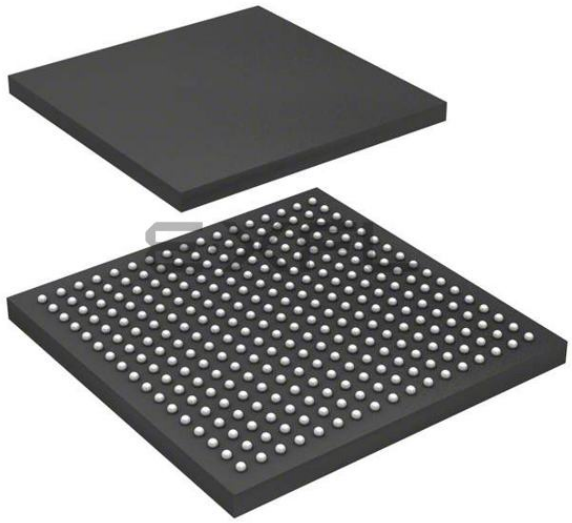


**Welcome to E-XFL.COM**

## Understanding Embedded - Microprocessors

Embedded microprocessors are specialized computing chips designed to perform specific tasks within an embedded system. Unlike general-purpose microprocessors found in personal computers, embedded microprocessors are tailored for dedicated functions within larger systems, offering optimized performance, efficiency, and reliability. These microprocessors are integral to the operation of countless electronic devices, providing the computational power necessary for controlling processes, handling data, and managing communications.

## Applications of Embedded - Microprocessors

Embedded microprocessors are utilized across a broad spectrum of applications, making them indispensable in

| Details | |
|---|---|
| Product Status | Active |
| Core Processor | ARM® Cortex®-A5 |
| Number of Cores/Bus Width | 1 Core, 32-Bit |
| Speed | 500MHz |
| Co-Processors/DSP | Multimedia; NEON™ MPE |
| RAM Controllers | LPDDR1, LPDDR2, LPDDR3, DDR2, DDR3, DDR3L, QSPI |
| Graphics Acceleration | Yes |
| Display & Interface Controllers | Keyboard, LCD, Touchscreen |
| Ethernet | 10/100Mbps (1) |
| SATA | - |
| USB | USB 2.0 + HSIC |
| Voltage - I/O | 3.3V |
| Operating Temperature | -40°C ~ 105°C (TA) |
| Security Features | ARM TZ, Boot Security, Cryptography, RTIC, Secure Fusebox, Secure JTAG, Secure Memory, Secure RTC |
| Package / Case | 289-LFBGA |
| Supplier Device Package | 289-LFBGA (14x14) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/atsama5d27a-cn |

matches the size of the transfer but is managed differently for undefined burst length. See Section 18.10.1.1 “Undefined Length Burst Arbitration”.

- Slot Cycle Limit: when the slot cycle counter has reached the limit value indicating that the current master access is too long and must be broken. See Section 18.10.1.2 “Slot Cycle Limit Arbitration”.

#### 18.10.1.1 Undefined Length Burst Arbitration

In order to prevent long AHB burst lengths that can lock the access to the slave for an excessive period of time, the user can trigger the rearbitration before the end of the incremental bursts. The rearbitration period can be selected from the following Undefined Length Burst Type (ULBT) possibilities:

- Unlimited: no predetermined end of burst is generated. This value enables 1 Kbyte burst lengths.
- 1-beat bursts: predetermined end of burst is generated at each single transfer during the INCR transfer.
- 4-beat bursts: predetermined end of burst is generated at the end of each 4-beat boundary during INCR transfer.
- 8-beat bursts: predetermined end of burst is generated at the end of each 8-beat boundary during INCR transfer.
- 16-beat bursts: predetermined end of burst is generated at the end of each 16-beat boundary during INCR transfer.
- 32-beat bursts: predetermined end of burst is generated at the end of each 32-beat boundary during INCR transfer.
- 64-beat bursts: predetermined end of burst is generated at the end of each 64-beat boundary during INCR transfer.
- 128-beat bursts: predetermined end of burst is generated at the end of each 128-beat boundary during INCR transfer.

The use of undefined length 8-beat bursts, or less, is discouraged since this may decrease the overall bus bandwidth due to arbitration and slave latencies at each first access of a burst.

However, if the usual length of undefined length bursts is known for a master it is recommended to configure the ULBT accordingly.

This selection can be done through the ULBT field of the Master Configuration Registers (MATRIX_MCFG).

#### 18.10.1.2 Slot Cycle Limit Arbitration

The Bus Matrix contains specific logic to break long accesses, such as very long bursts on a very slow slave (e.g., an external low speed memory). At each arbitration time, a counter is loaded with the value previously written in the SLOT_CYCLE field of the related Slave Configuration Register (MATRIX_SCFG) and decreased at each clock cycle. When the counter elapses, the arbiter has the ability to rearbitrate at the end of the current AHB access cycle.

Unless a master has a very tight access latency constraint, which could lead to data overflow or underflow due to a badly undersized internal FIFO with respect to its throughput, the Slot Cycle Limit should be disabled (SLOT_CYCLE = 0) or set to its default maximum value in order not to inefficiently break long bursts performed by some masters.

In most cases, this feature is not needed and should be disabled for power saving.

**Warning:** This feature cannot prevent any slave from locking its access indefinitely.

### 18.10.2 Arbitration Priority Scheme

The bus Matrix arbitration scheme is organized in priority pools, each corresponding to an access criticality class as shown in the “Latency Quality of Service” column in Table 18-7.

**Table 18-7: Arbitration Priority Pools**

| Priority Pool | Latency Quality of Service |
|---|---|
| 3 | Latency Critical |
| 2 | Latency Sensitive |
| 1 | Bandwidth Sensitive |
| 0 | Background Transfers |

Round-robin priority is used in the highest and lowest priority pools 3 and 0, whereas fixed level priority is used between priority pools and in the intermediate priority pools 2 and 1. See Section 18.10.2.2 “Round-robin Arbitration”.

