REG[SPI_EN] = 1.
7. Write to TX FIFO by programming SPI_FIFO_WRITE_REG[SPI_FIFO_WRITE]. Write access is permitted only when SPI_FIFO_STATUS_REG[SPI_TX_FIFO_FULL] = 0.
8. Read from RX FIFO by programming SPI_FIFO_READ_REG[SPI_FIFO_READ]. Read is permitted only when SPI_FIFO_STATUS_REG[SPI_RX_FIFO_EMPTY] = 0.
9. To disable the SPI chip select, set SPI_CS_CONFIG_REG[SPI_CS_SELECT] = 0 to deselect the slave and set SPI_CTRL_REG[SPI_FIFO_RESET] = 1 to reset the SPI FIFO.

9.16.3.2 Slave Mode

1. Set the appropriate GPIO ports in SPI clock mode (input), SPI Chip Select mode (input), SPI Data Out mode (output), and SPI Data In mode (input).
2. Enable the SPI clock by setting CLK_PER_REG[SPI_ENABLE] = 1.
3. Reset the SPI FIFO by setting SPI_CTRL_REG[SPI_FIFO_RESET] = 1.
4. Set the SPI mode (CPOL or CPHA) by programming SPI_CONFIG_REG[SPI_MODE].
5. Set the SPI module in slave controller by setting SPI_CONFIG_REG[SPI_SLAVE_EN] = 1.
6. Define the SPI word length (from 4-bit to 32-bit) by programming SPI_CONFIG_REG[SPI_WORD_LENGTH]. SPI_WORD_LENGTH = word length - 1.

To read/write the following sequence must be performed:

1. Set SPI FIFO in normal operation by setting SPI_CTRL_REG[SPI_FIFO_RESET] = 0.
2. Enable the SPI TX path by setting SPI_CTRL_REG[SPI_TX_EN] = 1.
3. Enable the SPI RX path by setting SPI_CTRL_REG[SPI_RX_EN] = 1.
4. Enable the SPI controller by setting SPI_CTRL_REG[SPI_EN] = 1.
5. Write the first data byte directly to TX buffer by programming the SPI_TXBUFFER_FORCE_L_REG[SPI_TXBUFFER_FORCE_L].
6. Write the rest of the data to TX FIFO by programming SPI_FIFO_WRITE_REG[SPI_FIFO_WRITE]. Write access is permitted only if SPI_FIFO_STATUS_REG[SPI_TX_FIFO_FULL] = 0.
7. Read from RX FIFO by programming SPI_FIFO_READ_REG[SPI_FIFO_READ]. Read is permitted only if SPI_FIFO_STATUS_REG[SPI_RX_FIFO_EMPTY] = 0.

9.17 Quadrature Decoder

9.17.1 Introduction

The RA6W2 has an 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. It can also be programmed to simply count rising/falling edges on any of the channel pairs. This block can be used to wake up the chip as soon as there is a movement from the connected external device. The block diagram of the quadrature decoder is presented in Figure 102.

Features

- Three 16-bit signed counters that provide the step count and direction on each of the axes (X, Y, and Z) and one 8-bit counter counting the overall edges from all three counters.
- Programmable system clock sampling at a maximum of 16 MHz.
- APB interface for control and programming.
- Programmable source from the GPIOs.
- Digital filter on the channel inputs to avoid spikes.

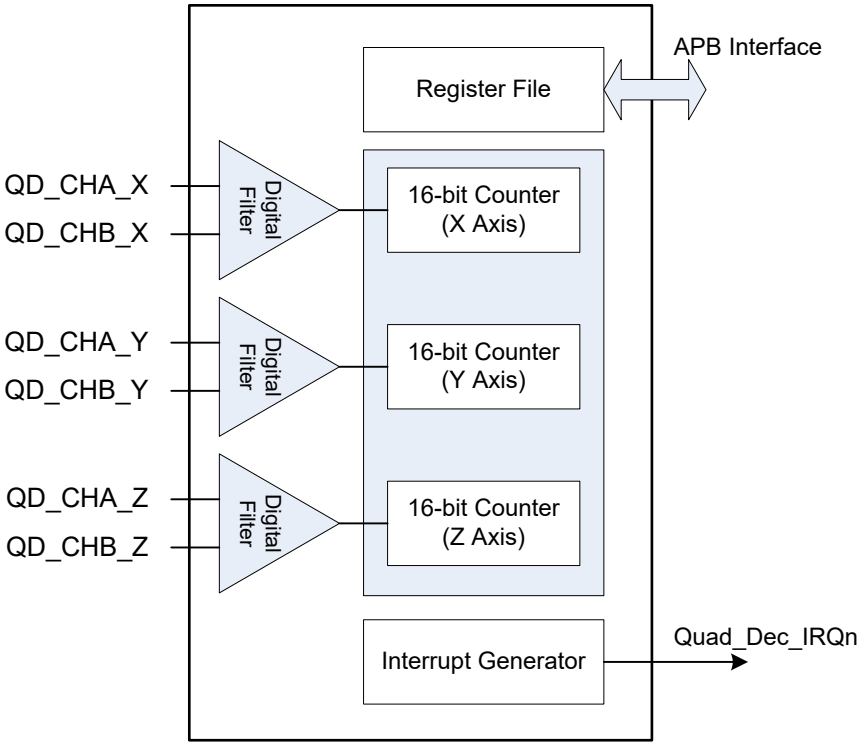


Figure 102. Quadrature decoder block diagram

9.17.2 Architecture

Channels are expected to provide a pulse train with 90 degrees rotation as displayed in Figure 103 and Figure 104.

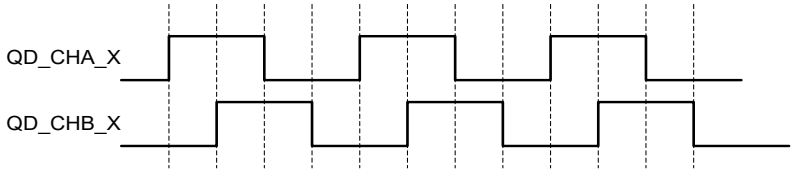


Figure 103. Moving forward on axis X

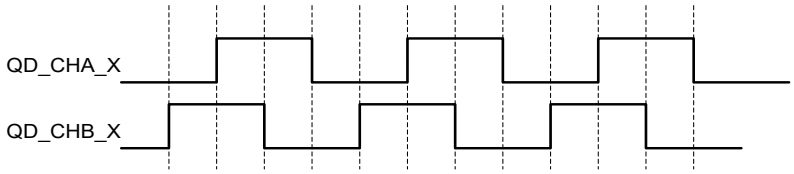


Figure 104. Moving backwards on axis X

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.

Users can choose which GPIOs to use for the channels by programming the QDEC_CTRL2_REG register. The block supports two modes of operation: quadrature counting and edges counting. The quadrature counting mode reads the patterns of successive pulses as in Figure 103 and Figure 104, while the edges counting mode simply counts all positive and negative edges on any of the two channels of a pair.

NOTE
If two edges happen at the same time, the counter only counts one.

The digital filter eliminates the spikes shorter than three clock periods. It is followed by an edge detection circuitry and they are shown in Figure 105.

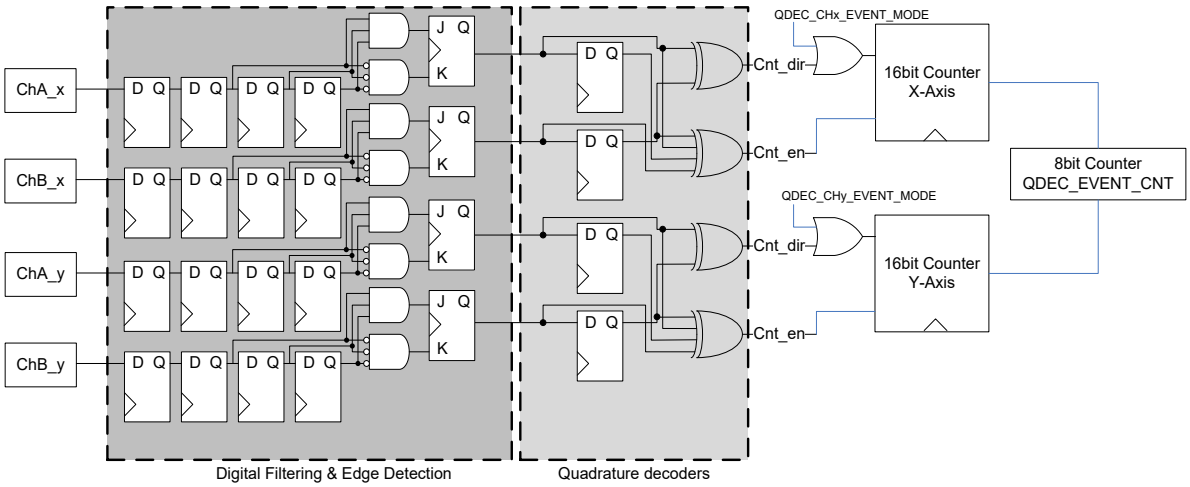


Figure 105. Digital filtering and edge detection circuit

A counter for its dedicated axis holds the movement events of the channels. When a channel is disabled, the counter is reset. The counters are accessible via the APB bus.

The QDEC_EVENT_CNT gathers all edges on all channels regardless of the mode of operation. If two edges happen at the same time, this counter is only increased by one.

The quadrature decoder operates on the system clock. The QDEC_CLOCKDIV register defines the number of clock cycles during which the decoding logic samples the data on the channel inputs. The division is automatically disabled when the lp_clk is used as the system clock.

9.17.3 Programming

To program the quadrature decoder for actual quadrature counting or edge counting:

1. Configure the clock frequency by configuring the QDEC_CLOCKDIV register. The value in this register is dividing the sys_clk. However, if sys_clk = lp_clk, this divider is completely bypassed.
2. Define which pin pairs represent the different channels for the X, Y, and Z axes or the GPIOs from which the edges are counted. Configure such information at QDEC_CHX_PORT_SEL, QDEC_CHY_PORT_SEL, and QDEC_CHZ_PORT_SEL registers.
3. Configure the interrupt threshold upon which an interrupt is generated at QDEC_CTRL_REG[QDEC_IRQ_THRES]. Note that the interrupt threshold is based on the value of QDEC_EVENT_CNT_REG which keeps on counting after the interrupt is generated.

4. Define the mode of operation by configuring the respective QDEC_CHx_EVENT_MODE field in the QDEC_CTRL2_REG.
5. Enable the clock of the block by writing at CLK_PER_REG[QUAD_ENABLE].
6. Wait for the interrupt and then read X, Y, and Z values at QDEC_XCNT_REG, QDEC_YCNT_REG, and QDEC_ZCNT_REG (in the quadrature counting case) or the QDEC_EVENT_CNT_REG (in the edges counting case).
7. Clear the interrupt (by writing at QDEC_CTRL_REG[QDEC_IRQ_STATUS]) and the edge counter (by writing at QDEC_CTRL_REG[QDEC_EVENT_CNT_CLR) if needed.

9.18 Clockless Wake-Up Controller

9.18.1 Introduction

The Clockless Wake-Up Controller implements a circuit that enables the RC32K clock which in turn triggers the hardware Startup FSM to allow the system to be woken up by an external event (a GPIO toggle). This controller is only used when the system is in hibernation mode, that is, when all clocks are stopped.

Features

- Wake up the system from specific GPIOs (P0_1, P0_2, P0_3, P0_4, and P0_5)
- Configurable polarity of the GPIO signals (single configuration for all GPIOs)
- Special RC filtered inputs feeding the wake-up control circuit (Type B pads)
- Always powered.

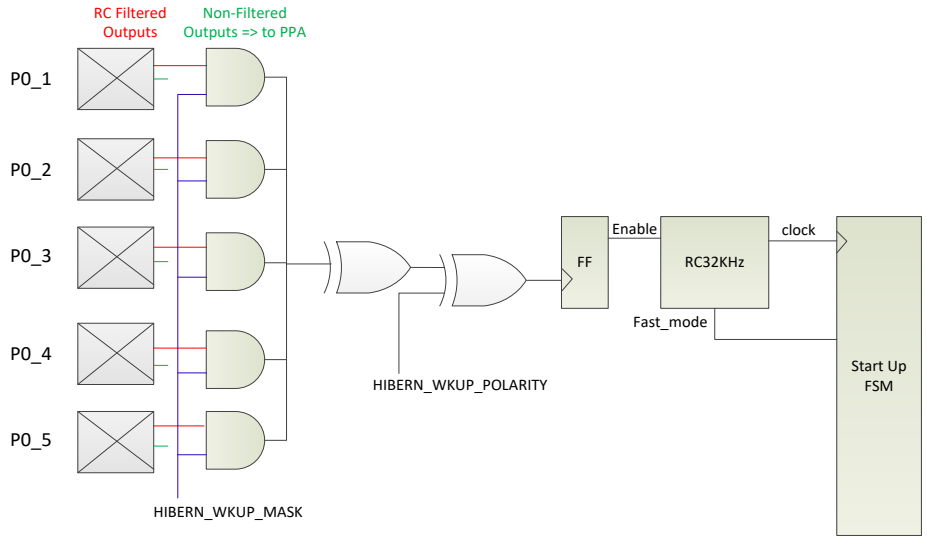


Figure 106. Clockless wake-up controller circuit

9.18.2 Architecture

The Clockless Wake-Up Controller automatically enables the RC32K oscillator when a toggle is identified in one of the five specific GPIOs (P0_1, P0_2, P0_3, P0_4, and P0_5). These GPIO signals are connected to the Type B pads, which provide two outputs to the digital domain. One output goes through a Schmitt trigger and the other one goes through an RC filter with a cutoff frequency of 100 kHz and a Schmitt trigger. The output going through the RC filter and a Schmitt trigger feeds the clockless wake-up controller circuitry. Hence, any spikes larger than 100 kHz are filtered out without waking up the system.

The triggering GPIO can be defined by means of a masking register. If no GPIO is masked out, any toggle in any of these GPIOs creates an edge which serves as a clock for a Flip-Flop to lock and enable the oscillator. The polarity of the edge is also programmable, but it is common for the GPIOs and not a dedicated bit per GPIO.

Note that, if two opposite edges occur exactly at the same time on two GPIOs that are allowed to wake up the system, the XOR output does not change its value and no wake up occurs. However, even if the GPIO signal events have a couple of ns difference, the circuit still understands it and the clock is started.

9.18.3 Programming

To program the clockless wake-up controller before setting the system into hibernation mode:

1. Define which pins are allowed to wake up the system from hibernation by configuring the HIBERN_CTRL_REG[HIBERN_WKUP_MASK].
2. Define the polarity of the waking up events at HIBERN_CTRL_REG[HIBERN_WKUP_POLARITY]. This should be done by reading the value of the unmasked GPIOs and programming the polarity register with their XOR'ed state.
3. Enable the hibernation mode by programming the HIBERN_CTRL_REG[HIBERNATION_ENABLE] bit field. Note that this action stops all clocks when the system drops to sleep.

4. Allow RAM to be retained by programming the RAM_PWR_CTRL_REG accordingly.
5. Define where address 0 is to be mapped at SYS_CTRL_REG[REMAP_ADR0] so that the CPU can execute code right after waking up. If RAM is retained, REMAP_ADR0 should point to 0x2 or 0x3.
6. Clear RESET_STAT_REG. Clear this register means that the system wakes up from the hibernation mode.
7. Put the system to sleep by executing the WFI command with the SCR bit set.

9.19 Clocked Wake-Up Controller

9.19.1 Introduction

The Clocked Wake-Up Controller can be programmed to wake up the RA6W2 from deep sleep mode and extended sleep mode upon a pre-programmed number of GPIO events on a maximum of two pins in parallel. This wake-up controller resides in the PD_SLP power domain and operates on the LP_CLK.

Figure 107 shows the block diagram with the wake-up function.

Features

- Monitors GPIO state changes
- Implements debouncing time from 0 ms up to 63 ms on two GPIOs in parallel
- Accumulates external events and compares the number to a programmed value
- Generates an interrupt to the CPU's WIC.

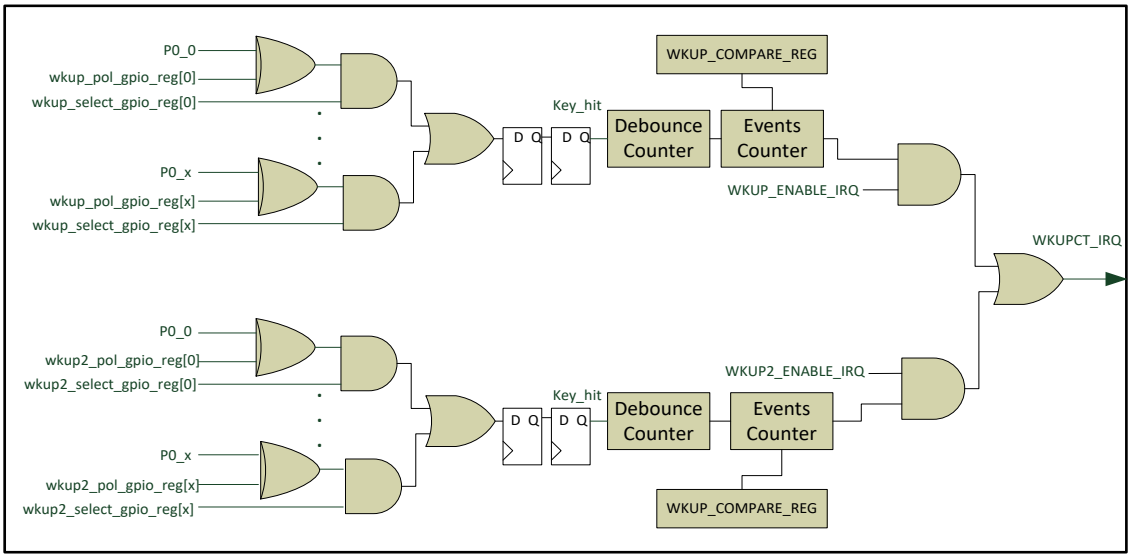


Figure 107. Clocked wake-up controller block diagram

9.19.2 Architecture

The controller comprises two identical circuits that implement the edge detection, debouncing, and event counting before generating a wake-up interrupt towards the CPU.

A LOW to HIGH level transition on the selected input port sets internal signal "key_hit" to 1, while WKUP_POL_GPIO_REG[y] = 0. This signal triggers the event counter state machine as shown in Figure 108. The debounce counter is loaded with the value of WKUP_CTRL_REG[WKUP_DEB_VALUE]. The timer counts down every 1 ms. The signal state is constantly monitored. If the debounce counter reaches 0, it means that the key_hit signal state has been stable over the amount of clock cycles counted by the debounce counter.