For each slave, each master is assigned to one of the slave priority pools through the priority registers for slaves (MxPR fields of MATRIX_PRAS and MATRIX_PRBS). When evaluating master requests, this priority pool level always takes precedence.

After reset, most of the masters belong to the lowest priority pool (MxPR = 0, Background Transfer) and are therefore granted bus access in a true round-robin order.

### 21.9.10 AIC Interrupt Pending Register 3

**Name:** AIC_IPR3

**Address:** 0xFC02002C (AIC), 0xF803C02C (SAIC)

**Access:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| PID127 | PID126 | PID125 | PID124 | PID123 | PID122 | PID121 | PID120 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PID119 | PID118 | PID117 | PID116 | PID115 | PID114 | PID113 | PID112 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PID111 | PID110 | PID109 | PID108 | PID107 | PID106 | PID105 | PID104 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PID103 | PID102 | PID101 | PID100 | PID99 | PID98 | PID97 | PID96 |

**PIDx: Interrupt Pending**

0: The corresponding interrupt is not pending.

1: The corresponding interrupt is pending.

### 26.6.7 RTC Time Alarm Register (UTC_MODE)

**Name:** RTC_TIMALR (UTC_MODE)

**Address:** 0xF80480C0

**Access:** Read/Write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| UTC_TIME | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| UTC_TIME | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| UTC_TIME | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| UTC_TIME | | | | | | | |

This configuration is relevant only if UTC = 1 in RTC_MR.

**UTC_TIME: UTC_TIME Alarm**

This field is the alarm field corresponding to the UTC time counter. To change it, proceed as follows:

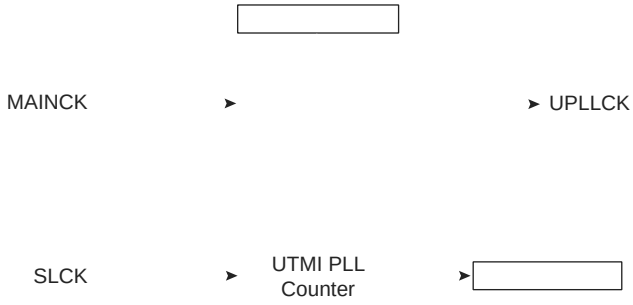
1. Disable the UTC alarm by clearing the UTCEN bit in RTC_CALALR if it is not already cleared.
2. Change the UTC_TIME alarm value.
3. Enable the UTC alarm by setting the UTCEN bit in RTC_CALALR.

## 32.7 UTMI PLL Clock

The source of the UTMI PLL (UPLL) is the Main clock (MAINCK). MAINCK must select the Main crystal oscillator to meet the frequency accuracy required by USB.

The crystal frequency selection among 12, 16 or 24 MHz must be configured to the correct value in the field SFR_UTMICKTRIM.FREQ, in order to apply the correct multiplier, x40, x30 or x20, respectively.

**Figure 32-6: UTMI PLL Block Diagram**




Whenever the UTMI PLL is enabled by writing UPLLEN in the UTMI Clock register (CKGR_UCKR), PMC_SR.LOCKU is automatically cleared. The values written in CKGR_UCKR.UPLLCOUNT are loaded in the UTMI PLL counter. The UTMI PLL counter then decrements at the speed of the Slow clock divided by 8 until it reaches 0. At this time, the PMC_SR.LOCKU is set in and can trigger an interrupt to the processor. The user has to load the number of Slow clock cycles required to cover the UTMI PLL transient time into CKGR_UCKR.UPLLCOUNT.

## 32.8 Audio PLL

The Audio PLL is a high-resolution fractional-N digital PLL specifically designed for low jitter operation.

In audio applications, the CLK_AUDIO output pin typically serves as the Master clock frequency generator for external components such as Audio DAC, Audio ADCs, or Audio Codecs, thus saving one crystal on the board.

The reference clock of the Audio PLL is the fast crystal oscillator. The PLL core operating frequency is defined as:

$$f_{AUDIOCORECLK} = f_{ref}\left(ND + 1 + \frac{FRACR}{2^{22}}\right)$$

where $f_{ref}$ is the frequency of the main crystal oscillator. Refer to Section 66.8 “PLL Characteristics” for the limits of $f_{AUDIOCORECLK}$.

The PLL core features two post-dividers enabling the generation of two output clock signals, AUDIOPLLCLK and AUDIOPINCLK. AUDIOPLLCLK is dedicated to the PMC and can be sent to the GCLK input of peripherals or to the Programmable clock outputs PCKx. AUDIOPINCLK is dedicated to driving the external audio pin CLK_AUDIO.

The AUDIOPLLCLK frequency is defined by the following formula:

$$f_{AUDIOPLLCLK} = \frac{f_{AUDIOCORECLK}}{(QDPMC + 1)}$$

The AUDIOPINCLK frequency is defined by the following formula:

$$f_{AUDIOPINCLK} = \frac{f_{AUDIOCORECLK}}{(DIV \times QDAUDIO)}$$

The typical programming sequence of the audio PLL is the following:

1. Disable the PLL by writing ‘0’ in bits PLLEN and RESETN in the Audio PLL Control register 0 (PMC_AUDIO_PLL0).
2. Release the reset of the PLL by writing ‘1’ in PMC_AUDIO_PLL0.RESETN.
3. Configure the PLL frequency by writing QDPMC and ND in PMC_AUDIO_PLL0, QDAUDIO, DIV and FRACR in PMC_AUDIO_PLL1. ND and FRACR must be configured so as to set AUDIOCORECLK frequency in its authorized range. Refer to Section 66. “Electrical Characteristics”.
4. Enable the PLL by writing ‘1’ in PMC_AUDIO_PLL0.PLLEN, PMC_AUDIO_PLL0.PADEN and PMC_AUDIO_PLL0.PMCEN.
5. Wait for the startup time of this PLL. Refer to Section 66. “Electrical Characteristics”.

### 33.22.9 PMC Clock Generator Main Clock Frequency Register

**Name:** CKGR_MCFR

**Address:** 0xF0014024

**Access:** Read/Write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | CCSS |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| – | – | – | RCMEAS | – | – | – | MAINFRDY |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| MAINF | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MAINF | | | | | | | |

This register can only be written if the WPEN bit is cleared in the PMC Write Protection Mode Register.

**MAINF: Main Clock Frequency**

Gives the number of cycles of the clock selected by the bit CCSS within 16 Slow clock periods. To calculate the frequency of the measured clock:

$$f_{SELCK} = (MAINF \times f_{SLCK}) / 16$$

where frequency is in MHz.

**MAINFRDY: Main Clock Frequency Measure Ready**

0: MAINF value is not valid or the measured oscillator is disabled or a measure has just been started by means of RCMEAS.

1: The measured oscillator has been enabled previously and MAINF value is available.

**Note:** To ensure that a correct value is read on the MAINF field, the MAINFRDY flag must be read at 1 then another read access must be performed on the register to get a stable value on the MAINF field.

**RCMEAS: RC Oscillator Frequency Measure (write-only)**

0: No effect.

1: Restarts measuring of the frequency of the Main clock source. MAINF will carry the new frequency as soon as a low to high transition occurs on the MAINFRDY flag.

The measure is performed on the main frequency (i.e., not limited to RC oscillator only), but if the Main clock frequency source is the 8 to 24 MHz crystal oscillator, the restart of measuring is not needed because of the well known stability of crystal oscillators.

**CCSS: Counter Clock Source Selection**

0: The clock of the MAINF counter is the RC oscillator.

1: The clock of the MAINF counter is the crystal oscillator.