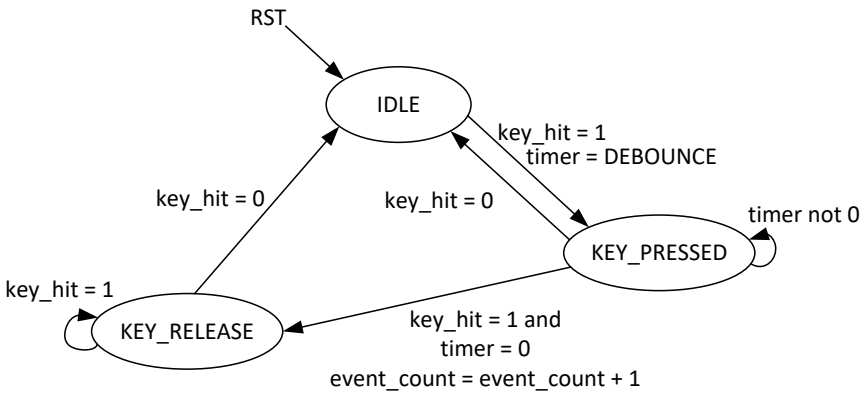


Figure 108. Event counter state machine for the wake-up interrupt generator

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

If the event counter is equal to the value set in the WKUP_COMPARE_REG register, the counter is reset and an interrupt is generated, if the interrupt generation has been enabled by WKUP_ENABLE_IRQ and WKUP2_ENABLE_IRQ in the WKUP_CTRL_REG.

NOTE
There is only one register for both circuits that contains the number of events before an interrupt is issued.

The interrupt can be cleared by writing a value to the register WKUP_RESET_IRQ_REG. The event counter can be reset by writing a value to the register WKUP_RESET_CNTR_REG. The value of the event counter can be read at any time by reading register WKUP_COUNTER_REG.

Any of the GPIO inputs can be selected to generate an event by programming the corresponding WKUP/WKUP2_SELECT_GPIO_REG register. When both WKUP/WKUP2_SELECT_GPIO_REG registers are configured to generate a wake-up interrupt, a toggle on any GPIO wakes up the system.

The input signal edge can be selected by programming the WKUP/WKUP2_POL_GPIO_REG registers.

NOTE
A minimum of 2 low power clocks pulse is required on a GPIO to be correctly identified as a wake-up edge trigger

9.19.3 Programming

To configure the clocked wake-up controller:

1. Define the polarity of the triggering GPIOs at WKUP/WKUP2_POL_GPIO_REG.
2. Configure the debouncing counters by programming WKUP_CTRL_REG[WKUP_DEB_VALUE] with the amount of time (ms) during which the signal should be re-sampled before its state is decided. Note that there is a single bit field for both debouncing counters.
3. Define the number of events that are needed to trigger the wake-up interrupt by programming the WKUP_COMPARE_REG. Note there is only one register for both circuits.
4. Allow the interrupt generation by configuring the WKUP_ENABLE_IRQ and WKUP2_ENABLE_IRQ bit fields, respectively, in the WKUP_CTRL_REG.
5. Define which GPIOs are allowed to trigger a wake-up event at WKUP/WKUP2_SELECT_GPIO_REG.
6. Set the system to deep sleep mode or extended sleep mode by executing the WFI command with the SCR bit set.

9.20 Timer 0

9.20.1 Introduction

Timer 0 is a 16-bit general purpose software programmable timer, which has the ability of generating Pulse Width Modulated (PWM) signals 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. Figure 109 shows the block diagram of Timer 0.

Features

- 16-bit general purpose timer
- Ability to generate two PWM signals (PWM0 and PWM1)
- Programmable output frequency (f) with N = 0 to (2^{16}-1) and M = 0 to (2^{16}-1)

$$f = \frac{(16,\ 8,\ 4,\ 2\ MHz\ or\ 32\ kHz)}{(M+1)+(N+1)}$$

- Programmable duty cycle (δ):

$$\delta = \frac{M+1}{(M+1)+(N+1)} \times 100\ \%$$

- Separately programmable interrupt timer:

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

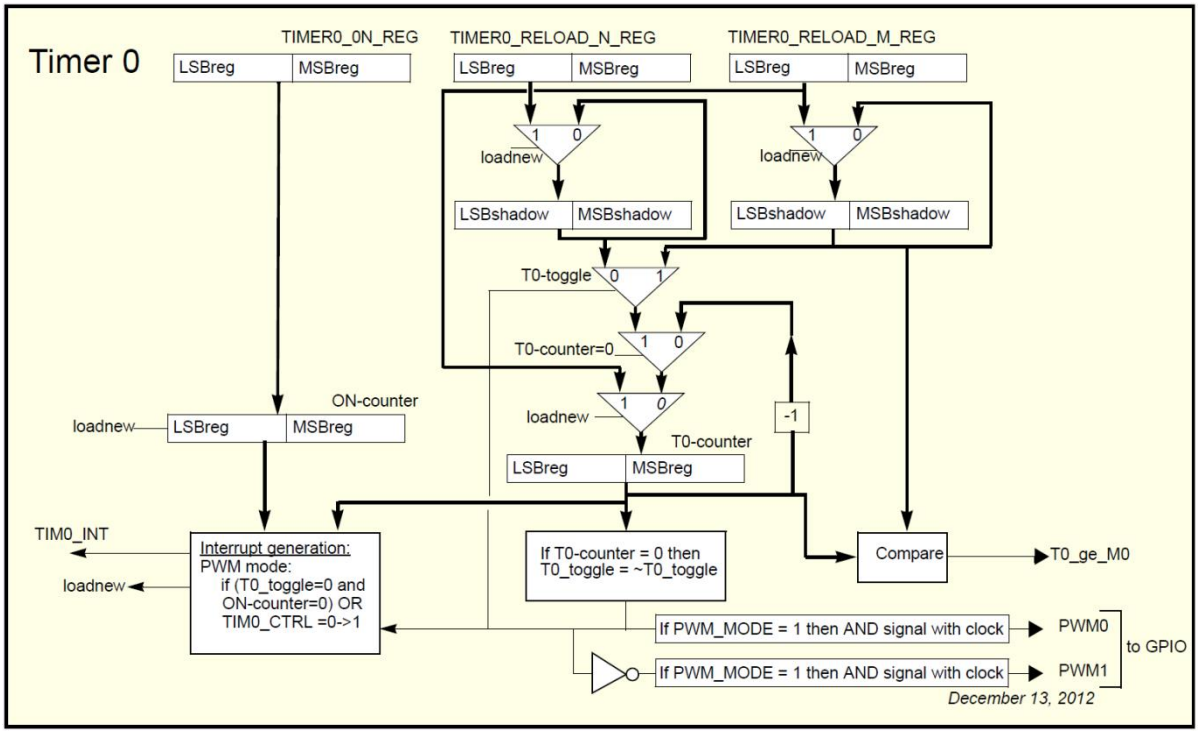


Figure 109. Timer 0 block diagram

9.20.2 Architecture

The 16-bit Timer 0 consists of two counters, that is, T0-counter and ON-counter, and three registers, that is, TIMER0_RELOAD_M_REG, TIMER0_RELOAD_N_REG, and TIMER0_ON_REG. Upon reset, the counter and register values are 0x0000. Timer 0 generates a PWM signal PWM0, of which the frequency and duty cycle are determined by the contents of the TIMER0_RELOAD_N_REG and the TIMER0_RELOAD_M_REG registers. The PWM1 signal is the inverted version of PWM0.

Timer 0 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 (by default the TMR_ENABLE bit is disabled). This register also controls the four higher clock frequency on which Timer 0 runs via the TMR_DIV bits. An extra clock divider is available and 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 the clock for the ON-counter by 10.

NOTE
If the LP clock is selected as system clock, the CLK_AMBA_REG[HCLK] bit field should always be 0 to ensure the proper operation of the timer.

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 starts running. SWTIM_IRQ is generated, and the T0-counter loads its start value from the TIMER0_RELOAD_M_REG register and decrements on each clock cycle. 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 is toggled to select the TIMER0_RELOAD_N_REG whose value is loaded in the T0-counter. Each time the T0-counter reaches zero, it is alternately be reloaded with the values of the M0- and N0-shadow registers. PWM0 is 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 when the 32-kHz clock is used), but the selected clock frequency is divided by two to get a 50% duty cycle.

If the ON-counter reaches zero, it remains 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 then generates an interrupt (SWTIM_IRQ). The ON-counter is reloaded with the value of the TIMER0_ON_REG register. The T0-counter as well as the M0-shadow register is loaded with the value of the TIMER0_RELOAD_M_REG register. At the same time, the N0-shadow register is loaded with the value of TIMER0_RELOAD_N_REG register. Both counters are decremented on the next clock again and the sequence is repeated.

NOTE
It is possible to generate interrupts at a high rate by selecting a high clock frequency and low counter values. This could result in missed interrupt events.

During the time when 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 is decrementing the value loaded from the TIMER0_RELOAD_N_REG register (Figure 110).

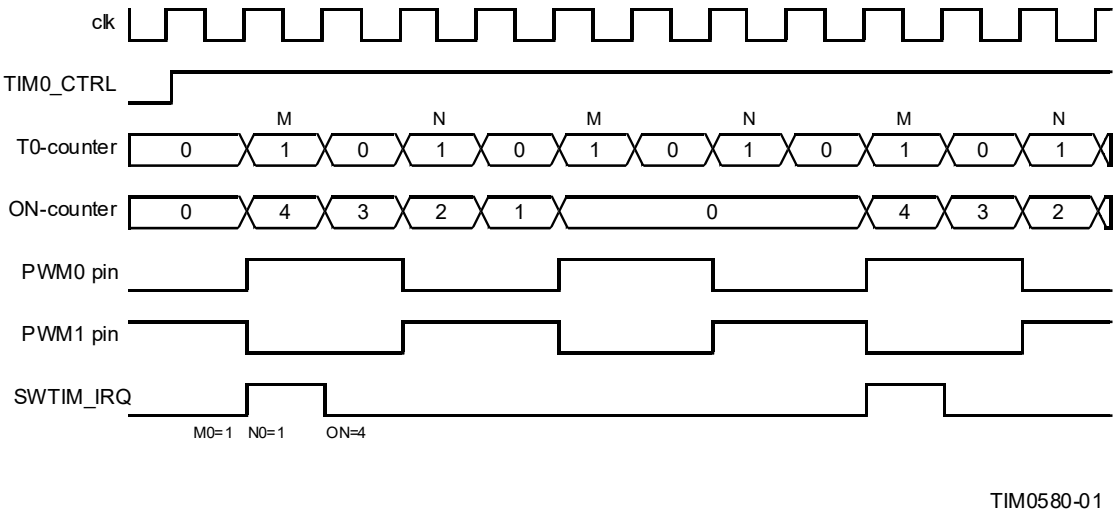


Figure 110. Timer 0 PWM mode

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

The timer input registers, that is, 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 frozen, the timer counters are not decremented. This freezes all the timer registers at their last value. The timer continues its operation again when bit FRZ_SWTIM is cleared via register RESET_FREEZE_REG.

9.20.3 Programming

When LP clock is selected as system clock, CLK_AMBA_REG[HCLK_DIV] should be set to 0.

When LP clock is selected as Timer clock, CLK_PER_REG[TMR_DIV] should be set to 0.

9.20.3.1 Timer Functionality

Timer 0 supports the functionality of a timer for generating interrupts after specific time intervals. To configure the timer operation, follow the steps below:

1. Select the timer clock by programming TIMER0_CTRL_REG[TIM0_CLK_SEL]. The system or the LP clock can be selected using this option.
2. Select the clock division scaler by programming CLK_PER_REG[TMR_DIV]. Note that this setting only applies to the system clock.
3. Define whether Timer 0 uses the clock frequency as is or divided by 10 by programming TIMER0_CLK_REG[TIM0_CLK_DIV].
4. Enable Timer 0 clock by programming CLK_PER_REG[TMR_ENABLE].
5. For 16-bit counting, program TIMER0_ON_REG with the expired time in timer 0 clock cycles TIMER0_RELOAD_M_REG and TIMER0_RELOAD_N_REG must be set to 0.
6. For 17-bit counting, load 65535 to TIMER0_RELOAD_M_REG and the rest to TIMER0_RELOAD_N_REG. TIMER0_ON_REG must be set to 0.
7. Enable Timer 0 by programming TIMER0_CTRL_REG[TIM0_CTRL].

9.20.3.2 PWM Generation

Timer 0 also supports PWM generation. To configure the PWM generation functionality, follow the steps below:

1. Select the timer clock by programming TIMER0_CTRL_REG[TIM0_CLK_SEL]. The system or the LP clock can be selected by this option.

2. Select the clock division scaler by programming CLK_PER_REG[TMR_DIV]. Note that this setting only applies to the system clock.
3. Select the PWM mode by programming the TIMER0_CTRL_REG[PWM_MODE]. There are two modes supported. In the first one, the PWM signals are '1' during high time. In the second one, the PWM signals send out the (fast) clock divided by two during high time, so the clock frequency is in the range of 1 to 8 MHz.
4. Enable Timer 0 clock by programming CLK_PER_REG[TMR_ENABLE].
5. Load the "high" value of the duty cycle in TIMER0_RELOAD_M_REG and the "low" value of the duty cycle in TIMER0_RELOAD_N_REG.
6. Set the desired GPIOs in PWM0/PWM1 and output mode.
7. Enable Timer 0 by programming TIMER0_CTRL_REG[TIM0_CTRL].

9.21 Timer 1

9.21.1 Introduction

Timer 1 is an 11-bit timer that can count up or down. It supports a free-running mode with an interrupt generated when zero is reached (also by wrapping around). It can be configured to use the system clock (sys_clk) or the LP clock (lp_clk) as the clock source. It supports capturing events on two GPIO channels when the number of clock cycles between these events is known. It can also generate an interrupt after a programmable number of clock cycles after an event. Figure 111 shows the block diagram of Timer 1.

Features

- 11-bit up/down counter with free running mode
- Selectable system or LP clock as source
- Two channels for capture input triggered by GPIOs
- Capture capability from two GPIO events with programmable polarity
- Programmable number of events between the two GPIOs for capturing
- Timer 1 or RTC snapshot on capture events
- Interrupt generation.

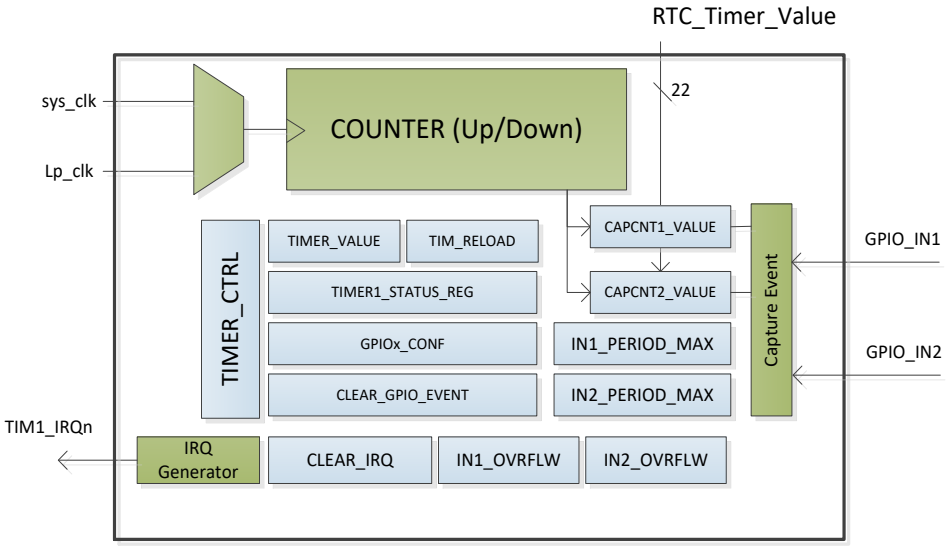


Figure 111. Timer 1 block diagram

9.21.2 Architecture

Timer 1 is placed in the PD_TIM power domain which can be kept powered even when the system power domain (containing the CPU) is shut down.

The main operation of Timer 1 is to count up or down, generating an interrupt when it reaches the maximum/minimum value or the threshold that has been programed as the reload value.

Moreover, Timer 1 comes with a sense block that allows for sensing positive or negative edges on two GPIOs (configurable). The sense block operates in two modes:

- **Counting mode**: Timer1 generates an interrupt upon a configurable amount of edges on a GPIO has been detected. The number of clock cycles was counted between timer start and captured the N - events is stored to CAPCNTx_VALUE register. When timer detects the first N-events, automatically starts to detect the next N-events until timer is disabled
- **Capture mode**: Timer 1 saves a snapshot of either its own counter (11 bits) or the RTC port (22 bits) after an edge on a GPIO has been detected. If there is a pending interrupt, a new snapshot is not saved, and the TIMER1_STATUS_REG[TIMER1_INx_OVRFLW] bit is set. This bit is cleared together with the TIMER1_STATUS_REG[TIMER1_Inx_EVENT] bit

The same GPIO can be used for both modes.

If Timer 1 is used in the counting mode, it can measure the frequency applied to a GPIO port (see section 9.21.3.3).

9.21.3 Programming

When LP clock is selected as system clock, CLK_AMBA_REG[HCLK_DIV] should be set to 0.

9.21.3.1 Timer Functionality

Timer 1 supports the functionality of a timer for generating interrupts after specific time intervals. To configure the timer functionality, follow the steps below:

1. Select the timer clock by programming TIMER1_CTRL_REG[TIMER1_USE_SYS_CLK]. The system or the LP clock can be selected by this option.

NOTE
If the LP clock is selected as system clock, the CLK_AMBA_REG[HCLK] bit field should always be 0 to ensure the proper operation of the timer.

2. Enable or disable the free run mode by programming TIMER1_CTRL_REG[TIMER1_FREE_RUN_MODE_EN]. The free run mode can only be used when Timer 1 counts up.
3. Enable Timer 1 interrupt by programming TIMER1_CTRL_REG[TIMER1_IRQ_EN].
4. Set Timer 1 to count up or down by programming TIMER1_CTRL_REG[TIMER1_COUNT_DOWN_EN].
5. Specify the reload value by programming TIMER1_CTRL_REG[TIMER1_RELOAD].
6. Enable the Timer 1 clock by programming TIMER1_CTRL_REG[TIMER1_CLK_EN].
7. Enable Timer 1 by programming TIMER1_CTRL_REG[TIMER1_ENABLE].