**Figure 36-13: Self-refresh Mode Entry, TIMEOUT = 0**

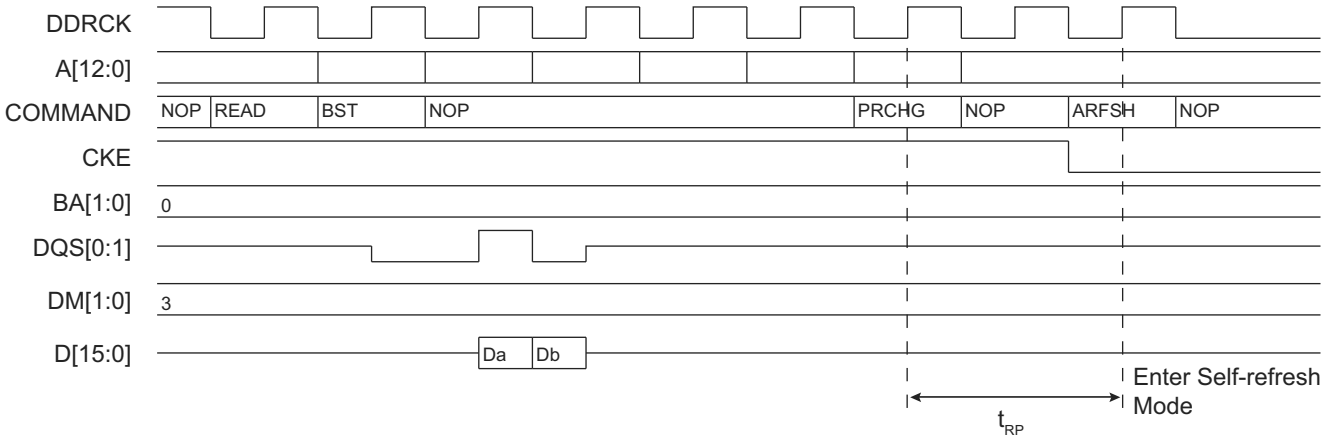



**Figure 36-14: Self-refresh Mode Entry, TIMEOUT = 1 or 2**

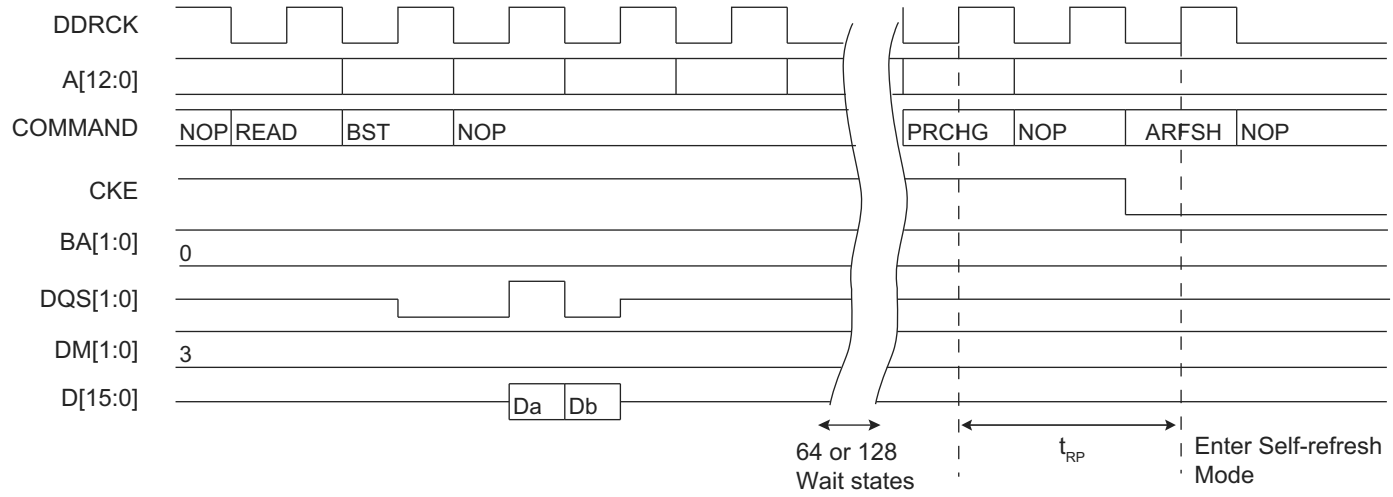



**Figure 36-15: Self-refresh Mode Exit**

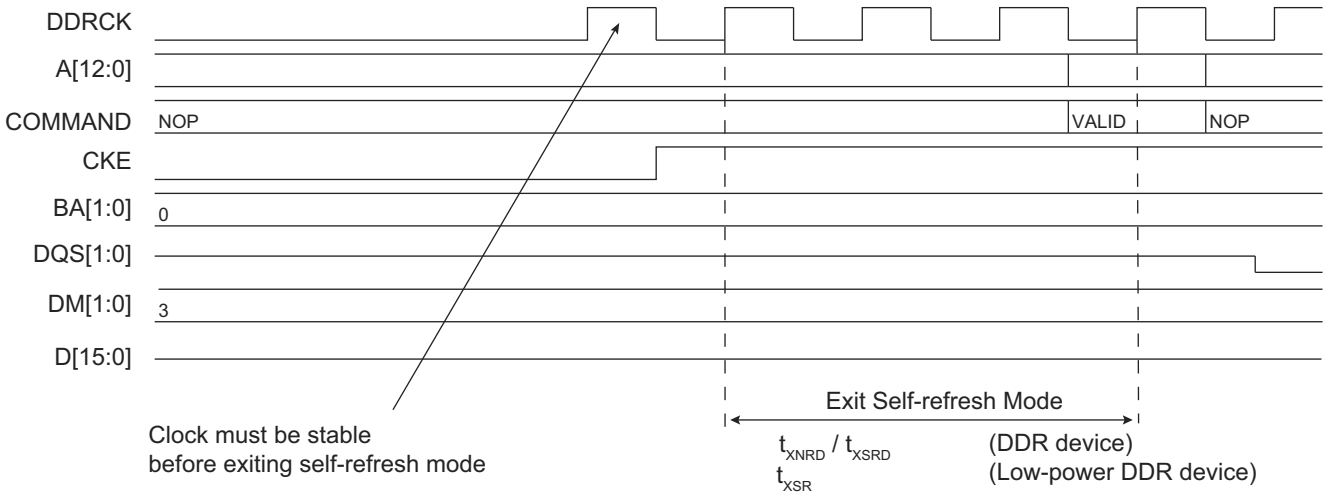
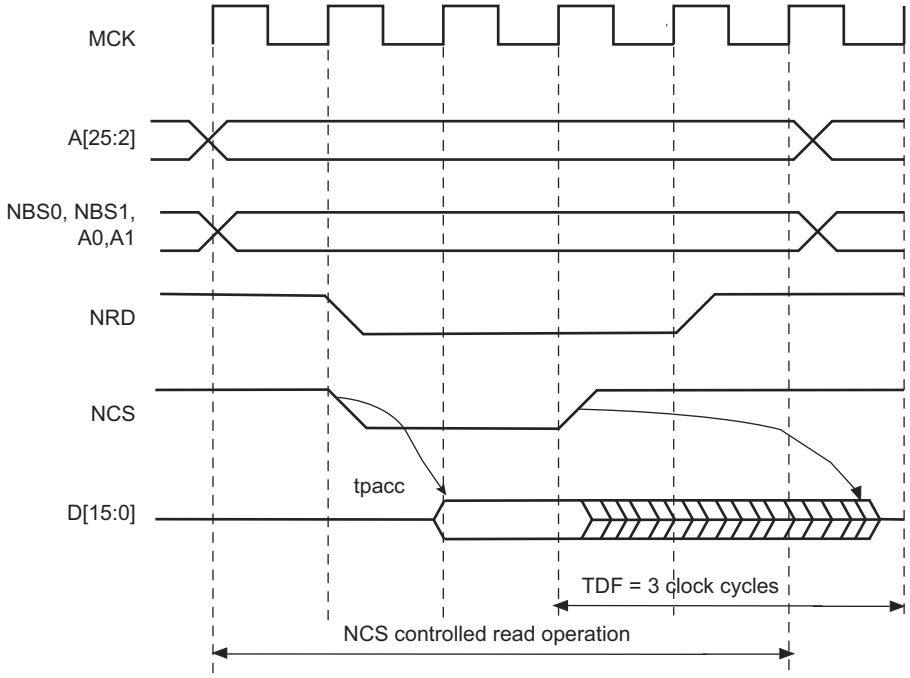


**Figure 37-18: TDF Period in NCS Controlled Read Operation (TDF = 3)**




### 37.13.2 TDF Optimization Enabled (TDF_MODE = 1)

When the TDF_MODE of the HSMC_MODE register is set to 1 (TDF optimization is enabled), the SMC takes advantage of the setup period of the next access to optimize the number of wait states cycle to insert.