9.21.3.2 Capture Functionality

Timer 1 can capture a snapshot of the value of RTC or Timer1 counts after a GPIO edge is detected. To configure the capture functionality, follow the steps below:

1. Depending on the source of the snapshot value, configure and enable RTC or Timer 1 or both in the capture mode.
2. Set the edge type (rising or falling edge) by programming TIMER1_CAPTURE_REG[TIMER1_IN1_EVENT_FALL_EN] or TIMER1_CAPTURE_REG[TIMER1_IN2_EVENT_FALL_EN], depending on the channel that is used.
3. Set the timer in capture mode by setting TIMER1_CAPTURE_REG[TIMER1_IN1_COUNT_EN]=0 or TIMER1_CAPTURE_REG[TIMER1_IN2_COUNT_EN]=0, depending on the channel that is used.
4. Enable capture interrupt by setting TIMER1_CAPTURE_REG[TIMER1_IN1_IRQ_EN] = 1 or TIMER1_CAPTURE_REG[TIMER1_IN2_IRQ_EN] = 1, depending on the channel that is used.
5. Set the source of the snapshot value (RTC or Timer 1 count) by setting TIMER1_CAPTURE_REG[TIMER1_IN1_STAMP_TYPE] or TIMER1_CAPTURE_REG[TIMER1_IN2_STAMP_TYPE], depending on the channel that is used.
6. Set the GPIO that is used to trigger the capture by setting TIMER1_CAPTURE_REG[TIMER1_GPIO1_CONF] or TIMER1_CAPTURE_REG[TIMER1_GPIO1_CONF], depending on the channel that is used.
   Note that the values from 1 to 12 define the P0 pins from 0 to 11.
7. When an interrupt is generated, the capture value is saved in TIMER1_CAPCNT1_VALUE_REG[TIMER1_CAPCNT1_VALUE] or TIMER1_CAPCNT2_VALUE_REG[TIMER1_CAPCNT2_VALUE], depending on the channel that is used.
8. Write 1 to TIMER1_CLR_EVENT_REG[TIMER_CLR_Inx_EVENT] to clear the event.

9.21.3.3 Frequency Measuring Functionality

Timer 1 can measure the frequency applied to a GPIO port. To configure the frequency measure functionality, follow the steps below:

1. Configure and enable Timer 1 in count up free mode using the system clock.

2. Set Timer 1 in count mode by setting TIMER1_CAPTURE_REG[TIMER1_IN1_COUNT_EN] = 1 or TIMER1_CAPTURE_REG[TIMER1_IN2_COUNT_EN] = 1, depending on the channel that is used.
3. Enable capture interrupt by setting TIMER1_CAPTURE_REG[TIMER1_IN1_IRQ_EN] = 1 or TIMER1_CAPTURE_REG[TIMER1_IN2_IRQ_EN] = 1, depending on the channel that is used.
4. Set the rising edge by programming TIMER1_CAPTURE_REG[TIMER1_IN1_EVENT_FALL_EN] = 0 or TIMER1_CAPTURE_REG[TIMER1_IN2_EVENT_FALL_EN] = 0, depending on the channel that is used.
5. Set the number of periods plus one, in which Timer 1 counts, by setting TIMER1_CAPTURE_REG[TIMER1_IN1_PERIOD_MAX] or TIMER1_CAPTURE_REG[TIMER1_IN2_PERIOD_MAX], depending on the channel that is used.
6. Set the GPIO that is used to trigger the capture by setting TIMER1_CAPTURE_REG[TIMER1_GPIO1_CONF] or TIMER1_CAPTURE_REG[TIMER1_GPIO1_CONF], depending on the channel that is used.
   Note that the values from 1 to 12 define the P0 pins from 0 to 11.
7. After the interrupt is triggered, read the value in TIMER1_CAPCNT1_VALUE_REG[TIMER1_CAPCNT1_VALUE] or TIMER1_CAPCNT2_VALUE_REG[TIMER1_CAPCNT2_VALUE], depending on the channel that is used. This value indicates the number of cycles that have passed during the period defined in step 5.
8. To calculate the frequency applied to the GPIO, divide the number of periods (step 5) by the cycles (step 7) and multiply the result with the frequency of Timer 1 clock.
9. Write 1 to TIMER1_CLR_EVENT_REG[TIMER_CLR_Inx_EVENT] to clear the event.

9.22 Timer 2

9.22.1 Introduction

Timer 2 is basically a PWM generator. It has six PWM outputs. The block diagram is shown in Figure 112.

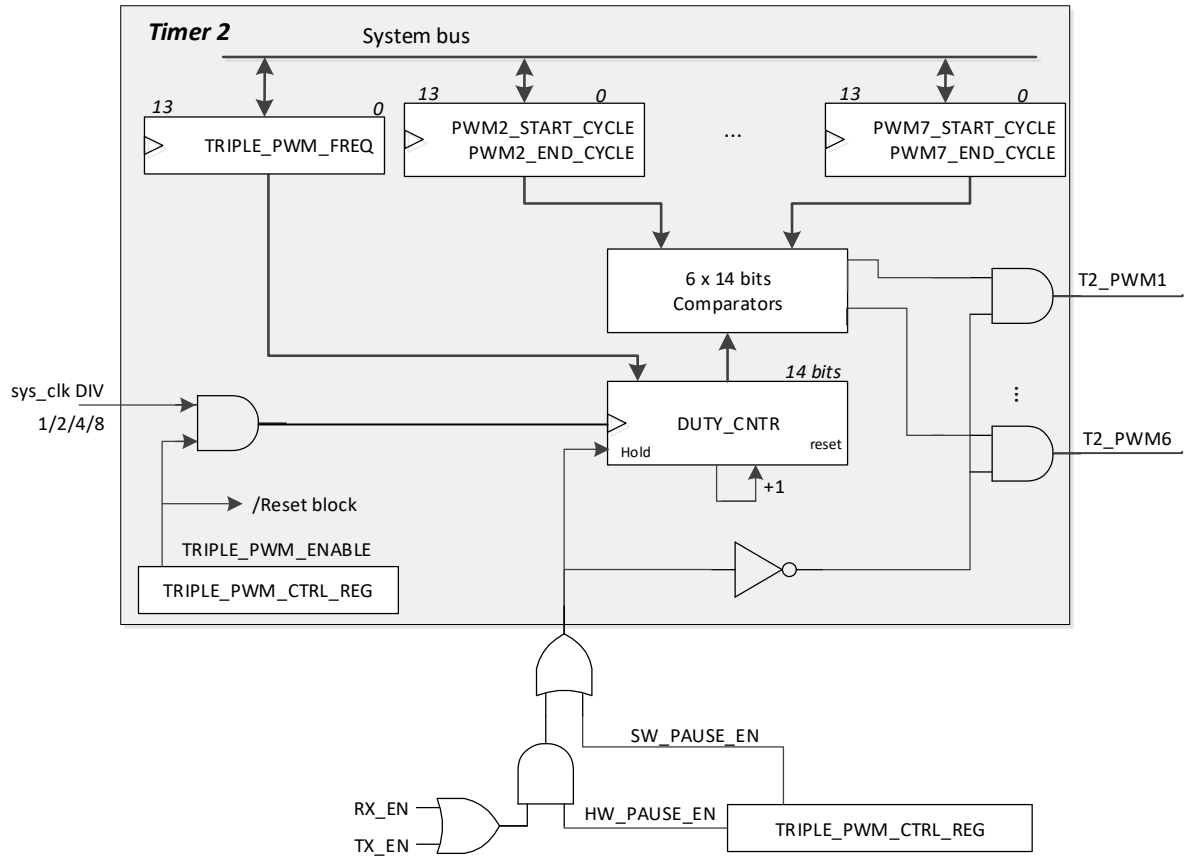


Figure 112.Timer 2 block diagram

Features

- 14-bit general purpose timer
- Ability to generate six PWM signals (PWM2, PWM3, PWM4, PWM5, PWM6, and PWM7,)
- Input clock frequency (f_{IN}) with N = 1, 2, 4, or 8 and sys_clk = 16 MHz or 32 kHz:

$$f_{IN} = \frac{sys\_clk}{N}$$

- Programmable output frequency (f_{OUT}):

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

- Six outputs with a programmable duty cycle from 0% to 100%
- Used for white LED intensity (on/off) control or motor control.

9.22.2 Architecture

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].

TRIPLE_PWM_FREQUENCY determines the output frequency of the PWM outputs.

NOTE
There is a single frequency register for all six PWM outputs.

DUTY_CNTR is an up-counter counting from 0 up to TRIPLE_PWM_FREQUENCY.

If DUTY_CNTR is equal to the value stored in the respective PWMn_END_CYCLE register, it resets the PWMn output to 0.

If DUTY_CNTR is equal to the value stored in the respective PWMn_START_CYCLE register, it sets the PWMn output to 1.

Note that the value of PWMn_END_CYCLE and PWMn_START_CYCLE must be less than or equal to 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 113.

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, DUTY_CNTR is frozen and PWMx output is switched to 0 to disable the selected PWMn. As soon as the Radio is idle, that is, RX_EN or TX_EN signals are zero, 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.

The RX_EN and TX_EN signals are not software driven but controlled by the Bluetooth LE core hardware.

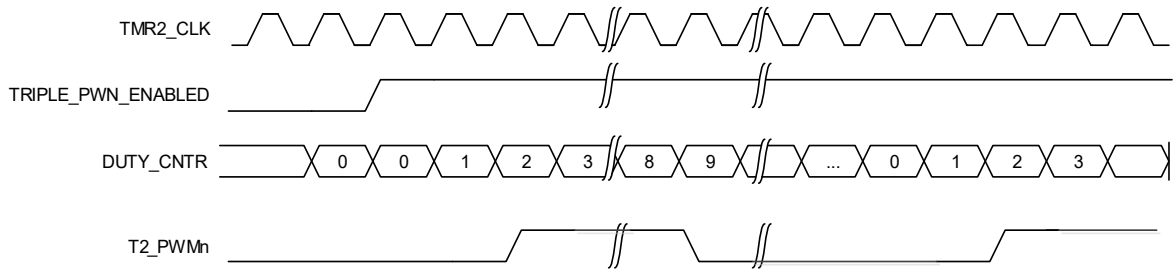


Figure 113. Timer 2 timing diagram

9.22.3 Programming

When LP clock is selected as system clock, CLK_AMBA_REG[HCLK_DIV] should be set to 0.

When LP clock is selected as Timer clock, CLK_PER_REG[TMR_DIV] should be set to 0.

9.22.3.1 PWM Generation

Timer 2 only supports PWM generation and does not support a normal, interrupt generating, timer functionality as the previous timers do. To configure the PWM generation functionality, follow the steps below:

1. Select the clock source for Timer 2 by programming TRIPLE_PWM_CTRL_REG[TRIPLE_PWM_CLK_SEL]. System clock or LP clock can be selected. Please note that if the (fast) system clock is selected, the division scaler is the same as the one for Timer 0.
2. Define the GPIOs to which the PWM signals are mapped by programming the respective PID number.
3. Define the frequency of Timer 2 that feeds the PWM waveforms by programming the TRIPLE_PWM_FREQUENCY with a value that conforms to the following equation. For example, if Timer 2 clock is 32000 Hz (lp_clk) and the required frequency for the PWM is 16 kHz, this register should be written with 0x1.

$$\text{Timer2\_clk\_freq\_Hz}/\text{Required\_freq\_Hz} - 1 \quad (4)$$

NOTE
There is a single frequency register for all six PWM outputs.

4. Define the duty cycle of each PWM signal. Program the start and end cycle of the pulse at PWMx_START_CYCLE and PWMx_END_CYCLE, respectively. The available amount of cycles is depicted in the contents of TRIPLE_PWM_FREQUENCY register. For example, if the TRIPLE_PWM_FREQUENCY has a value of 0x8 and the START/END_CYCLE bit fields have a value of 3 and 5, respectively, the PWM signals rise after three Timer 2 clock cycles and fall after five clock cycles. Every PWM signal has its own register to configure its duty cycle.
5. Enable the PWM signals by programming TRIPLE_PWM_CTRL_REG[TRIPLE_PWM_ENABLE] = 1.

9.22.3.2 Freeze Functionality

There is a provision to allow hardware to pause PWM signals while RF is active. This can be done by programming TRIPLE_PWM_CTRL_REG[HW_PAUSE_EN] = 1. It can also be done via software control by programming TRIPLE_PWM_CTRL_REG[SW_PAUSE_EN] = 1.

9.23 Watchdog Timer

9.23.1 Introduction

The Watchdog Timer is an 8-bit timer with a 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 (WDOG reset) or a Non-Maskable Interrupt (NMI). Figure 114 shows the block diagram of the Watchdog Timer.

Features

- 8-bit down counter with a 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.

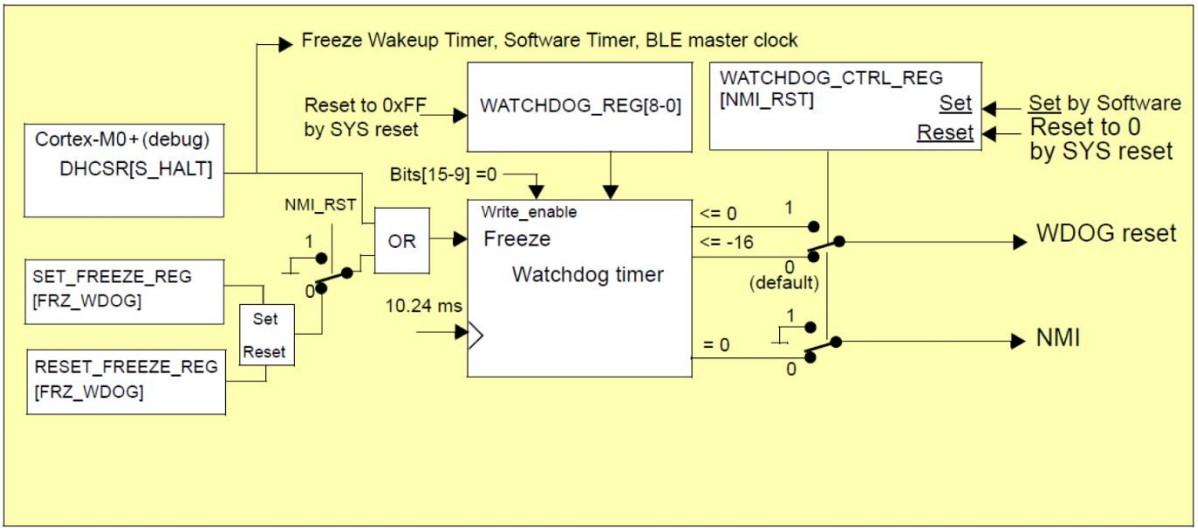


Figure 114. Watchdog timer block diagram

9.23.2 Architecture

The 8-bit 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] bit must be 0. This provides extra filtering for a software run-away by writing ones to all the bits in the WATCHDOG_REG. If the watchdog timer reaches 0, its value gets a negative value by setting bit 8. The counter sequence becomes 1, 0, $1FF_{16}$ (-1), $1FE_{16}$(-2), till $1F0_{16}$ (-16).

If WATCHDOG_CTRL_REG[NMI_RST] = 0, the watchdog timer generates an NMI when it reaches 0 and a WDOG reset when it becomes less or equal to -16 ($1F0_{16}$). The NMI handler must write a value that is larger than -16 to the WATCHDOG_REG to prevent the generation of a WDOG reset when the watchdog timer reaches the value -16 after 16 × 10.24 = 163.8 ms.

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

The WDOG reset is one of the system (SYS) reset sources and resets almost the whole device, including resetting the WATCHDOG_REG register to 255. For an overview of the complete reset circuit and conditions, see Section 9.6.2.1.

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, see Table 42). This is automatically done by the debugging tool, for example, during step-by-step debugging. Note that this bit also freezes the Wake-Up Timer, the Software Timer, and the Bluetooth LE master clock. For additional information see the DEBUG_REG[DEBUGS_FREEZE_EN] mask register. The C_DEBUGEN bit cannot be accessed by the user software so that freezing the watchdog is prevented.

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 is only reset on a SYS reset. Note that if the system is not remapped, that is, the SysRAM is at address 0x07FC0000, a watchdog fire triggers the BootROM code to be executed again.

9.23.3 Programming

To program the Watchdog Timer:

1. Freeze watchdog by setting the SET_FREEZE_REG[FRZ_WDOG] bit (optional).
2. Select NMI and reset events (WATCHDOG_CTRL_REG[NMI_RST]).
3. Enable writing of the watchdog timer (WATCHDOG_REG[WDOG_WEN] = 0]).
4. Write the reload value of the watchdog timer (WATCHDOG_REG[WDOG_VAL], see the register description).
5. Resume watchdog (RESET_FREEZE_REG[FRZ _WDOG] = 1), if frozen.

9.24 Temperature Sensor

The RA6W2 features a built-in temperature sensor.

Features

- Temperature range -40 °C to 105 °C
- Absolute accuracy after one-point calibration +/- 4 °C (assuming 25 °C reference temperature)
- 25 °C single point calibration reference value provided in OTP memory.

9.24.1 Architecture

The temperature sensor can be read out via the GP_ADC.

Figure 115 shows the relationship between the actual ambient temperature and the calculated temperature from the GP_ADC readout, including possible inaccuracies in $T_{SENSE\_ACC\_OTP}$ (offset) and TC_{SENSE} (angle).

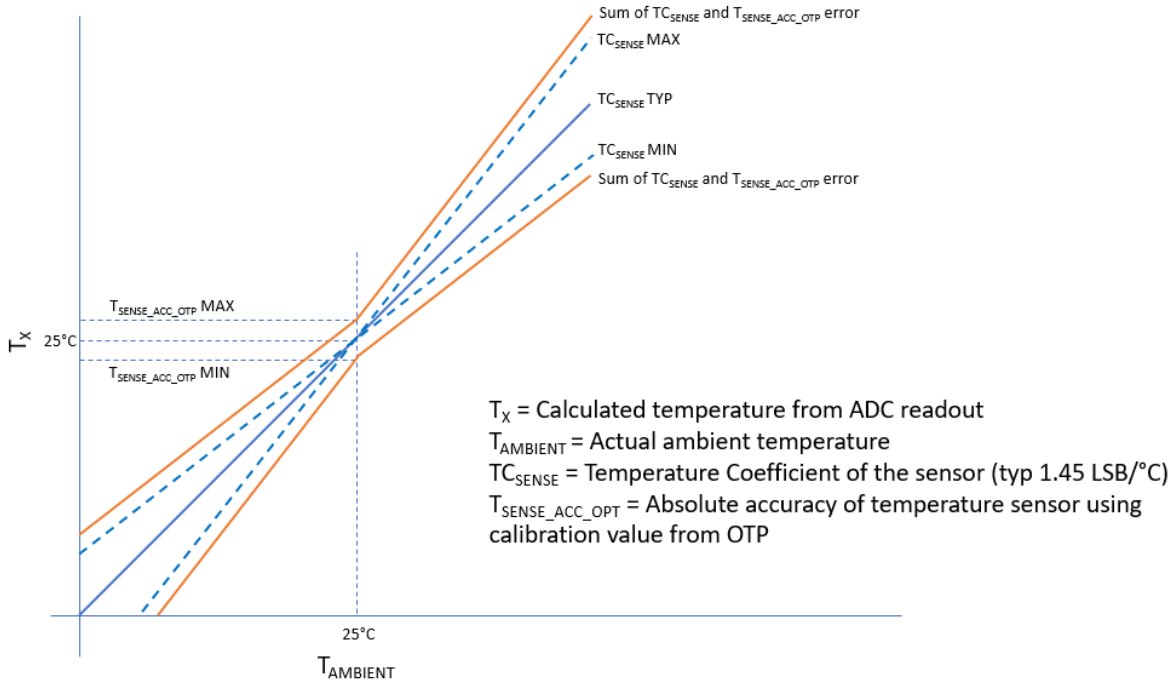


Figure 115. Temperature sensor behavior

The recommended formula for single point calibrated temperature reading is as follows:

$T_x = 25 + (ADC_x - ADC_{OTP\_CAL\_25C})/(TC_{SENSE} \times 64)$

Where:

- T_x = calculated single point calibrated die temperature in [°C]
- ADC_x = 16-bit GP_ADC_VAL readout (converted to decimal) at temperature T_x
- $ADC_{OTP\_CAL\_25C}$ = 25 °C OTP calibration value recorded during production testing (based on the 16-bit readout)
- TC_{SENSE} = temperature coefficient in [LSB/°C], typical value is 1.45 LSB/°C
- 25 = reference base value in [°C]
- 64 = correction for 16-bit to 10-bit ADC values

For uncalibrated temperature sensor measurements, ADC_{OTP_CAL25C} can be replaced by the default value using the formula below:

$T_x = 25 + (ADC_x - 30272)/(TC_{SENSE} \times 64)$

Note that this is not recommended since it can result in large offsets.