Figure 37-19 shows a read access controlled by NRD, followed by a write access controlled by NWE, on Chip Select 0. Chip Select 0 has been programmed with:

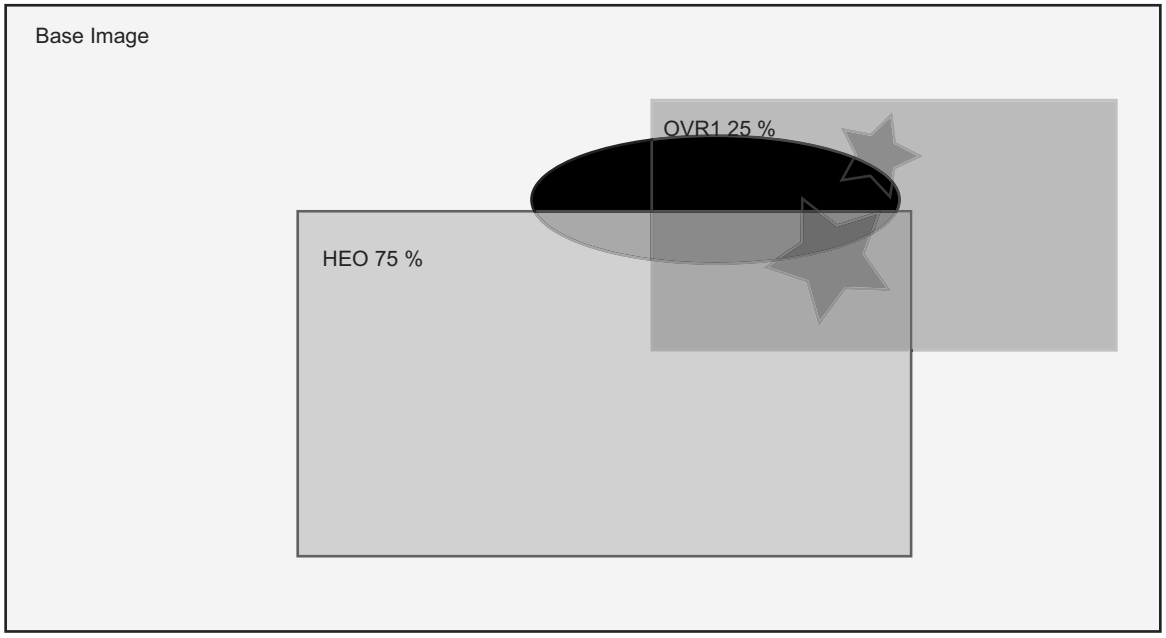
NRD_HOLD = 4; READ_MODE = 1 (NRD controlled)

NWE_SETUP = 3; WRITE_MODE = 1 (NWE controlled)

TDF_CYCLES = 6; TDF_MODE = 1 (optimization enabled).

#### 39.6.10.4 Window Blending

**Figure 39-13: 256-level Alpha Blending**




Video Prioritization Algorithm 1: OVR1 > HEO > BASE

#### 39.6.10.5 Color Keying

Color keying involves a method of bit-block image transfer (Blit). This entails blitting one image onto another where not all the pixels are copied. Blitting usually involves two bitmaps: a source bitmap and a destination bitmap. A raster operation (ROP) is performed to define whether the iterated color or the overlay color is to be visible or not.

- Source Color Keying

If the masked overlay color matches the color key, the iterated color is selected and Source Color Keying is activated using the following configuration sequence:

1. Select the overlay to blit.
2. Write a ‘0’ to DSTKEY.
3. Activate Color Keying by writing a ‘1’ to CRKEY.
4. Configure the Color Key by writing RKEY, GKEY and BKEY fields.
5. Configure the Color Mask by writing RKEY, GKEY and BKEY fields.

When the field RMASK, GMASK, or BMASK is configured to ‘0’, the comparison is disabled and the raster operation is activated.

- Destination Color Keying

If the iterated masked color matches the color key then the overlay color is selected, Destination Color Keying is activated using the following configuration sequence:

1. Select the overlay to blit.
2. Write a ‘1’ to DSTKEY.
3. Activate Color Keying by writing a ‘1’ to CRKEY bit
4. Configure the Color Key by writing RKEY, GKEY and BKEY fields.
5. Configure the Color Mask by writing RKEY, GKEY and BKEY fields.

When the field RMASK, GMASK, or BMASK is configured to ‘0’, the comparison is disabled and the raster operation is activated.

### 39.6.11 LCDC PWM Controller

This block generates the LCD contrast control signal (LCDPWM) to make possible the control of the display's contrast by software. This is an 8-bit PWM (Pulse Width Modulation) signal that can be converted to an analog voltage with a simple passive filter.

### 39.7.48 Overlay 1 Configuration Register 3

**Name:** LCDC_OVR1CFG3

**Address:** 0xF0000178

**Access:** Read/Write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | YSIZE | | |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| YSIZE | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | XSIZE | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| XSIZE | | | | | | | |

**XSIZE: Horizontal Window Size**

Overlay 1 window width in pixels. The window width is set to (XSIZE + 1).

The following constraint must be met: XPOS + XSIZE ≤ PPL

**YSIZE: Vertical Window Size**

Overlay 1 window height in pixels. The window height is set to (YSIZE + 1).

The following constraint must be met: YPOS + YSIZE ≤ RPF

### 42.7.10 UHPHS Asynchronous List Address Register

**Name:** UHPHS_ASYNCLISTADDR

**Access:** Read/Write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| LPL | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| LPL | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LPL | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LPL | | | – | | | | |

This 32-bit register contains the address of the next asynchronous queue head to be executed. If the host controller is in 64-bit mode (as indicated by a 1 in the 64-bit Addressing Capability field in the UHPHS_HCCPARAMS register), then the most significant 32 bits of every control data structure address comes from the UHPHS_CTRLDSSEGMENT register (See **Section 42.7.8 “UHPHS Control Data Structure Segment Register”**). Bits [4:0] of this register cannot be modified by system software and will always return a zero when read. The memory structure referenced by this physical memory pointer is assumed to be 32-byte (cache line) aligned. This register must be written as a DWord. Byte writes produce undefined results.

**LPL: Link Pointer Low**

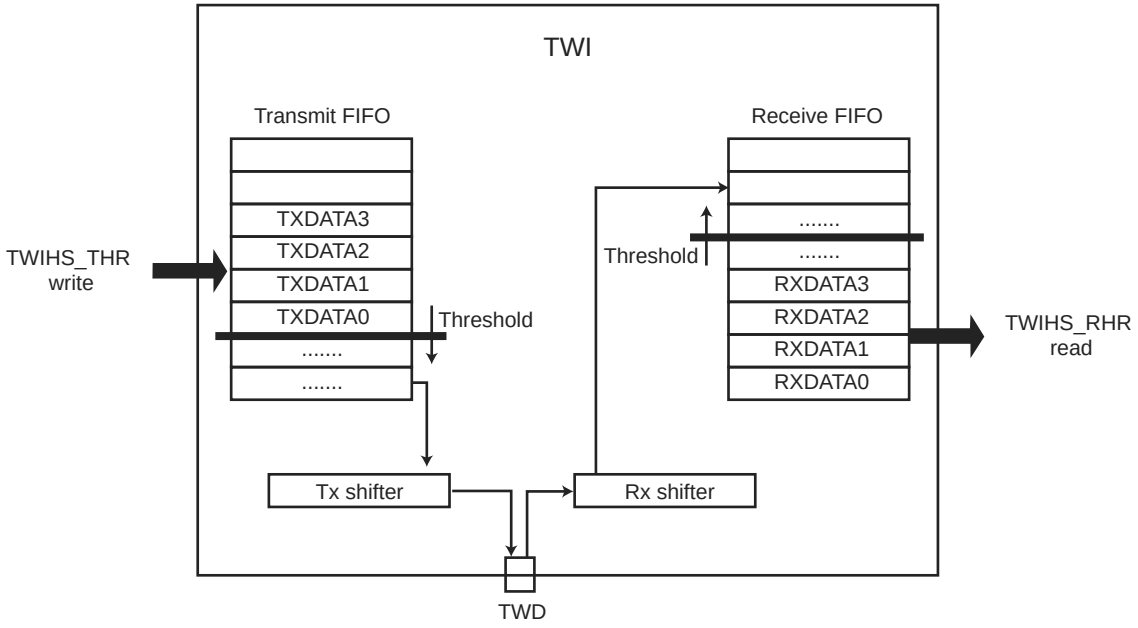
These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (QH).

#### 46.6.5.11 FIFOs

The TWIHS includes two FIFOs which can be enabled/disabled using the FIFOEN/FIFODIS bits in the TWIHS_CR. It is recommended to disable both Master and Slave modes before enabling or disabling the FIFO (MSDIS and SVDIS bit in TWIHS_CR).

Writing the FIFOEN bit to ‘1’ will enable a 16-data Transmit FIFO and a 16-data Receive FIFO.

**Figure 46-51: FIFOs Block Diagram**




- Sending/Receiving Data with FIFO Enabled

With the Transmit FIFO enabled, any write access to the TWIHS_THR will bring the written data to the Transmit FIFO. As a consequence, it is not mandatory any more to monitor the TXRDY flag state to send multiple data without DMAC.

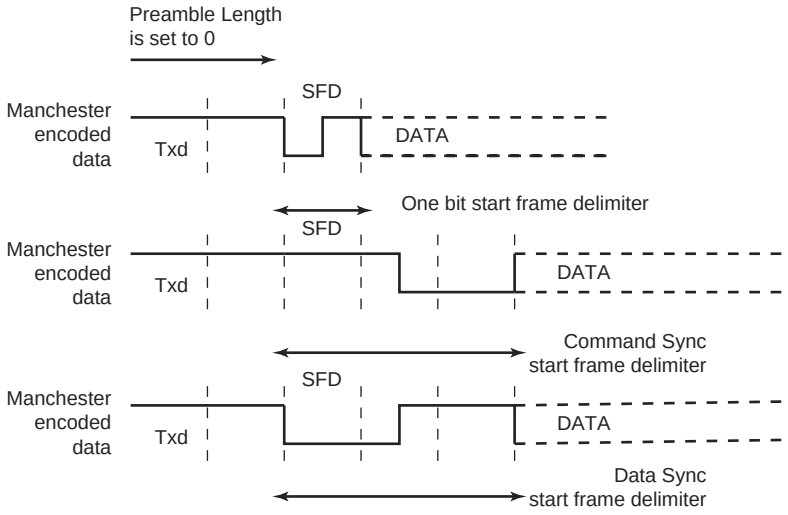
Knowing the number of data to send and provided there is enough space in the Transmit FIFO, all the data to send can be written successively in the TWIHS_THR without checking the TXRDY flag between each access. The Transmit FIFO state can be checked reading the TXFL field in the TWIHS_FLR.

With Receive FIFO enabled, any read access on TWIHS_RHR will pull out a data from the Receive FIFO. As a consequence, it is not mandatory any more to monitor the RXRDY flag when DMAC is not used and there are multiple data to read.

When data are present in the Receive FIFO (RXRDY flag set to ‘1’), the exact number of data can be checked with the RXFL field in the TWIHS_FLR and all the data read successively in the TWIHS_RHR without checking the RXRDY flag between each access.

a half bit times, then a transition to logic zero for the second one and a half bit times. If the FLEX_US_MR.MODSYNC bit is set to 1, the next character is a command. If it is set to 0, the next character is a data. When direct memory access is used, the MODSYNC bit can be immediately updated with a modified character located in memory. To enable this mode, the FLEX_US_MR.VAR_SYNC bit must be set. In this case, the FLEX_US_MR.MODSYNC bit is bypassed and the sync configuration is held in the FLEX_US_THR.TXSYNH bit. The USART character format is modified and includes sync information.