NOTE
While measuring and/or calibration, the system's power dissipation should be kept the same, otherwise, the measurement is affected by the internal thermal gradient.

9.24.2 Programming

There is a certain programming sequence required to read the temperature sensor. There are two reading options available:

- Absolute temperature (single-point calibration)
- Relative temperature.

9.24.2.1 Absolute Temperature

A calibration value at 25 °C is stored in OTP for absolute temperature measurements. When the calibration value from OTP is used, the default GP_ADC offset calibration settings should be used.

- To enable OTP in normal read mode:
  - CLK_AMBA_REG[OTP_ENABLE] = 1
  - OTPC_MODE_REG[OTPC_MODE_MODE] = 2
- To read the calibration value at 25°C:
  - Read $ADC_{OTP\_CAL\_25C}$: the content of $ADC_{OTP\_CAL\_25C}$ is at the address 0x7F87F28 of the OTP
- To disable OTP:
  - OTPC_MODE_REG[OTPC_MODE_MODE] = 0
  - CLK_AMBA_REG[OTP_ENABLE] = 0
- To read back the offset calibration:
  - Offp = GP_ADC_OFFP_REG[GP_ADC_OFFP]
  - Offn = GP_ADC_OFFN_REG[GP_ADC_OFFN]

  (Store the data if the original values are needed later for the application)
- To overwrite the defaults (the settings during factory calibration) with the ADC offset values
  - GP_ADC_OFFP_REG[GP_ADC_OFFP] = 200
  - GP_ADC_OFFN_REG[GP_ADC_OFFN] = 200
- To enable the temperature sensor:
  - GP_ADC_CTRL_REG[DIE_TEMP_EN] = 1
- Wait 25 µs for the temperature sensor to start up
- To set the advised ADC settings:
  - GP_ADC_TRIM_REG[GP_ADC_LDO_LEVEL] = 4
  - GP_ADC_CTRL_REG[GP_ADC_CHOP] = 1
  - GP_ADC_CTRL_REG[GP_ADC_SE] = 1
  - GP_ADC_CTRL_REG[GP_ADC_EN] = 1
  - GP_ADC_CTRL2_REG[GP_ADC_I20U] = 1
  - GP_ADC_SEL_REG[GP_ADC_SEL_P] = 4
  - GP_ADC_CTRL2_REG[GP_ADC_STORE_DEL] = 0
- To set sample time and averaging of the ADC sampling
  - GP_ADC_CTRL2_REG[GP_ADC_SMPL_TIME] = F
  - GP_ADC_CTRL2_REG[GP_ADC_CONV_NRS] = 6
- To perform ADC conversion:
  - GP_ADC_CTRL_REG[GP_ADC_START] = 1
- To wait for the conversion to finish, read the register
  - GP_ADC_RESULT_REG[GP_ADC_VAL]
- To write back the original offset values

- GP_ADC_OFFP_REG[GP_ADC_OFFP] = Offp
- GP_ADC_OFFN_REG[GP_ADC_OFFN] = Offn

(Restore the original data if needed by the application)

9.24.2.2 Relative Temperature

For relative temperature measurements, the single-point calibration is not needed. The programming sequence is presented below.

- To enable GP_ADC:
  - GP_ADC_CTRL_REG[DIE_TEMP_EN] = 1
- Wait 25 µs for the temperature sensor to start up
- To set the advised ADC settings:
  - GP_ADC_TRIM_REG[GP_ADC_LDO_LEVEL] = 4
  - GP_ADC_CTRL_REG[GP_ADC_CHOP] = 1
  - GP_ADC_CTRL_REG[GP_ADC_SE] = 1
  - GP_ADC_CTRL_REG[GP_ADC_EN] = 1
  - GP_ADC_CTRL2_REG[GP_ADC_I20U] = 1
  - GP_ADC_SEL_REG[GP_ADC_SEL_P] = 4
  - GP_ADC_CTRL2_REG[GP_ADC_STORE_DEL] = 0
- To set sample time and averaging of the ADC sampling
  - GP_ADC_CTRL2_REG[GP_ADC_SMPL_TIME] = F
  - GP_ADC_CTRL2_REG[GP_ADC_CONV_NRS] = 6
- To perform ADC conversion:
  - GP_ADC_CTRL_REG[GP_ADC_START] = 1
- To wait for the conversion to finish, read the register
  - GP_ADC_RESULT_REG[GP_ADC_VAL]

9.25 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 up to 12 selectable GPIOs (GPIO_IRQx).

Features

- 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.

The block diagram of the Keyboard Controller is presented in the Figure 116.

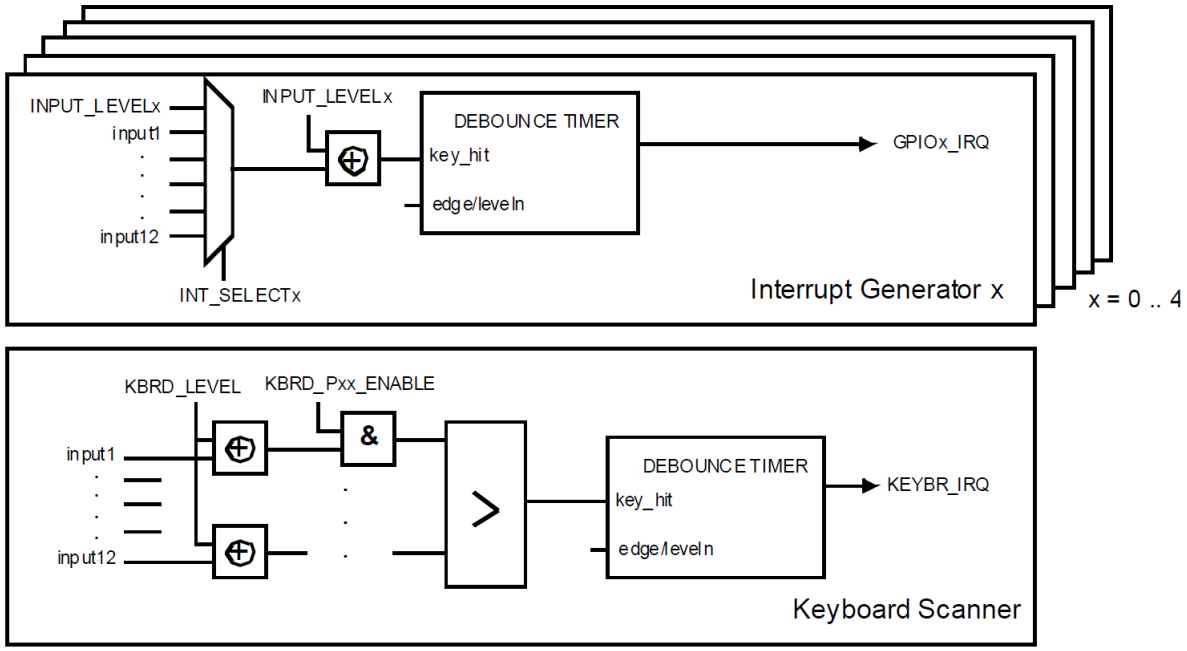


Figure 116. Keyboard controller block diagram

9.25.1 Architecture

9.25.1.1 Keyboard Scanner

A HIGH-to-LOW transition on one of the GPIO inputs sets the internal signal "key_hit" to 1, while KBRD_IRQ_IN_SEL0_REG[KBRD_LEVEL] = 0 and KBRD_IRQ_IN_SELx_REG[KBRD_Pyy_EN] = 1. This signal triggers the state machine of the keyboard interface shown in Figure 117. The debounce timer is loaded with the value of 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 the value of KBRD_IRQ_IN_SEL0_REG[KEY_REPEAT], generating a repeating sequence of interrupts every time when the timer reaches 0.

When the key is released (key_hit = 0) and the 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.

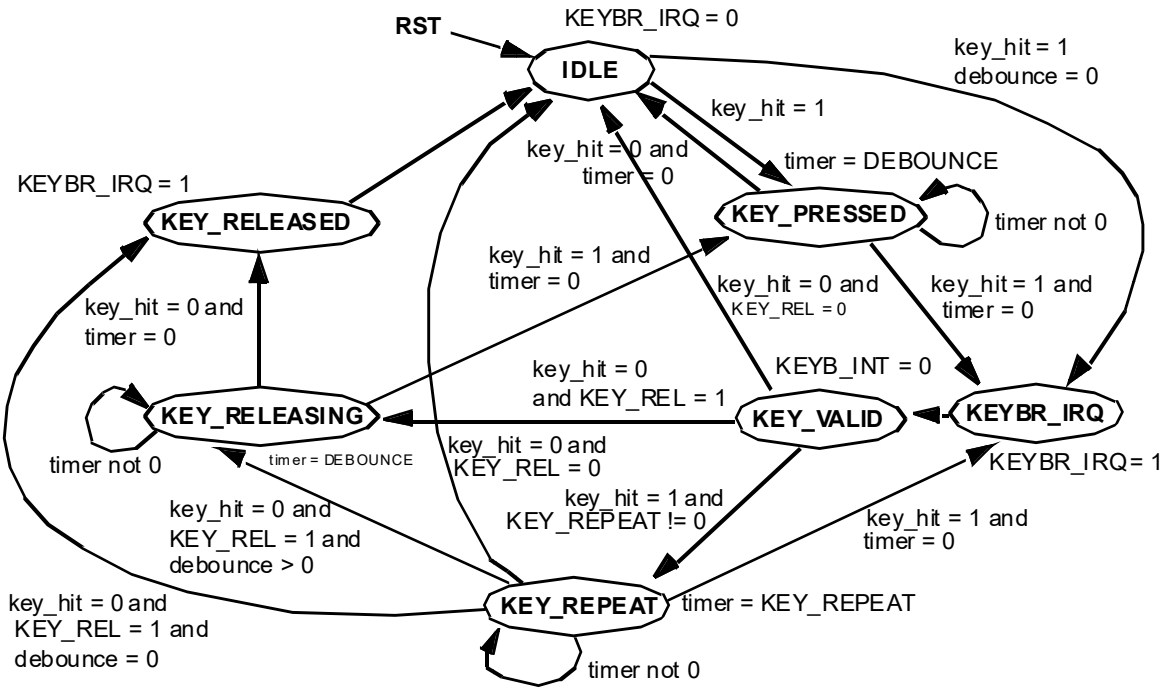


Figure 117. Keyboard scanner state machine

9.25.1.2 GPIO Interrupt Generator

Five identical GPIO interrupt generators support the generation of up to five interrupts (GPIO0_IRQ to GPIO4_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 GPIO inputs sets the internal signal "key_hit" to 1, while the bit INPUT_LEVELx = 0. This signal triggers the state machine of the GPIO Interrupt Generator shown in Figure 118. The debounce timer is loaded with the value of 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 is generated. The debounce timer for each interrupt can be disabled with GPIO_DEBOUNCE_REG[DEB_ENABLEx].

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

To detect both signal edges, the edge polarity INPUT_LEVELx must be inverted in the WAIT_FOR_RELEASE state. This results in "key_hit" = 0 and advances the state machine to the IDLE state, allowing the next inverted edge to be detected.

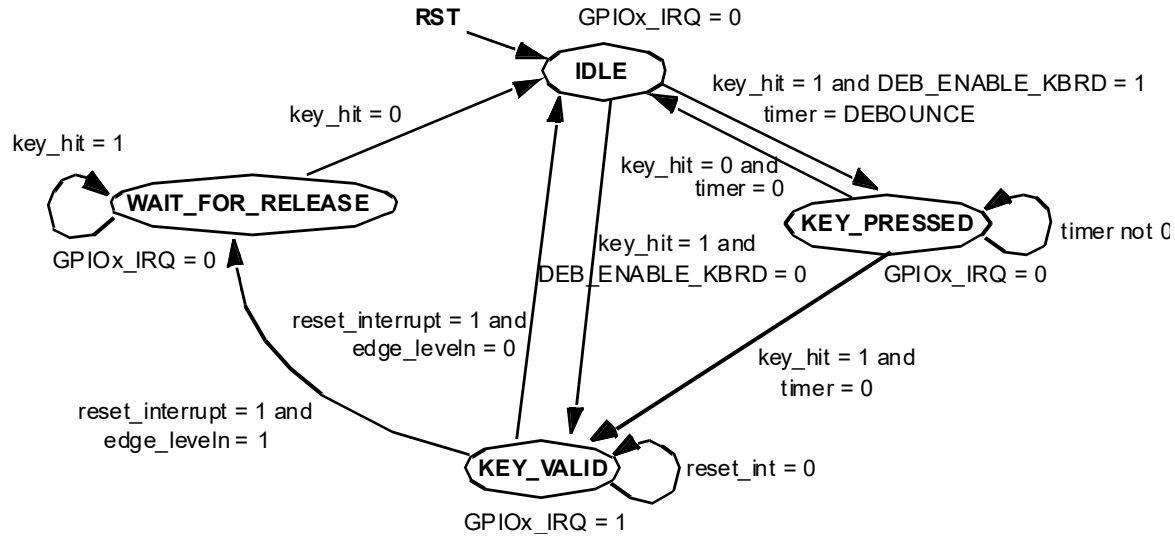


Figure 118. GPIO interrupt generator state machine

9.25.2 Programming

To configure and use the Keyboard controller, follow the steps.

Keyboard Scanner:

1. Enable a keyboard interrupt for the P0_x by setting the KBRD_IRQ_IN_SEL0_REG[KBRD_Px_EN] bit.
2. Select the logic level by which the interrupt is generated (KBRD_CTRL_REG[KBRD_LEVEL]).
3. Select whether a key release also generates an interrupt (KBRD_CTRL_REG[KBRD_REL]).
4. Select whether repeated interrupts is generated when a key is held pressed (KBRD_CTRL_REG[KEY_REPEAT]).
5. Set up the debounce time for each key stroke (GPIO_DEBOUNCE_REG[DEB_VALUE]).
6. Enable the debounce timer (GPIO_DEBOUNCE_REG[DEB_ENABLE_KBRD]).

GPIO Interrupts:

1. Enable a GPIO interrupt for the P0_x by setting the GPIO_IRQx_IN_SEL_REG[KBRD_IRQ0_SEL] bit.
2. Select the logic level by which the interrupt is generated (GPIO_INT_LEVEL_CTRL_REG[INPUT_LEVELx]).
3. Select whether a key release is needed for an interrupt to be generated after a generated IRQ is cleared (GPIO_INT_LEVEL_CTRL_REG[EDGE_LEVELNx]).
4. Set up the debounce time for GPIO trigger (GPIO_DEBOUNCE_REG[DEB_VALUE]).
5. Enable the debounce timer for the selected IRQ (GPIO_DEBOUNCE_REG[DEB_ENABLEx]).

9.26 Input/Output Ports

9.26.1 Introduction

The RA6W2 has an I/O pin assignment that can be configured by the software and is organized into the Port 0. Figure 119 shows the block diagram of the IO and its programmability options.

Features

- Fully programmable pin assignment
- Selectable 25 kΩ pull-up and pull-down resistors per pin
- Programmable driving strength outputs
- Fixed assignment for analog pin ADC[3:0]
- Pins can retain their last state when system enters a Sleep mode.