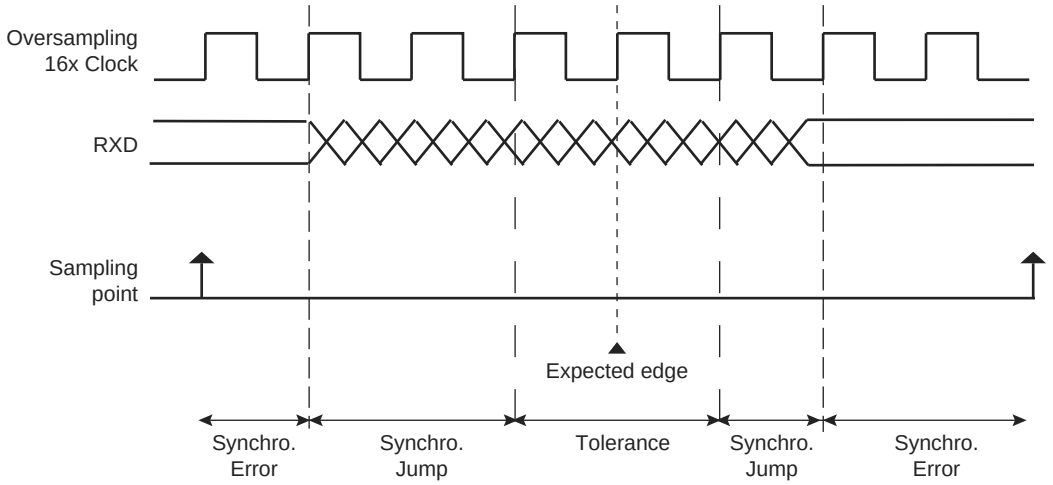
**Figure 47-9: Start Frame Delimiter**




- Drift Compensation

Drift compensation is available only in 16X Oversampling mode. An hardware recovery system allows a larger clock drift. To enable the hardware system, the bit in the FLEX_US_MAN register must be set. If the RXD edge is one 16X clock cycle from the expected edge, this is considered as normal jitter and no corrective actions is taken. If the RXD event is between 4 and 2 clock cycles before the expected edge, then the current period is shortened by one clock cycle. If the RXD event is between 2 and 3 clock cycles after the expected edge, then the current period is lengthened by one clock cycle. These intervals are considered to be drift and so corrective actions are automatically taken.

**Figure 47-10: Bit Resynchronization**




#### 47.7.3.3 Asynchronous Receiver

If the USART is programmed in Asynchronous operating mode (SYNC = 0), the receiver oversamples the RXD input line. The oversampling is either 16 or 8 times the baud rate clock, depending on the FLEX_US_MR.OVER bit.

### 53.6.39 MCAN Transmit Buffer Add Request

**Name:** MCAN_TXBAR

**Address:** 0xF80540D0 (0), 0xFC0500D0 (1)

**Access:** Read/Write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| AR31 | AR30 | AR29 | AR28 | AR27 | AR26 | AR25 | AR24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| AR23 | AR22 | AR21 | AR20 | AR19 | AR18 | AR17 | AR16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| AR15 | AR14 | AR13 | AR12 | AR11 | AR10 | AR9 | AR8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| AR7 | AR6 | AR5 | AR4 | AR3 | AR2 | AR1 | AR0 |

**ARx: Add Request for Transmit Buffer x**

Each Transmit Buffer has its own Add Request bit. Writing a ‘1’ will set the corresponding Add Request bit; writing a ‘0’ has no impact. This enables the processor to set transmission requests for multiple Transmit Buffers with one write to MCAN_TXBAR. MCAN_TXBAR bits are set only for those Transmit Buffers configured via TXBC. When no Transmit scan is running, the bits are reset immediately, else the bits remain set until the Transmit scan process has completed.

0: No transmission request added.

1: Transmission requested added.

**Note:** If an add request is applied for a Transmit Buffer with pending transmission request (corresponding MCAN_TXBRP bit already set), this Add Request is ignored.

## 55.5 Product Dependencies

### 55.5.1 I/O Lines

The pins used for interfacing the PDMIC are multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the peripheral functions to PDMIC pins.

**Table 55-2: I/O Lines**

| Instance | Signal | I/O Line | Peripheral |
|---|---|---|---|
| PDMIC | PDMIC_CLK | PB12 | D |
| PDMIC | PDMIC_CLK | PB27 | D |
| PDMIC | PDMIC_DAT | PB11 | D |
| PDMIC | PDMIC_DAT | PB26 | D |

### 55.5.2 Power Management

The PDMIC is not continuously clocked. The user must first enable the PDMIC peripheral clock and the PDMIC Generic Clock in the Power Management Controller (PMC) before using the controller.

### 55.5.3 Interrupt Sources

The PDMIC interrupt line is connected on one of the internal sources of the Interrupt Controller. Using the PDMIC interrupt requires the Interrupt Controller to be programmed first.

**Table 55-3: Peripheral IDs**

| Instance | ID |
|---|---|
| PDMIC | 48 |

### 56.7.17 PWM Interrupt Status Register 2

**Name:** PWM_ISR2

**Address:** 0xF802C040

**Access:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| CMPU7 | CMPU6 | CMPU5 | CMPU4 | CMPU3 | CMPU2 | CMPU1 | CMPU0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| CMPM7 | CMPM6 | CMPM5 | CMPM4 | CMPM3 | CMPM2 | CMPM1 | CMPM0 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | UNRE | – | – | WRDY |

**WRDY: Write Ready for Synchronous Channels Update**

0: New duty-cycle and dead-time values for the synchronous channels cannot be written.

1: New duty-cycle and dead-time values for the synchronous channels can be written.

**UNRE: Synchronous Channels Update Underrun Error**

0: No Synchronous Channels Update Underrun has occurred since the last read of the PWM_ISR2 register.