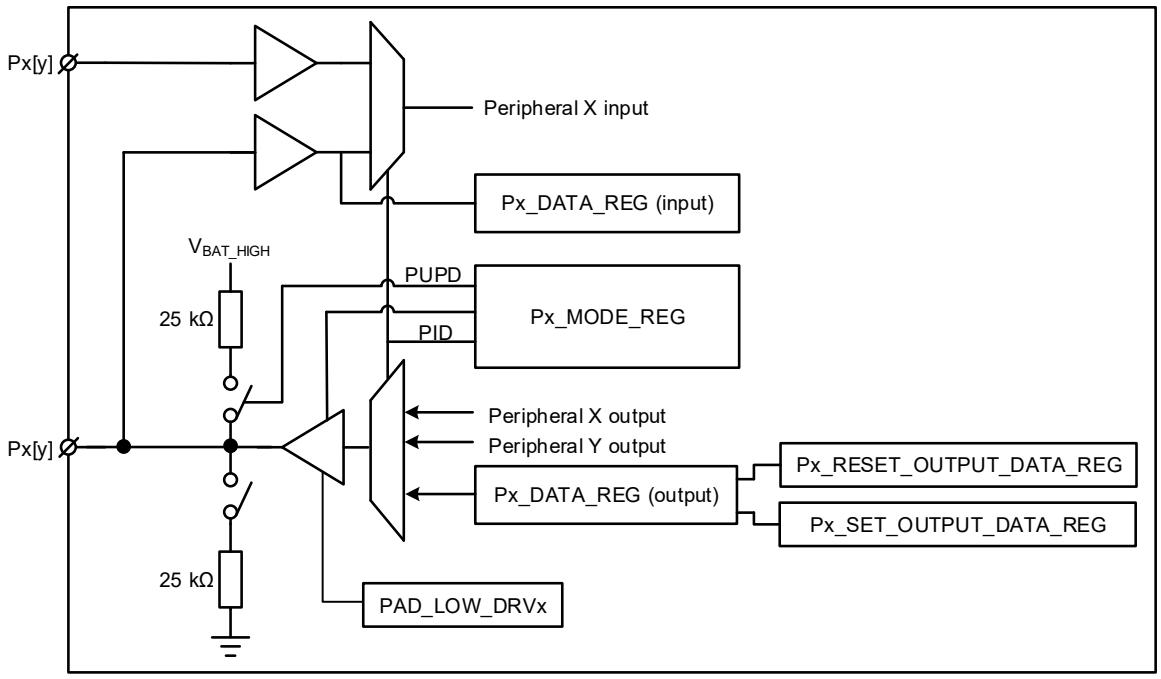


Figure 119. Port P0 with programmable pin assignment and driving strength

9.26.2 Architecture

9.26.2.1 Programmable Pin Assignment

The Programmable Pin Assignment (PPA) provides a multiplexing function to the I/O pins 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 0x1F: Peripheral IO ID (PID)
```

Refer to the registers of Px_MODE_REG (x = 00, 01, 02, to 11) for an overview of the available PIDs. The analog ADC has a fixed pin assignment so that the interference with the digital domain is limited. The SWD interface (JTAG) is mapped on P0_5 (or P0_1 or P0_10, see Section 3) for the SWDIO and P0_2 for the SWCLK.

9.26.2.1.1 Priority

The firmware can assign the same peripheral output to more than one pin. It is the users' responsibility to make a unique assignment.

If more than one input signal is assigned to a peripheral input, the left most pin in the lowest port pin number has priority.

9.26.2.1.2 Direction Control

The port direction is controlled by setting Pxy_MODE_REG[9-8] to:

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

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

9.26.2.2 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.

9.26.2.2.1 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 Cortex 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 this register changes the output value.

9.26.2.2.2 Port Set Data Output Register

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

9.26.2.2.3 Port Reset Data Output Register

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

9.26.2.3 Fixed Assignment Functionality

Certain signals have a fixed mapping on specific general purpose IOs. This assignment is shown in Table 79.

Table 79: Fixed assignment of specific signals

GPIO	Reset/SWD (Note 1)	QUADRATURE DECODER (Note 2)	ADC (Note 3)
P0_0	RST	CH6_A	
P0_1	SWDIO (alternative)	CH1_A	ADC_0
P0_2	SWCLK	CH1_B	ADC_1
P0_3		CH2_A	
P0_4		CH2_B	
P0_5	SWDIO	CH3_A	
P0_6		CH3_B	ADC_2
P0_7		CH4_A	ADC_3
P0_8		CH6_B	
P0_9		CH5_A	
P0_10	SWDIO (alternative)	CH5_B	
P0_11		CH4_B	

Note 1 The SWD signal mapping is defined by SYS_CTRL_REG[DEBUGGER_ENABLE]. However, these signals are mapped on the ports by default. The alternative SWD mapping is selected by the SYS_CTRL_REG[DEBUGGER_ENABLE] bit field. The RST default functionality can be disable by the HWR_CTRL_REG[DISABLE_HWR] bit.

Note 2 The mapping of the quadrature decoder signals on the respective pins is overruled by the QDEC_CTRL2_REG[CHx_PORT_SEL] register.

Note 3 The ADC function can be selected by the PID bit field on the respective Px port.

9.26.2.4 Types of GPIO Pads

There are two different types for the GPIO pads, namely, type A and type B. Their block diagrams are presented in Figure 120 and Figure 121.

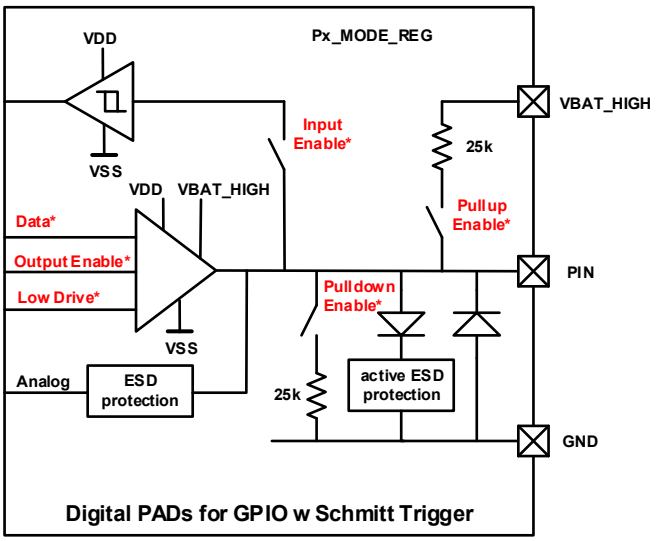


Figure 120. Type A GPIO pad – GPIO with schmitt trigger on input

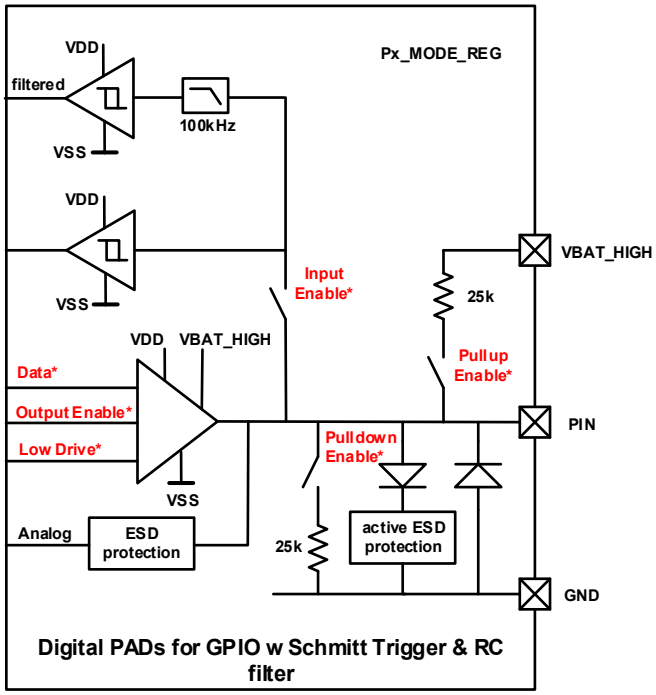


Figure 121. Type B GPIO pad – GPIO with schmitt trigger and RC filter on input

Red signals are latched when the system enters a Sleep mode.

9.26.2.5 Driving Strength

Pads can be configured regarding their driving capability using PAD_WEAK_CTRL_REG. There are only 2 levels available for the load that the pad can support, namely normal = 3.5 mA typical, and reduced = 0.35 mA typical.

9.27 General Purpose ADC

The RA6W2 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 0.9 V, which represents the full-scale reference voltage. Figure 122 shows the block diagram of the GPADC.

Features

- 10-bit dynamic ADC with 125 ns typical conversion time
- Maximum sampling rate 1 Msample/s
- 128× averaging; conversion time 1 ms, up to 11b ENOB
- Ultra-low power (20 µA typical supply current at 100 ksample/s)
- Four single-ended or two differential external input channels (GPIOs)
- Battery, DC-DC outputs, and the internal V_{DD} monitoring channels
- Chopper function
- Offset adjust
- Common-mode input level adjust
- Configurable attenuator: 1×, 2×, 3× and 4×
- Input shifter.

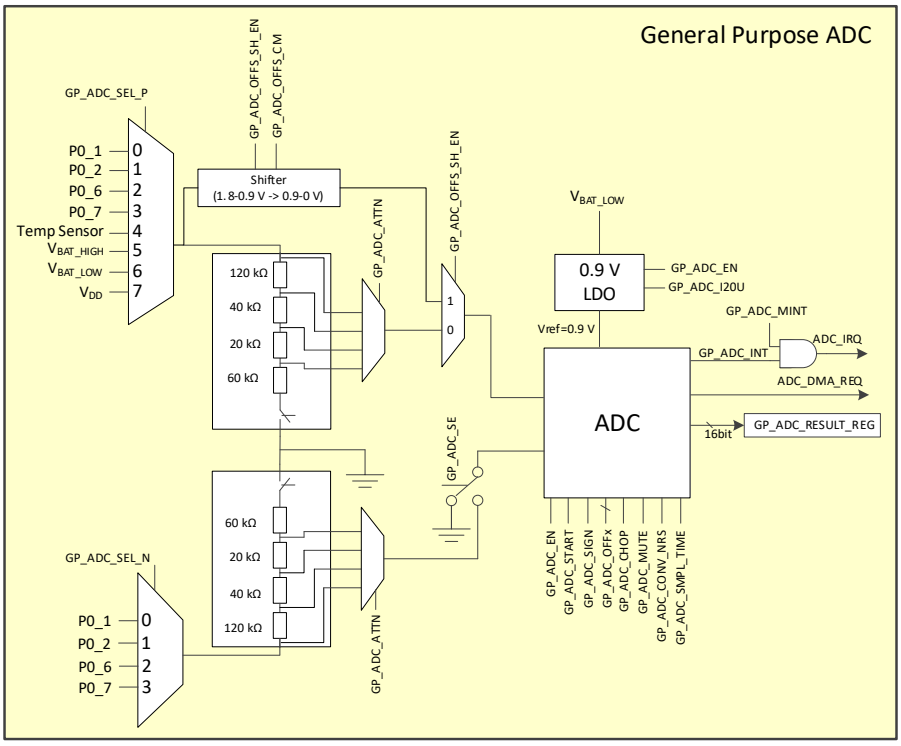


Figure 122. Block diagram of GPADC

9.27.1 Architecture

The ADC architecture shown in Figure 122 has the following subblocks:

- Analog to Digital converter (ADC):
  - ADC analog part internally clocked with 100 MHz.
  - ADC logic part clocked with the ADC_CLK which is the 16 MHz system clock (sys_clk).
- 0.9 V LDO for the ADC supply with a high PSRR enabled with GP_ADC_CTRL_REG[GP_ADC_EN].
- Configurable attenuator with 1×, 2×, 3×, and 4× attenuation controlled by GP_ADC_CTRL2_REG[GP_ADC_ATTN].

- Input shifter which shifts the battery voltage range from 0.85 V to 1.75 V (with a common mode adjustment) to the ADC input range from 0 V to 0.9 V controlled by GP_ADC_CTRL2_REG[GP_ADC_OFFS_SH_EN] and GP_ADC_CTRL2_REG[GP_ADC_OFFS_CM].
- APB Bus interface clocked with the APB clock. Control and status registers are available through registers GP_ADC_*.
- Maskable Interrupt (ADC_IRQ) and DMA request (ADC_DMA_REQ).
- ADC input channel selector. Up to four GPIO ports, the battery and DC-DC output (V_{BAT_HIGH} and V_{BAT_LOW}), the internal V_{DD}, and the analog ground level (AVS) can be measured.

9.27.1.1 Input Channels

Table 80 summarizes the ADC input channels. The GPIO signals at the channels [3:0] can be monitored both single-ended and differentially. The signals at the 4-7 inputs can be monitored single-ended or differentially with respect to the GPIOs.

Table 80: ADC input channels

Channel	Signal	Description
3:0	GPIO [P0_1, P0_,2, P0_6, P0_7]	General Purpose Inputs
4	Temperature Sensor	Temperature Sensor
5	V_{BAT_HIGH}	V_{BAT_HIGH} rail
6	V_{BAT_LOW}	V_{BAT_LOW} rail
7	V_{DD}	V_{DD} rail for the digital power domain

Table 81 summarizes the voltage ranges which can be handled with the single-ended or differential operation for different attenuation values. The single-ended/differential mode is controlled by the bit GP_ADC_CTRL_REG[GP_ADC_SE], and the attenuation is handled by the bit GP_ADC_CTRL2_REG[GP_ADC_ATTN].

Table 81: GPADC external input channels and voltage range

GP_ADC_ATTN	GP_ADC_SE	Input scale	Input limits
0 (1 ×)	0	-0.9 V to +0.9 V	-1 V to +1 V
	1	0 V to +0.9 V	-0.1 V to 1V
1 (2 ×)	0	-1.8 V to +1.8 V	-1.9 V to +1.9 V
	1	0 V to +1.8 V	-0.1 V to 1.9 V
2 (3 ×)	0	-2.7 V to +2.7 V	-2.8 V to +2.8 V
	1	0 V to +2.7 V	-0.1 V to 2.8 V
3 (4 ×)	0	-3.6 V to +3.6 V	-3.45 V to +3.45 V
	1	0 V to +3.6 V	-0.1 V to 3.45 V

9.27.1.2 Operating Modes

The GPADC operation flow diagram is shown in Figure 123.

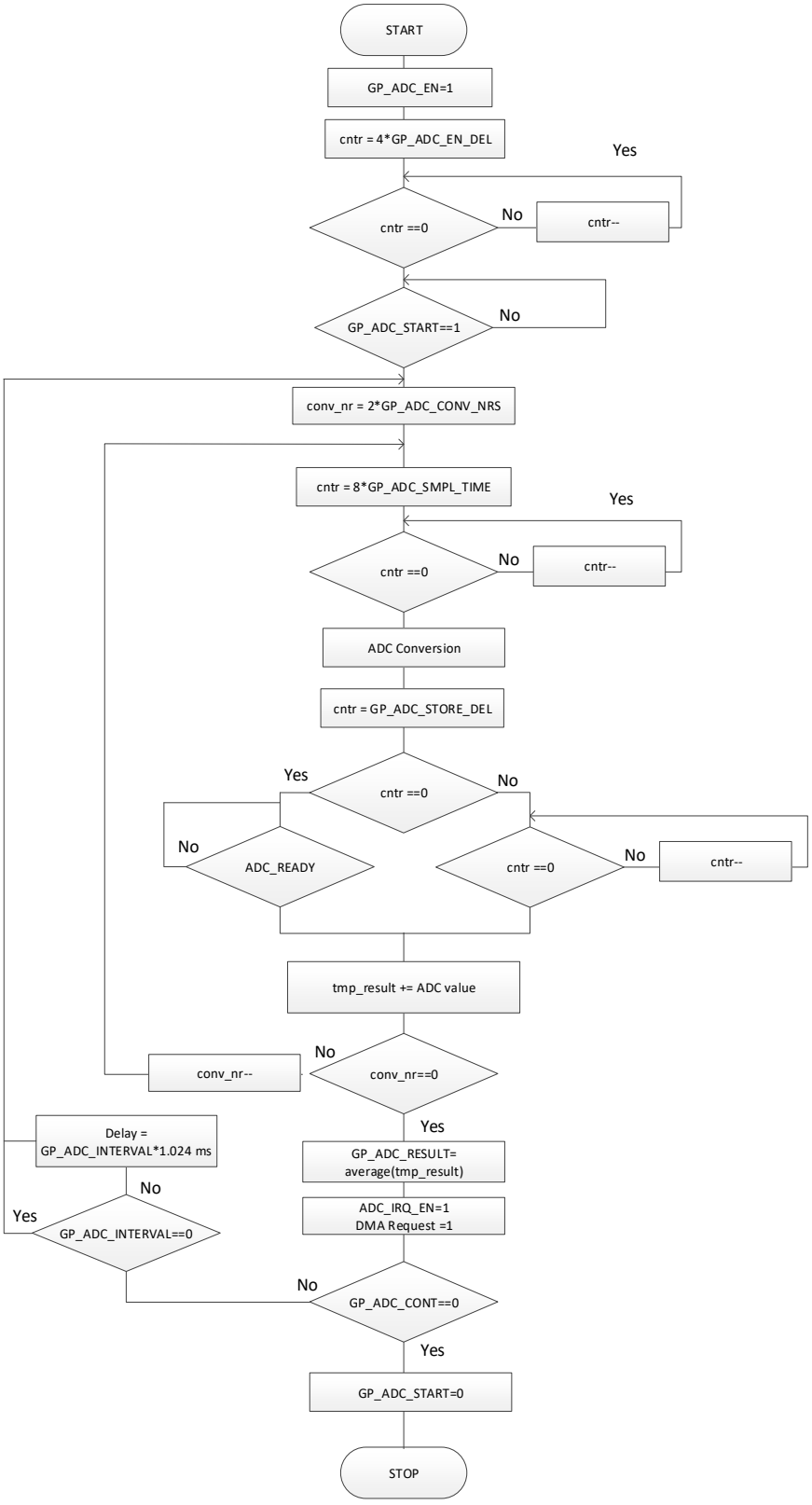


Figure 123. GPADC operation flow diagram

9.27.1.3 Enabling the ADC

Enabling/disabling of the ADC is triggered by configuring bit GP_ADC_CTRL_REG[GP_ADC_EN]. When the bit is set to 1, first the LDO is enabled. Then after the delay value set in GP_ADC_CTRL3_REG[GP_ADC_EN_DEL] (typically 16 µs to account for the LDO settling time), the ADC is enabled, and an AD conversion can be started. See Table 82 for recommended values.

Table 82: ADC_LDO start-up delay

f_{ADC_CLK}	GP_ADC_EN_DEL	$T_{ADC\_EN\_DEL}$
16 MHz	0x40	16 µs

Formula:

GP_ADC_EN_DEL = $T_{ADC\_EN\_DEL} \times f_{ADC\_CLK} / 4$

This value must be rounded up to the nearest integer.

The GPADC is a dynamic ADC and consumes no static power, except for the **ADC_LDO** which consumes approximately 20 µA. Therefore, GP_ADC_EN must be set to 0 if the ADC is not used.

9.27.1.3.1 Manual Mode

An AD conversion can be started by setting GP_ADC_START to 1. While a conversion is active, GP_ADC_START remains 1. When a conversion is finished, the hardware sets GP_ADC_START to 0 and GP_ADC_INT to 1 (interrupt), and GP_ADC_RESULT_REG contains the valid ADC value. While a conversion is active, writing 1 to GP_ADC_START does not start a new conversion. software should always check that bit GP_ADC_START = 0 before starting a new conversion.

9.27.1.3.2 Continuous Mode

Setting GP_ADC_CTRL_REG[GP_ADC_CONT] to 1 enables the continuous mode, which automatically starts a new AD conversion when the current conversion has been completed. The GP_ADC_START bit is only needed once to trigger the first conversion. If the continuous mode is active, GP_ADC_RESULT_REG always contains the latest ADC value.

To correctly terminate the continuous mode, it is required to disable the GP_ADC_CONT bit first and then wait until the GP_ADC_START bit is cleared to 0, so the ADC is in a defined state.

NOTE
Before making any changes to the ADC settings, users must disable the continuous mode by setting bit GP_ADC_CONT to 0 and waiting until bit GP_ADC_START = 0.

At full speed the ADC consumes approximately 50 to 60 µA. If the data rate is less than 100 ksample/s, the current consumption is in the order of 25 µA.

The time interval between two successive AD conversions is programmable with GP_ADC_CTRL3_REG[GP_ADC_INTERVAL] in steps of 1.024 ms. If GP_ADC_INTERVAL = 0, the conversion restarts immediately. If GP_ADC_INTERVAL is not zero, the ADC first synchronizes to the delay clock before starting the conversion. This can take up to 1 ms.

9.27.1.3.3 Conversion Modes

AD Conversion

Each AD conversion has three phases:

- Sampling
- Conversion
- Storage

The AD conversion starts with the sampling phase. This phase ends after the time set in GP_ADC_CTRL2_REG[GP_ADC_SMPL_TIME] and triggers the conversion phase. If GP_ADC_CTRL2_REG[GP_ADC_STORE_DEL] = 0, handshaking is used, that is, the ADC result is stored when a conversion is finished. Otherwise, a fixed (programmable) delay is used, and the result is stored regardless of whether the conversion is finished or not.

The total conversion time of an AD conversion depends on various settings. In short, it is as follows.

$$T_{ADC} = \frac{N_{CONV} \cdot \left(N_{CYCL\_SMPL} + N_{CYLC\_STORE}\right)}{f_{ADC\_CLK}} \quad (5)$$

Where

- NCONV = the number of conversions. This is related to the value programmed in GP_ADC_CTRL2_REG[GP_ADC_CONV_NRS], following 2GP_ADC_CONV_NRS. When GP_ADC_CTRL_REG[GP_ADC_CHOP] is set, the minimum value for NCONV is always 2.
- NCYLC_SMPL = the number of ADC_CLK cycles used for sampling, which is 8 × GP_ADC_CTRL2_REG[GP_ADC_SMPL_TIME].
- NCYCL_STORE = the number of ADC_CLK cycles until the result is stored. When GP_ADC_CTRL2_REG[GP_ADC_STORE_DEL] = 0, handshaking is used. With handshaking, the number of ADC_CLK cycles is typically three. This value may spread from sample to sample and over temperature, otherwise the number of ADC_CLK cycles is GP_ADC_CTRL2_REG[GP_ADC_STORE_DEL] + 1.

Sampling Phase

The sampling time can be programmed via GP_ADC_CTRL2_REG[GP_ADC_SMPL_TIME] and depends on the sampling time constant in combination with the desired sampling accuracy. This sampling time constant, T_{ADC_SMPL} (Table 83), then depends on the output impedance of the source, the internal resistive dividers, and the internal sampling capacitor. And the number of required time constants is given by the natural logarithm of the desired accuracy, that is, ln(2^NBIT). For NBIT = 10-bit accuracy, 7-time constants are required.

Table 83: ADC sampling time constant (T_{ADC_SMPL})

ADC Input	T_{ADC_SMPL}
GPADC0, GPADC1 (GP_ADC_ATTN = 0)	R_{OUT} × 0.5 pF (Differential Input) R_{OUT} × 1 pF (Single-Ended Input)
GPADC0, GPADC1 (GP_ADC_ATTN = 1)	(R_{OUT} + 120 kΩ) × 0.5 pF (Differential Input) (R_{OUT} + 120 kΩ) × 1 pF (Single-Ended Input)

Formula:

$GP\_ADC\_SMPL\_TIME = \ln(2^{NBIT}) \times T_{ADC\_SMPL} \times f_{ADC\_CLK} / 8$

This value must be rounded up to the nearest integer.

Conversion and Storage Phase

One AD conversion typically takes around 125 ns with a 100 MHz clock. The result can be stored either by handshaking or after a fixed number of cycles (programmable).

- Handshake mode (GP_ADC_STORE_DEL = 0):

  In handshake mode the conversion result is available in GP_ADC_RESULT_REG after two sampling ADC_CLK cycles plus two conversion ADC_CLK cycles plus two ADC_CLK cycles for synchronization.
- Fixed delay mode (GP_ADC_STORE_DEL > 0):

  In fixed delay mode the conversion result is available in GP_ADC_RESULT_REG after the programmed storage delay, regardless of whether the conversion is ready or not. Note that when the delay is too short (that is, the conversion is not finished in the allocated time), the old (previous) ADC result is stored.

Averaging

To reduce noise and improve performance, multiple samples can be averaged out (assuming the time average of noise equals zero). This is handled by hardware and can be controlled by setting GP_ADC_CTRL2_REG[GP_ADC_CONV_NRS] to a non-zero value. The actual number of the consecutive samples taken is by $2^{GP\_ADC\_CONV\_NRS}$.

Because the internal noise also acts as a form of dither, the actual accuracy can be improved. Therefore, the ADC result is not truncated to 10-bit but stored as 16-bit left aligned, and truncation is left for you. The expected Effective Number of Bits (ENOB) is shown in Table 84.

Table 84: ENOB in Oversampling mode

GP_ADC_CONV_NRS	ENOB (Left Aligned) in GP_ADC_RESULT_REG
0	> 9
1	> 9
2	> 9
3	> 10

GP_ADC_CONV_NRS	ENOB (Left Aligned) in GP_ADC_RESULT_REG
4	> 10
5	> 10
6	> 11
7	> 11

Chopper Mode

Inherently, the ADC has a DC offset (EOFS). When GP_ADC_CTRL_REG[GP_ADC_CHOP] is set to 1, the hardware triggers two consecutive AD conversions and flips the sign of the offset in-between. Summing the two samples effectively cancels out the inherent ADC offset. This method also smooths other non-ideal effects and is recommended for DC and the slowly changing signals.

When combined with averaging, every other AD conversion is taken with opposite sign. Without averaging two AD conversions are always triggered.

Note that a DC offset causes saturation effects at zero scale or full scale. When chopping is used without offset calibration, non-linear behavior is introduced towards zero scale and full scale.

Additional Settings

The hardware also supports pre-ADC attenuation via GP_ADC_CTRL2_REG[GP_ADC_ATTN]:

- Setting 0 disables the attenuator
- Setting 1 scales the input range by a factor of two
- Setting 2 scales the input range by a factor of three
- Setting 3 scales the input range by a factor of four.

With bit GP_ADC_CTRL_REG[GP_ADC_MUTE] = 1, the input is connected to 0.5 × ADC reference. So, the ideal ADC result should be 511.5. Any deviation from this is the ADC offset.

With bit GP_ADC_CTRL_REG[GP_ADC_SIGN] = 1, the sign of the offset is inverted. When chopper is used, the hardware alternates GP_ADC_SIGN = 0 and 1. This bit is typically only used for the offset calibration routine and has no specific use to the end user.

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 (E_G) of the GPADC affects the effective input range. The offset error (E_{OFS}) causes the effective input scale to become non-centered. The offset error can be reduced by chopping and/or by offset calibration.

The ADC result also includes some noise. If the input signal itself is noise free (inductive effects included), the average noise level is ±1 LSB. Reducing noise effects can be done by taking more samples and calculating the average value. This can be done by programming GP_ADC_CTRL2_REG[GP_ADC_CONV_NRS] to a non-zero value.

With a “perfect” input signal (for example, 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 RA6W2 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]. Bit GP_ADC_I20U enables a constant 20 µA load current at the regulator output so that the current does not drop to zero. This, obviously, increases power consumption by 20 µA.

Offset Calibration

A relatively high offset error (E_{OFS}, up to 30 mV, so approximately 30 LSB) is caused by a very small dynamic comparator. This offset error can be cancelled with the chopping function, but it still causes unwanted saturation effects at zero scale or full scale. With GP_ADC_OFFP_REG and GP_ADC_OFFN_REG, the offset error can be compensated in the ADC network itself. To calibrate the ADC, follow the steps in Table 85. In this routine, 0x200 is the target mid-scale of the ADC.

Table 85: GPADC calibration procedure for Single-Ended and Differential modes

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

To increase the accuracy, it is recommended to set the GP_ADC_CTRL2_REG[GP_ADC_SMPL_TIME] = 2 or 3 and GP_ADC_CTRL2_REG[GP_ADC_CONV_NRS] = 3 or 4 prior to this routine.

It is recommended to implement the above calibration routine during the initialization phase of RA6W2. To verify the calibration results, check whether the GP_ADC_RESULT value is close to 0x200 while bit GP_ADC_CTRL_REG[GP_ADC_MUTE] = 1.

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.

Common Mode Adjustment

The common mode level of the differential signal must be 0.45 V = Full Scale/2 (or 1.35 V with GP_ADC_ATTN = 2, that is, 3× attenuation). If the common mode input level of 0.45 V cannot be achieved, the common mode level of the GPADC can be adjusted via GP_ADC_OFFP_REG and GP_ADC_OFFN_REG according to Table 86. The GPADC can tolerate a common mode margin of up to 50 mV.

Table 86: Common mode adjustment

CM Voltage (V_{ccm})	GP_ADC_OFFP = GP_ADC_OFFN
0.225 V	0x300
0.450 V	0x200
0.675 V	0x100

Any other common mode levels between 0.0 V and 0.9 V can be calculated from Table 86. Offset calibration can be combined with common mode adjustment by replacing the 0x200 value in the offset calibration routine with the value required to get the appropriate common mode level.

Input Impedance, Inductance, and Input Settling

The GPADC has no input buffer stage. During the sampling phase, a capacitor of 0.5 pF in differential mode or 1 pF in single-ended mode is switched to the input line(s). The pre-charge of this capacitor is at midscale level, so the input impedance is infinite.

During the sampling phase, a certain settling time is required. A 10-bit accuracy requires at least seven-time constants τ_{ADC_SMPL}, determined by the output impedance of the input signal source, the internal resistive dividers, and the 0.5 pF or 1 pF sampling capacitor. See Table 83.

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 (single-ended mode) or from pin to pin (differential mode).

9.27.2 Programming

To program and use the GPADC:

1. Enable the GPADC by setting the GP_ADC_CTRL_REG[GP_ADC_EN] bit.
2. Set up the GPIO input (P0_x_MODE_REG[PID] = 15).

3. Select the input channel (GP_ADC_SEL_REG).
4. Select the sampling mode (differential or single ended) by writing the GP_ADC_CTRL_REG[GP_ADC_SE] bit.
5. Select between the manual mode and the continuous mode of sampling (GP_ADC_CTRL_REG[GP_ADC_CONT].
6. Set up extra options (see GP_ADC_CTRLx_REG description)
7. Start the conversion by setting GP_ADC_CTRL_REG[GP_ADC_START] bit.
8. Wait for GP_ADC_CTRL_REG[GP_ADC_START] to become 0 or interrupt being triggered (when used).
9. Clear the ADC interrupt by writing any value to GP_ADC_CLEAR_INT_REG.
10. Get the ADC result from the GP_ADC_RESULT_REG.

9.28 Real Time Clock

9.28.1 Introduction

The RA6W2 Bluetooth LE Subsystem is equipped with a Real Time Clock (RTC) which provides the complete clock and calendar information with automatic time units adjustment and easy configuration.

Features

- Complete time of day clock: 12/24 hour, hours, minutes, seconds, and hundredths of a second
- Calendar function: day of week, date of month, month, year, century, leap year compensation, and year 2000 compliant
- Alarm function: month, date, hour, minute, second, and hundredths of a second
- Event interrupt on any calendar or time unit
- Available during sleep if the power domain PD_TIM is kept alive
- Granularity of 10 ms (RTC_CLK)
- Provides 22 LSB to Timer 1 upon a capture trigger.

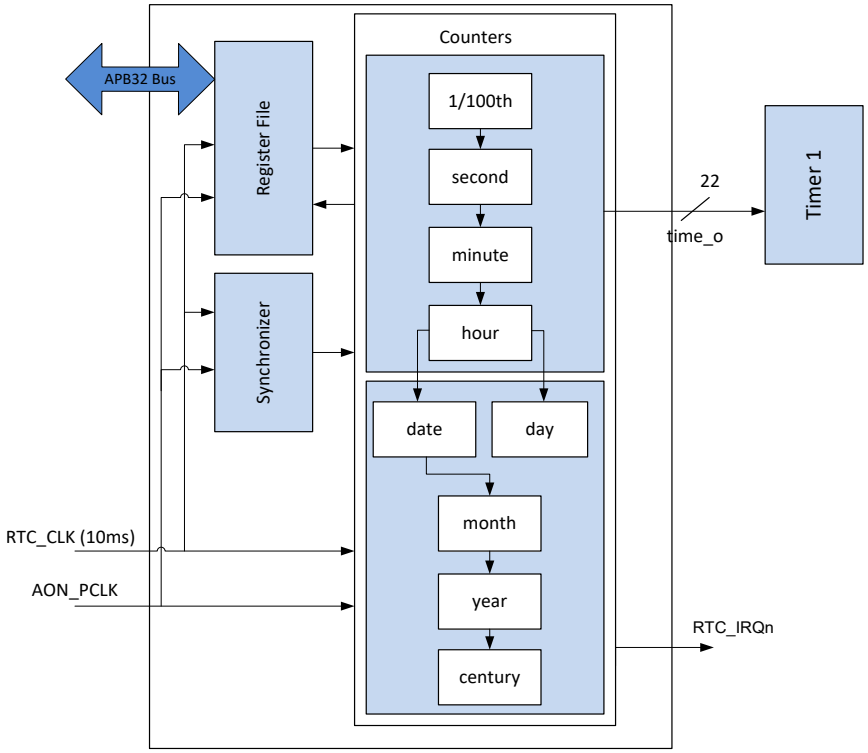


Figure 124. Real time clock block diagram

9.28.2 Architecture

The architecture of the RTC is shown in Figure 124.

The RTC supports a year range from 1900 to 2999 as well as full month, date, minute, second, and hundredth of second ranges. It also supports hour ranges of 0 to 23 (24-hour format) or 1 to 12 with a.m./p.m. flag (12-hour format).

Alarms can be generated in two ways, as a one-time alarm or as a recurring alarm. In addition to alarms, the RTC can detect when a particular event occurs. Each field of the calendar and time counter can generate an event when it rolls over. For example, an event can be generated every new month, new week, new day, new half day (12-hour mode), new minute, or new second. Both alarms and events can generate an interrupt. All the interrupts can be set, enabled, disabled, or masked at any time.

The LSB (22) of the port showing a full of 32-bit information on the current time is latched by Timer 1 (TIMER1_CAPCNT1/2_VALUE_REG) if instructed by Timer 1 configuration. This allows for storing an RTC based snapshot upon an event on a GPIO.

9.28.3 Programming

To configure the RTC:

1. Configure the 100 Hz RTC granularity if needed:
   a. Based on the selected LP clock (for example, 32768 kHz), set the CLK_RTCDIV_REG[RTC_DIV_INT] = 327 (= 0x147).
   This value should be equal to the integer divisor part of the formula
   $F_{LP\_CLK}/100$ = **327**.680.
   b. Based on the selected LP clock (for example, 32768 kHz), set the CLK_RTCDIV_REG[RTC_DIV_FRAC] = 680 (= 0x2A8).
   This value should be equal to the fractional divisor part of the formula
   $F_{LP\_CLK}/100$ = 327.**680**.
   c. To achieve a better accuracy of the divisor, configure the denominator for the fractional division accordingly (CLK_RTCDIV_REG[RTC_DIV_DENOM]).
   d. Enable the 100 Hz RTC granularity by setting the CLK_RTCDIV_REG [RTC_DIV_ENABLE] bit.
2. Enable the time functionality by clearing the RTC_CONTROL_REG[RTC_TIME_DISABLE].
3. Enable the calendar functionality by clearing the RTC_CONTROL_REG[RTC_CAL_DISABLE].
4. Choose between 12-hour or 24-hour mode (RTC_HOUR_MODE_REG[RTC_HMS]).
5. Configure the time (RTC_TIME_REG).
6. Configure the date (RTC_CALENDAR_REG).
7. Set up a time alarm if needed (RTC_ALARM_ENABLE_REG).
8. Set up a calendar alarm if needed (RTC_CALENDAR_ALARM_REG).
9. Enable the configured alarms (RTC_ALARM_ENABLE_REG[RTC_ALARM_xxxx_EN]).
10. Configure the interrupt generation when an alarm happens (RTC_INTERRUPT_ENABLE_REG). Disable the interrupt generation with RTC_INTERRUPT_DISABLE_REG.
11. Configure the event flag generation when an alarm happens (RTC_EVENT_FLAGS_REG).
12. Define whether a software reset resets the RTC (RTC_KEEP_RTC_REG[RTC_KEEP]).

9.29 Power

As discussed in Section 4.2, the integrated power management unit (PMU) comprises the DC-DC converter and various LDOs, the V_{DD} Clamp, and the POR circuitry for Bluetooth, LE functions. The details of these blocks are discussed in the following sections.

The RA6W2 can be used in various system configurations.

NOTE
Boost Power mode is not included in the system configuration. Any mention or explanation of Boost Power mode is intended for debug purposes only.

9.29.1 DC-DC Converter

The RA6W2 can be configured in buck and DC-DC bypass. The integrated part of the DC-DC is the same for all configurations, that is, the black building blocks in Figure 125. The buck configuration is configured on the PCB, distinguished with the red external components in Figure 125. In the bypass configuration the V_{BAT_HIGH} and V_{BAT_LOW} rails are connected, so the DC-DC is bypassed.

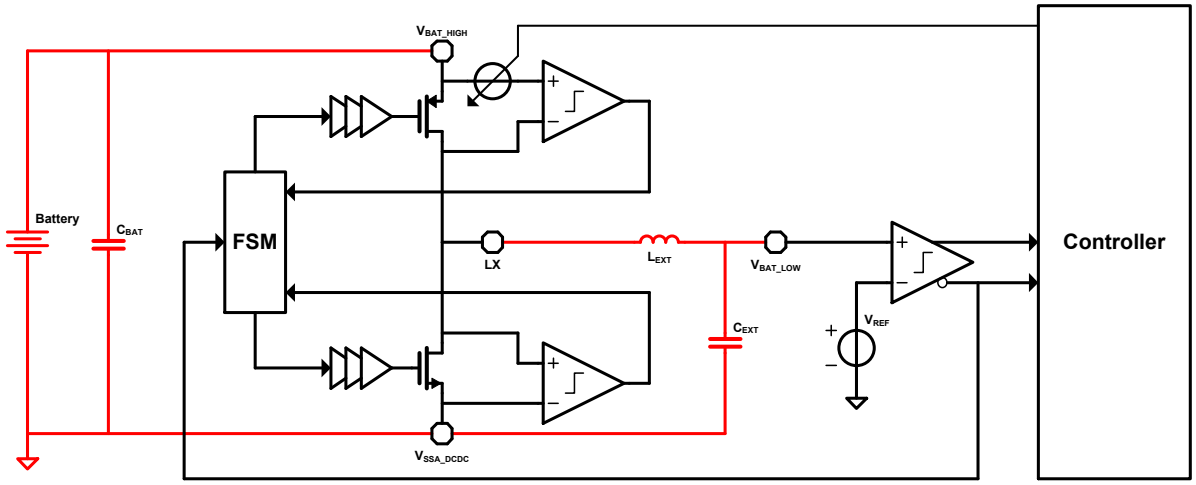


Figure 125. DC-DC block diagram - buck configuration

- In buck configuration the battery is connected to V_{BAT_HIGH}, and DC-DC supplies power to V_{BAT_LOW} rail

The DC-DC level can be programmed by POWER_LEVEL_REG[DCDC_LEVEL] and a fine trimming is available by POWER_LEVEL_REG[DCDC_TRIM].

When the system is allowed to go to sleep, the hardware FSM is activated and the DC-DC controller is automatically turned off. Upon a wake-up, the DC-DC is enabled by programming DCDC_CTRL_REG[DCDC_ENABLE]. If DCDC_ENABLE is kept set before the system goes to sleep, the DC-DC is started by the hardware FSM while waking up. If DCDC_ENABLE is reset before the system goes to sleep, the DC-DC can only be started by software setting this bit after wake-up.

For Buck configuration, a typical DC-DC efficiency at 25 °C as a function of the load current for different battery voltages (V_{BAT} = V_{BAT_HIGH}) is shown in Figure 126.

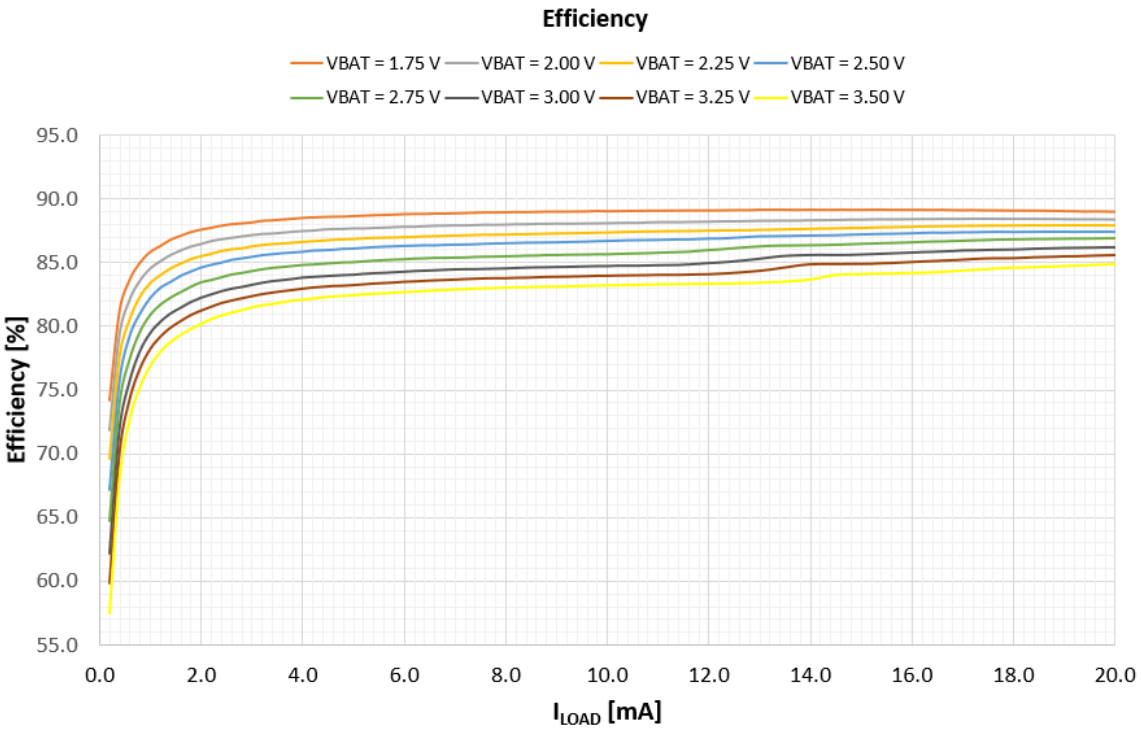


Figure 126. DC-DC efficiency in buck configuration

9.29.2 LDOs

Several LDOs are used in the RA6W2 to provide a stable power supply to the rails and the building blocks.

- V_{DD_Clamp} generates a trimmable ~0.75 V V_{DD} supply voltage for the AON (always on) DCORE power domain from V_{BAT_HIGH} or V_{BAT_LOW} when the system is in the hibernation mode
- L_{DO_LOW} provides power to the V_{BAT_LOW} rail in the buck configuration with a typical output voltage of 1.1 V. This LDO is used during start-up and can also be used after start-up. Alternatively, it can be disabled, and the V_{BAT_LOW} rail can be supplied by the DC-DC converter. The LDO has a low power setting which is used to maintain the V_{BAT_LOW} rail during sleep mode. See Section 4.2.3 for more details.
- L_{DO_CORE} supplies the internal V_{DD} from V_{BAT_LOW}. In the active mode it generates 0.9 V and in the Sleep mode 0.75 V
- LDOs for the RF and the analog building blocks generate 0.9 V when the particular blocks are active. When the blocks are switched off, the LDOs are disabled.

9.29.3 POR Circuit

The POR_LOW circuit issues a POR when the VBAT_LOW voltage is below the threshold voltage V_{IL} for more than 50 µs. The POR is cleared when the battery voltage is above V_{IH} for at least 25 µs. The threshold levels of the POR circuit are summarized in Section 3.12.

The POR_HIGH circuit issues a POR when the VBAT_HIGH voltage is below the V_{IL} for more than 50 µs. The POR is cleared when the battery voltage is above V_{IH} for at least 25 µs. The threshold levels of the POR circuit are summarized in Section 3.12.

9.30 Core

The Bluetooth Low Energy core used in the RA6W2 Bluetooth LE subsystem is a qualified Bluetooth 5.1 baseband controller compatible with the Bluetooth LE specification and it is in charge of packet encoding/decoding and frame scheduling.

The block diagram of Bluetooth LE core is presented in Figure 127.

Features

- Compliant with Bluetooth Core Specification, v5.1, Bluetooth SIG
  - Dual topology
  - Low duty cycle advertising
  - L2CAP connection-oriented channels
- All device classes support (Broadcaster, Central, Observer, and Peripheral)
- All packet types (Advertising, Data, and Control)
- Dedicated Encryption (AES/CCM)
- Bit stream processing (CRC and Whitening)
- FDMA/TDMA/events formatting and synchronization
- Frequency hopping calculation
- Operating clock 16 MHz or 8 MHz
- Low power modes supporting 32.0 kHz, 32.768 kHz, or 15 kHz
- Supports powerdown of the baseband during the protocol's idle periods.

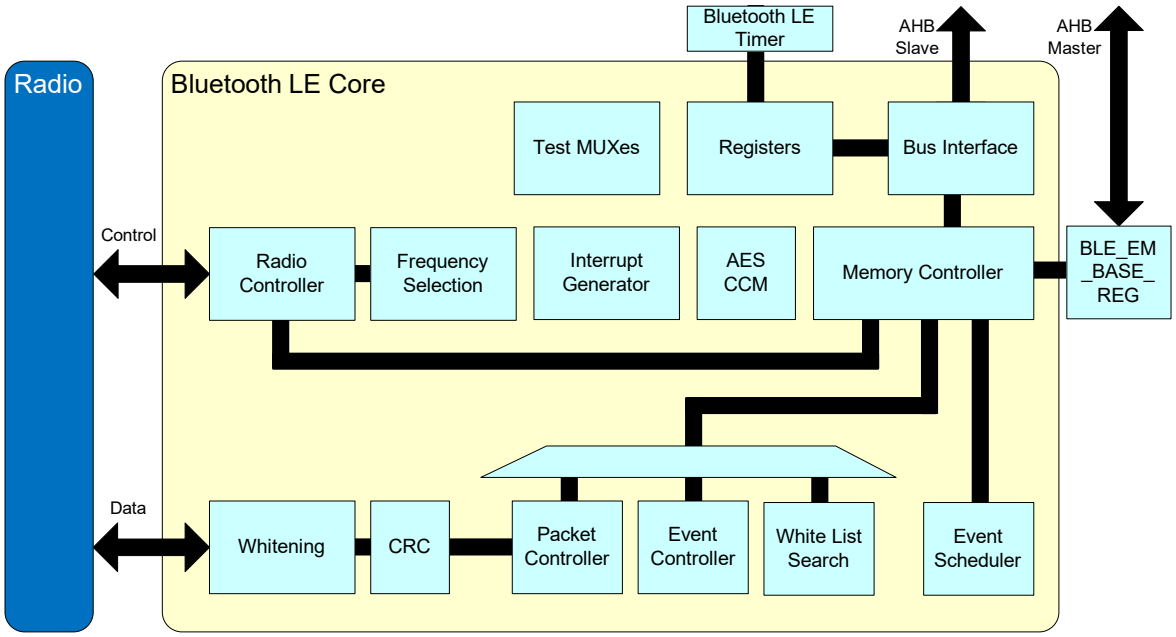


Figure 127. Bluetooth LE Core block diagram

9.30.1 Architecture

The Bluetooth LE 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 base address of the Exchange Memory is programmable by means of the BLE_EM_BASE register.

9.30.2 Programming

9.30.2.1 Wake-Up IRQ

When the Bluetooth LE core switches to "Bluetooth LE Deep Sleep mode", the only way to correctly exit from this state is by generating initially 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 "Bluetooth LE Deep Sleep mode", that is,

regardless of whether the Bluetooth LE Timer has expired or Bluetooth LE 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. A detailed description is as follows:

- **TWIRQ_SET:** it defines the number of "ble_lp_clk" cycles before the expiration of the Bluetooth LE Timer when the BLE_WAKEUP_LP_IRQ must be asserted. It is recommended to select a TWIRQ_SET value larger than the amount of time that is required to finish trimming the XTAL 32 MHz (refer to XTAL32M_TRIM_READY) plus the execution time of the IRQ Handler. If the programmed value of TWIRQ_SET is less than the minimum recommended value, the system wakes up but the actual Bluetooth LE sleep duration (see BLE_DEEPSLSTAT_REG) is larger than the programmed sleep duration (refer to BLE_DEEPSLWKUP_REG)
- **TWIRQ_RESET:** it defines the number of "ble_lp_clk" cycles before the expiration of the sleep period when the BLE_WAKEUP_LP_IRQ is de-asserted. It is recommended to always set its value to "1"
- **TWEXT:** it determines the high period of BLE_WAKEUP_LP_IRQ, if an external wake-up event (refer to GP_CONTROL_REG[BLE_WAKEUP_REQ]) occurs. Its minimum value is "TWIRQ_RESET + X", where X is the number of "ble_lp_clk" clock cycles that BLE_WAKEUP_LP_IRQ is held high. The 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 is asserted
- **Minimum Bluetooth LE Sleep Duration:** The minimum value of BLE_DEEPSLWKUP_REG[DEEPSLTIME] bit, measured in "ble_lp_clk" cycles, is the higher value between (a) "TWIRQ_SET + 1" and (b) the software execution time from setting BLE_DEEPSLCNTL_REG[DEEP_SLEEP_ON] up to preparing CPU to accept the BLE_WAKEUP_LP_IRQ (for example, to call the Cortex instruction WFI). If the programmed DEEPSLTIME is less than the minimum value of BLE_DEEPSLWKUP_REG[DEEPSLTIME], the BLE_WAKEUP_LP_IRQ Handler may execute sooner than the call of the Cortex WFI instruction in the example and cause software instability.

9.30.2.2 Switch from Bluetooth LE Active Mode to Bluetooth LE Deep Sleep Mode

Software can set the Bluetooth LE core into the "Bluetooth LE Deep Sleep mode" by first programming the timing of BLE_WAKEUP_LP_IRQ generation, then programming the desired sleep duration at BLE_DEEPSLWKUP_REG, and finally set the register bit BLE_DEEPSLCNTL_REG[DEEP_SLEEP_ON].

During the "Bluetooth LE Deep Sleep mode", the Bluetooth LE Core switches to the "ble_lp_clk" (15 kHz, 32.0 kHz, or 32.768 kHz) 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, software must disable the Bluetooth LE clocks ("ble_master1_clk", "ble_master2_clk", and "ble_crypt_clk") by setting the CLK_RADIO_REG[BLE_ENABLE] register bit to "0".

Finally, software 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 128 shows the waveforms when the Bluetooth LE Deep Sleep mode is entered. In this case, as soon as the software detects that RADIO_PWRDOWN_ALLOW is "1", it sets the PMU_CTRL_REG[RADIO_SLEEP] to power down the Radio Subsystem. In Figure 128, Figure 129, Figure 130, Figure 131, and Figure 132, the corresponding Bluetooth LE Core signals are marked with red while Radio Subsystem is in power-down state and they remain red-marked during the period when RADIO_SLEEP is set.

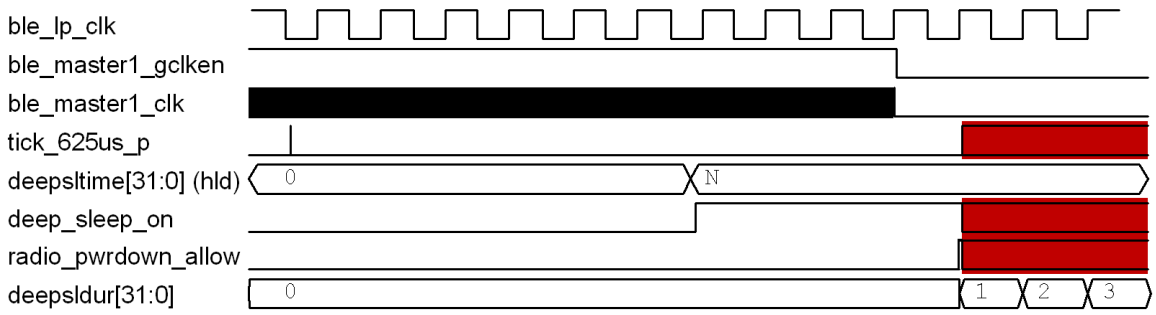


Figure 128. Entering Bluetooth LE Deep Sleep mode

9.30.2.3 Switch from Bluetooth LE Deep Sleep Mode to Bluetooth LE Active Mode

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

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

9.30.2.4 Switching at an Anchor Point

Figure 131 shows a typical Bluetooth LE deep sleep phase that is terminated at a predetermined time. After a configurable time before the scheduled wake-up time (configured via BLE_ENBPRESET_REG register bit fields), the Bluetooth LE Timer asserts the BLE_WAKEUP_LP_IRQ to wake up the CPU (powering up the System Power Domain). The BLE_WAKEUP_LP_IRQ Interrupt Handler prepares the code environment and the XTAL32M oscillator stabilization (refer to SYS_STAT_REG[XTAL32_SETTLED]) and decides when the Bluetooth LE Core is ready to exit the Bluetooth LE Deep Sleep mode.

When the software decides that the Bluetooth LE Core can wake up, it must enable the Bluetooth LE clocks (via CLK_RADIO_REG[BLE_ENABLE]) and power up the Radio Power Domain (see PMU_CTRL_REG[RADIO_SLEEP] and SYS_STAT_REG[RAD_IS_UP]).

After the sleep period expires (as specified in BLE_DEEPSLWKUP_REG[DEEPSLTIME]), the Bluetooth LE Timer does not exit the Bluetooth LE Deep Sleep mode until it detects that the Bluetooth LE Core is powered up. That means, if the software requires more time to power up the Bluetooth LE Core, the final sleep duration (provided by BLE_DEEPSLSTAT_REG) is longer than the preprogrammed value.

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

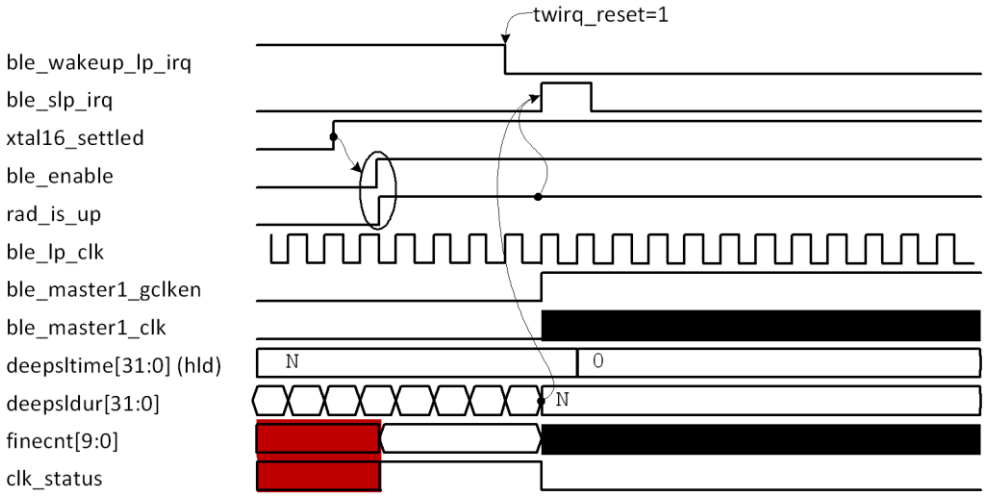


Figure 129. Exit Bluetooth LE Deep Sleep mode at predetermined time (zoom in)

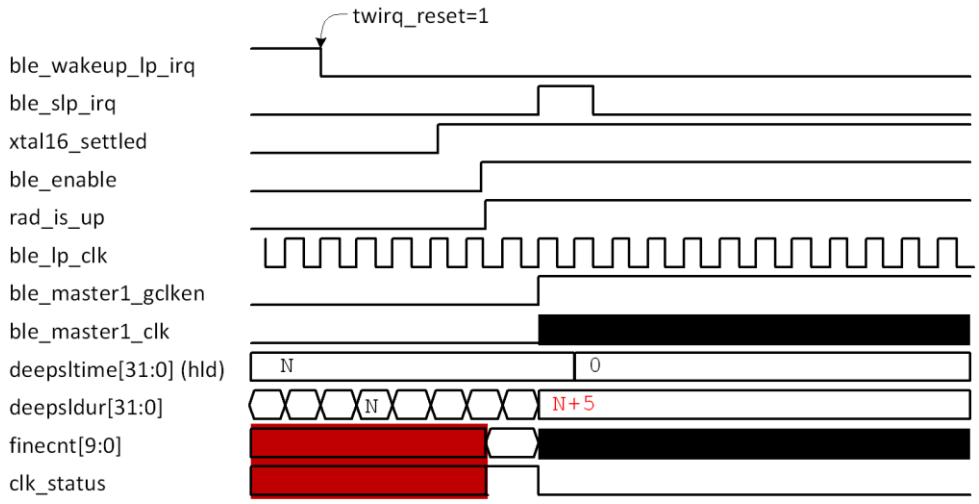


Figure 130. Exit Bluetooth LE Deep Sleep mode after predetermined time (zoom in)

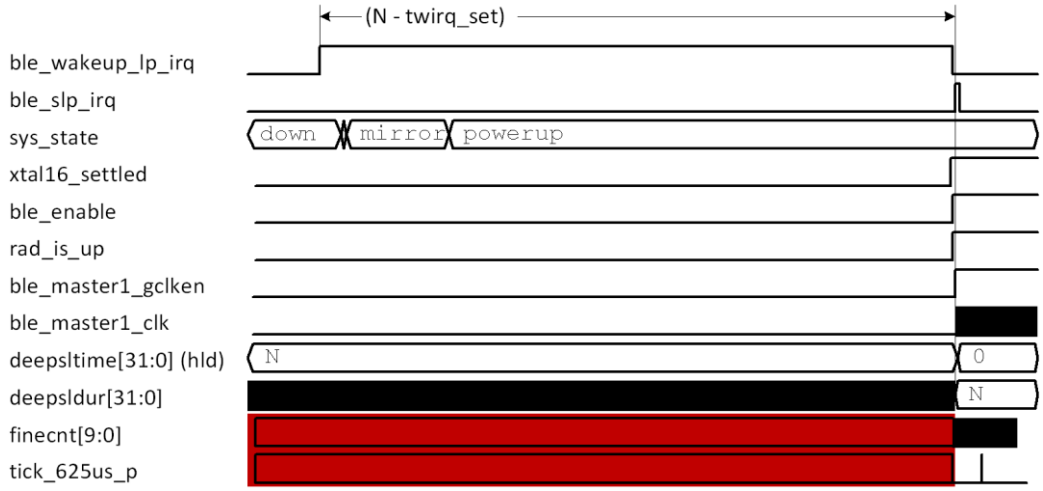


Figure 131. Exit Bluetooth LE Deep Sleep mode at predetermined time (zoom out)

9.30.2.5 Switching Due to an External Event

Figure 132 shows a wake-up from a Bluetooth LE deep sleep period forced by the assertion of register bit GP_CONTROL_REG[BLE_WAKEUP_REQ].

Assume that the system is in Extended Sleep state with all power domains switched off and both the wake-up timer and wake-up controller 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 software decides to wake up the Bluetooth LE core, then it sets the GP_CONTROL_REG[BLE_WAKEUP_REQ] to 1 to force the wake-up sequence.

In Figure 132 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 XTAL32_TRIM_READY, causing both BLE_WAKEUP_LP_IRQ and BLE_SLP_IRQ Handlers to be executed sequentially. The decision depends on the software structure and the application.

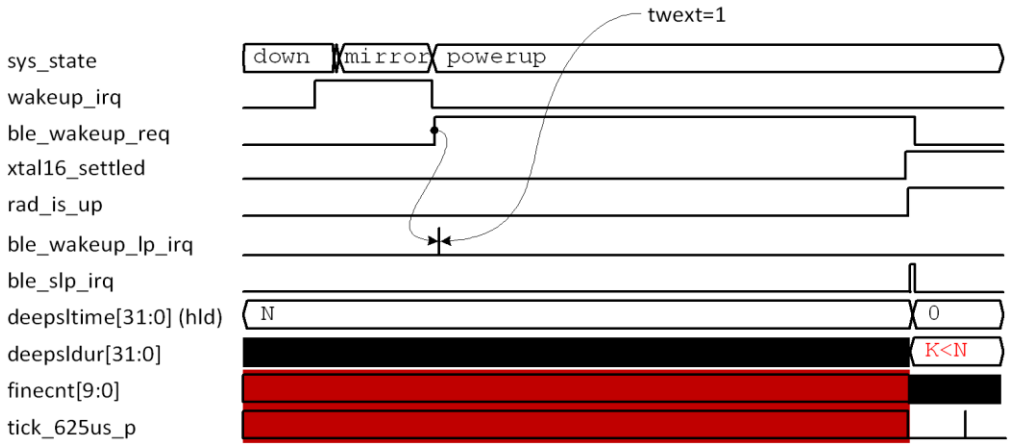


Figure 132. Exit Bluetooth LE Deep Sleep mode due to external event

As soon as the bit field BLE_WAKEUP_REQ is set to 1, the BLE_WAKEUP_LP_IRQ is 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 remains high for one "ble_lp_clk" period.

If the BLE_WAKEUP_REQ is high, entering the sleep mode is prohibited. BLE_WAKEUP_REQ event can be disabled by setting BLE_DEEPSLCNTL_REG[EXTWKUPDSB].

9.31 Radio

9.31.1 Introduction

The Radio Transceiver implements the RF part of the Bluetooth LE protocol. Together with the Bluetooth 5.1 PHY layer, it provides up to 93 dB RF link budget for a 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 radio block diagram is given in Figure 133. It comprises the Receiver, Transmitter, Synthesizer, RX/TX combiner block, and Biasing LDOs.

Features

- Single ended RFIO interface, 50 Ω matched
- Alignment free operation
- -90 dBm receiver sensitivity
- Configurable transmit output power from -19.5 dBm up to +2.5 dBm
- Ultra-low power consumption
- Fast frequency tuning minimizes overhead.

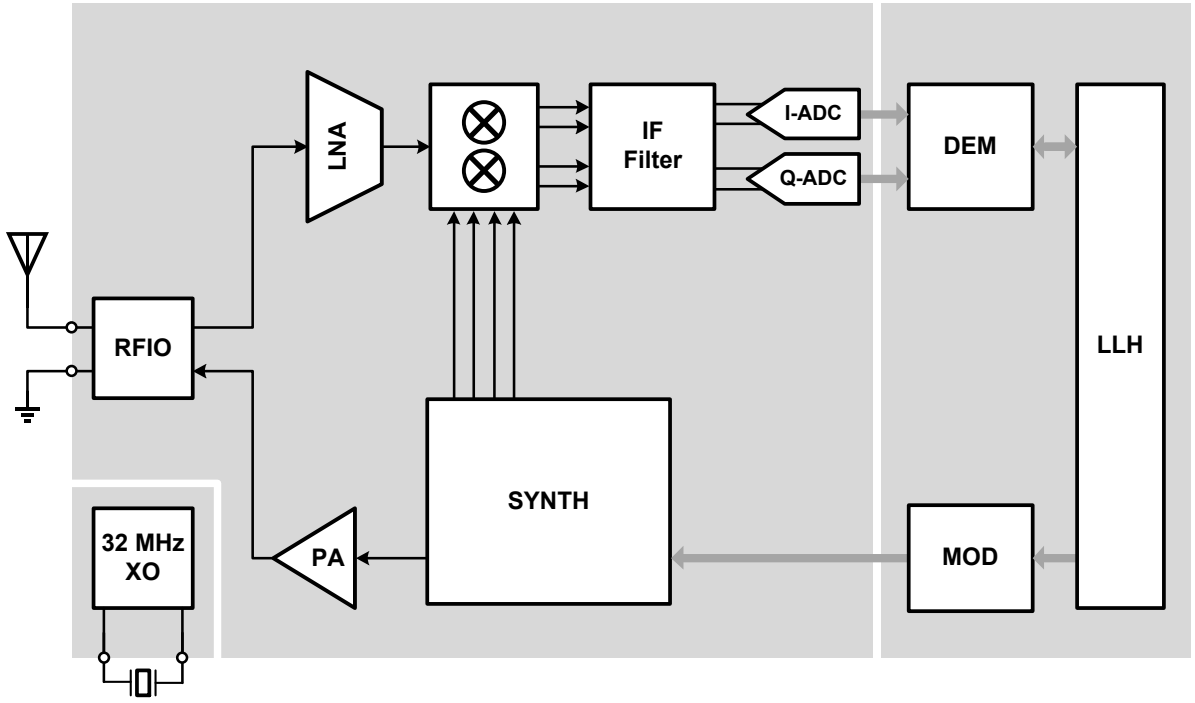


Figure 133. Bluetooth radio block diagram

9.31.2 Architecture

9.31.2.1 Receiver

The RX frontend consists of a selective matching network, a low noise amplifier (LNA), and an image rejection down conversion mixer. The intermediate frequency (IF) part of the receiver comprises a filter with a programmable gain. The LNA and IF Filter gains are controlled by the Automatic Gain Control (AGC). This provides the necessary signal conditioning prior to digitalization. The digital demodulator block (DEM) provides a synchronous bit stream.

9.31.2.2 Synthesizer

The RF Synthesizer generates the quadrature LO signal for the mixer but also generates the modulated TX output signal. The Digitally Controlled Oscillator (DCO) runs at twice the required frequency and a dedicated divide-by-2 circuit generates the 2.4 GHz signals in the required phase relations. The reference frequency is the 32 MHz crystal clock. The modulation of the TX frequency is performed by 2-point modulation.

9.31.2.3 Transmitter

The RF power amplifier (RFPA) is an extremely efficient Class-D structure, providing typically the power ranging from -19.5 dBm to +2.5 dBm to the antenna. It is fed by the DCO's divide-by-2 circuit and delivers its TX power to the antenna pin through the combined RX/TX matching circuit.

9.31.2.4 RFIO

The RX/TX combiner block is a unique feature of the RA6W2. 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 a resistor of 50 Ω to provide the simplest possible interfacing to the antenna on the printed circuit board.

9.31.2.5 Biasing

All RF blocks are supplied by on-chip LDOs. The bias scheme is programmable and optimized for minimum power consumption.

9.31.2.6 RF Monitoring

The Radio is equipped with a monitoring block whose responsibility is to acquire the data provided by the RF Unit and other analog resources, to combine them in words of 32 bits (when necessary), and to store them in system's memory. Data can be the output of the Demodulator (I and Q) or be provided by the GPADC. With the monitoring block, production tests of the corresponding block can be achieved.

10. Package Information

10.1 Moisture Sensitivity Level (MSL)

The MSL is an indicator for the maximum allowable time period (floor lifetime) 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% relative humidity (RH) before the solder reflow process.

BGA93 package is qualified for MSL 3.

Table 87: MSL definitions

MSL level	Floor lifetime
MSL 4	72 hours
MSL 3	168 hours
MSL 2A	4 weeks
MSL 2	1 year
MSL 1	Unlimited at 30 °C/85% RH

10.2 Soldering Information

Refer to the JEDEC standard J-STD-020 for relevant soldering information. This document can be downloaded from http://www.jedec.org.

10.3 Package Outline Drawing

The latest version of the package dimensions are accessible through the Renesas website – VFBGA.

11. Revision History

Revision	Date	Description
1.00	May 29, 2026	First release.

RoHS Compliance

Renesas Electronics' 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.

Appendix A ECAD Design Information

This information supports the development of the PCB ECAD model for this device. It is intended to be used by PCB designers.

A.1 Part Number Indexing

This information supports the development of the PCB ECAD model for this device. It is intended to be used by PCB designers.

Orderable Part Number	Number of Pins	Package Type	Package Code/POD Number
R7JA6W2DEDZAL	93	BGA	PSC-5068/BW0093AA

A.2 Symbol Pin Information

A.2.1 93-BGA

Pin Number	Primary Pin Name	Primary Electrical Type	Alternate Pin Name(S)
A1	VSSRF_5	Power	-
A3	RF_5G	I/O	-
A5	VDDRF	Power	-
A7	XTAL40M_M	Output	-
A9	P1_15	I/O	eMMC_DIO3/SDIO1_D3
A11	P1_13	I/O	eMMC_DIO1/SDIO1_D1
B2	VSSRF_6	Power	-
B4	VSSRF_7	Power	-
B6	VSSRF_PLL	Power	-
B8	XTAL40M_P	Input	-
B10	P1_12	I/O	eMMC_DIO0/SDIO1_D0
C1	VDDRF_PA	Power	-
C3	VSSRF_2	Power	-
C5	VSSRF_IF	Power	-
C9	P1_11	I/O	eMMC_CLK/SDIO1_CLK
C11	VDDIO_DIO2	Power	-
D2	VSSRF_1	Power	-
D4	VSSA_5	Power	-
D6	ESDN	Power	-
D8	VSSIO_2	Power	-
D10	VSSD_2	Power	-
E3	VSSA_4	Power	-
E5	SWDIO	I/O	-
E7	P1_14	I/O	eMMC_DIO2/SDIO1_D2
E9	P1_10	I/O	eMMC_CMD/SDIO1_CMD
E11	P1_08	I/O	OQSPI_CLK
F2	RFTX_2G	Output	-
F4	RST_N	Input	-
F6	P1_09	I/O	OQSPI_CS
F8	P1_02	I/O	OQSPI_D6/eMMC_DIO6
F10	P1_07	I/O	OQSPI_D3
G1	RFRX_2G	Input	-
G3	VSSA_3	Power	-
G5	SWCLK	Input	-
G7	P1_01	I/O	OQSPI_D5/eMMC_DIO5
G9	P1_05	I/O	OQSPI_D1
G11	P1_06	I/O	OQSPI_D2
H2	VSSRF_3	Power	-
H4	P0_06	I/O	ADC2/eMMC_DIO6
H6	P0_11	I/O	QSPIR_D1/SDIO0_D1/eMMC_DIO1
H8	P1_00	I/O	OQSPI_D4/eMMC_DIO4
H10	P1_04	I/O	OQSPI_D0
J1	VSSA_1	Power	-

Pin Number	Primary Pin Name	Primary Electrical Type	Alternate Pin Name(S)
J3	P0_00	I/O	RTC_WAKE_UP
J5	P0_10	I/O	QSPIR_D0/SDIO0_D0/eMMC_DIO0
J7	P1_03	I/O	OQSPI_D7/eMMC_DIO7
J9	VSSIO_3	Power	-
J11	VDDIO_FDIO	Power	-
K2	P0_01	I/O	sen_out
K4	P0_07	I/O	ADC3/MCLK/eMMC_DIO7
K6	VSSD_1	Power	-
K8	VSSA_2	Power	-
K10	VDDA_DCDC_PA	Power	-
L3	P0_04	I/O	ADC0/eMMC_DIO4
L5	VSSIO_1	Power	-
L7	P0_13	I/O	QSPIR_D3/SDIO0_D3/eMMC_DIO3
L9	VSSA_DCDC_PA	Power	-
L11	LX_PA	Power	-
M4	P0_09	I/O	QSPIR_CS/SDIO0_CMD/eMMC_CMD
M6	P0_12	I/O	QSPIR_D2/SDIO0_D2/eMMC_DIO2
M10	VSSA_DCDC_ANA	Power	-
N3	P0_08	I/O	QSPIR_CLK/SDIO0_CLK/eMMC_CLK
N11	VBAT	Power	-
P2	NC	Passive	-
P4	P0_05	I/O	ADC1/eMMC_DIO5
P6	P0_03	I/O	XTAL32K_P
P8	P0_02	I/O	XTAL32K_M
P10	LX_ANA	Power	-
R3	PB0_2	I/O	-
R5	PB0_1	I/O	-
R7	VBAT_HIGH	Power	-
R9	NC	Passive	-
R11	VDDA_DCDC_ANA	Power	-
T4	PB0_5	I/O	-
T6	RST_PB0_0	I/O	-
T8	PB0_4	I/O	-
T10	NC	Passive	-
U1	RFIOM	I/O	-
U3	GNDRF1	Power	-
U5	VSS	Power	-
U7	PB0_3	I/O	-
U9	NC	Passive	-
U11	VDDD	Power	-
V4	XTAL32Mp	I/O	-
V6	GND_DCDC	Power	-
V8	NC	Passive	
V10	NC	Passive	-
W1	RFIOP	I/O	-
W3	GNDRF2	Power	-
W5	XTAL32Mm	I/O	-
W7	VBAT_LOW	Power	-
W9	LX_BT	Power	-
W11	VDDIO_DIO1	Power	-

A.3 Symbol Parameters

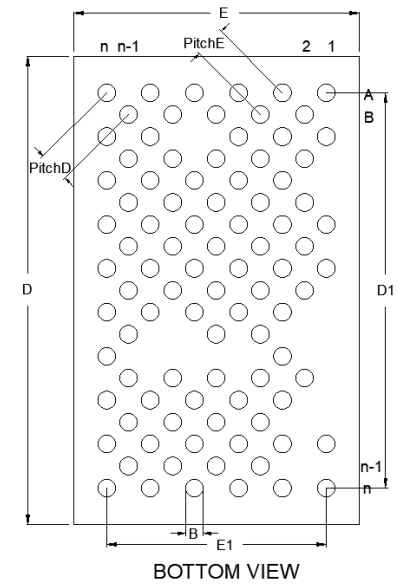
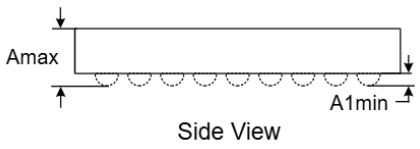
Orderable Part Number	Min Input Voltage	Max Input Voltage	Min Operating Temperature	Max Operating Temperature	Max Frequency	RAM Size	Interface	Number of A/D Converters	Number of SPI Channels	Number of I2C Channels	Number of I2S Channels	Number of UART Channels	Number of Timers/Counters	Number of Programmable I/O
R7JA6W2DEDZAL	1.62 V	3.6 V	-40 °C	85 °C	160MHz	768 kB	Radio, ADC, xSPI, I2C, I2S, SDIO, MMC, UART	4	2	2	1	3	8	35

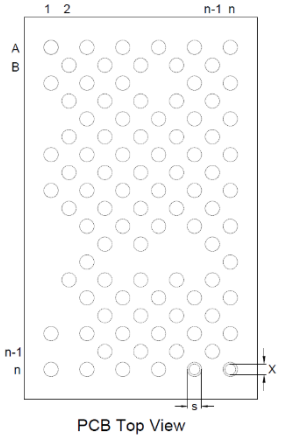
1. Available RAM for CPU (M33).
2. Maximum available I/Os when alternate functions (SPI, I2C, I2S) are not used.

A.4 Footprint Design Information

A.4.1 93-BGA

IPC Footprint Type	Package Code/POD Number	Number of Pins
BGA	PSC-5068/BW0093AA	93

Description	Dimension	Value (mm)	Diagram
Minimum body Length (vertical side)	Dmin	6.30	 BOTTOM VIEW
Maximum body Length (vertical side)	Dmax	6.50	
Average length of grid (vertical side)	D1ave	5.40	
Minimum body Width (horizontal side)	Emin	3.80	
Maximum body Width (horizontal side)	Emax	4.0	
Average length of grid (horizontal side)	E1ave	3.0	
Average ball diameter	Bnom	0.240	
Distance between the center of any two adjacent balls (vertical side)	PitchD	0.424	
Distance between the center of any two adjacent balls (horizontal side)	PitchE	0.424	
P = Plain Grid, S = Staggered Grid	GridType	S	
F = Full Matrix, P = Perimeter, SD = Selectively Depopulated, TE = Thermally Enhanced	MatrixType	SD	
Number of balls (vertical side)	Rows	11	
Number of balls (horizontal side)	Columns	19	
Maximum number of ball positions (Rows x Columns)	Nmax	209	
Number of actual balls present	PinCount	93	
Ball positions removed from matrix. Example: C5-H10, B6-B9, A1	DepopulateBalls	C7, E1, L1, M2, M8, N1, N5, N7, N9, R1, T2, V2	 Side View
Ball positions added back into depopulated matrix. Example: C8, D6-F9	RepopulateBalls	---	
Minimum Standoff Height	A1min	0.100	
Maximum Height	Amax	0.880	

Recommended Land Pattern (NSMD Design)			
Description	Dimension	Value (mm)	Reference Diagram[1]
Diameter of pad. If specified this overrides the calculated value. This can be used to specify a manufacturer's recommended pad size.	X	0.18	 PCB Top View
Solder Mask Expansion	S	0.28	

1. The diagrams show table attribute referencing. For the exact diagrams, see the package outline drawing.