1: At least one Synchronous Channels Update Underrun has occurred since the last read of the PWM_ISR2 register.

**CMPMx: Comparison x Match**

0: The comparison x has not matched since the last read of the PWM_ISR2 register.

1: The comparison x has matched at least one time since the last read of the PWM_ISR2 register.

**CMPUx: Comparison x Update**

0: The comparison x has not been updated since the last read of the PWM_ISR2 register.

1: The comparison x has been updated at least one time since the last read of the PWM_ISR2 register.

**Note:** Reading PWM_ISR2 automatically clears flags WRDY, UNRE and CMPSx.

#### 57.4.5.2 Fuse Programming

All the fuses can be written by software.

The sequence of instructions to program fuses is the following:

1. Write the key code 0xFB in the Key Register (SFC_KR).
2. Write the word to program in the corresponding Data Register (SFC_DRx).
   For example, if fuses 0 to 31 must be programmed, Data Register 0 (SFC_DR0) must be written. If fuses 32 to 61 must be programmed, Data Register 1 (SFC_DR1) must be written. Only the data bits set to level ‘1’ are programmed.
3. Wait for flag PGMC to rise in the Status Register (SFC_SR) by polling or interrupt.
4. Check the value of flag PGMF: if it is set to 1, it means that the programming procedure failed. After programming, the fuses are read back in the corresponding SFC_DRx.

#### 57.4.5.3 Fuse Masking

It is possible to mask a fuse array. Once the fuse masking is enabled, the data registers from SFC_DR20 to SFC_DR23 are read at a value of ‘0’, regardless of the fuse state (the registers that are masked depend on the SFC hardware customizing).

To activate fuse masking, the MSK bit of the SFC Mode Register (SFC_MR) must be written to level ‘1’. The MSK bit is set-only. Only a hardware reset can disable fuse masking.

The MSK bit has no effect on the programming of masked fuses.

### 57.4.6 Fuse Functions

The “Fuse Box Controller” section defines the fuse bits that can be used as general purpose bits when standard boot is used.

If secure boot is used, refer to the device “Secure Boot Strategy” application note included in the Secure Package.

### 59.4.6 AESB Interrupt Status Register

**Name:** AESB_ISR

**Address:** 0xF001C01C

**Access:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| URAT | | | | – | – | – | URAD |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | – | – | – | DATRDY |

**DATRDY: Data Ready**

0: Output data not valid.

1: Encryption or decryption process is completed.

DATRDY is cleared when a Manual encryption/decryption occurs (START bit in AESB_CR) or when a software triggered hardware reset of the AESB interface is performed (SWRST bit in AESB_CR).

AESB_MR.LOD = 0: In Manual and Auto modes, the DATRDY bit can also be cleared when at least one of the Output Data Registers is read.

AESB_MR.LOD = 1: In Manual and Auto modes, the DATRDY bit can also be cleared when at least one of the Input Data Registers is written.

**URAD: Unspecified Register Access Detection Status**

0: No unspecified register access has been detected since the last SWRST.

1: At least one unspecified register access has been detected since the last SWRST.

URAD bit is reset only by the SWRST bit in AESB_CR.

**URAT: Unspecified Register Access**

| Value | Name | Description |
|---|---|---|
| 0x0 | IDR_WR_PROCESSING | Input Data Register written during the data processing when SMOD = 0x2 mode |
| 0x1 | ODR_RD_PROCESSING | Output Data Register read during the data processing |
| 0x2 | MR_WR_PROCESSING | Mode Register written during the data processing |
| 0x3 | ODR_RD_SUBKGEN | Output Data Register read during the sub-keys generation |
| 0x4 | MR_WR_SUBKGEN | Mode Register written during the sub-keys generation |
| 0x5 | WOR_RD_ACCESS | Write-only register read access |

Only the last Unspecified Register Access Type is available through the URAT field.

URAT field is reset only by the SWRST bit in AESB_CR.

### 65.7.20 ADC Analog Control Register

**Name:** ADC_ACR

**Address:** 0xFC030094

**Access:** Read/Write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | IBCTL | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | – | – | PENDETSENS | |

This register can only be written if the WPEN bit is cleared in the ADC Write Protection Mode Register.

**Note 1:** By default, bits 12 and 13 are set to 1 and 0, respectively, and must not be modified.

**PENDETSENS: Pen Detection Sensitivity**

Modifies the pen detection input pull-up resistor value. Refer to section “Electrical Characteristics” for further details.

**IBCTL: ADC Bias Current Control**

Adapts performance versus power consumption. Refer to “Electrical Characteristics” for further details.

**Table 66-5: DC Characteristics**

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $R_{PULLUP}$ | Pull-up Resistor | GPIO_CLK, GPIO_IO, GPIO: 1.8V | 80 | 143 | 310 | kΩ |
| | | GPIO_CLK, GPIO_IO, GPIO: 3.3V | 40 | 66 | 130 | |
| $R_{PULLDOWN}$ | Pull-down Resistor | GPIO_CLK, GPIO_IO, GPIO: 1.8V | 80 | 161 | 430 | kΩ |
| | | GPIO_CLK, GPIO_IO, GPIO: 3.3V | 40 | 77 | 160 | |
| $R_{PULLUP}$ | Pull-up Resistor | GPIO_AD: 1.8V | 280 | 380 | 480 | kΩ |
| | | GPIO_AD: 3.3V | 280 | 380 | 480 | |
| $R_{PULLDOWN}$ | Pull-down Resistor | GPIO_AD: 1.8V | 280 | 380 | 480 | kΩ |
| | | GPIO_AD: 3.3V | 280 | 380 | 480 | |
| $R_{SERIAL}$ | Serial Resistor | GPIO | – | 30 | – | Ω |
| | | GPIO_IO | – | 13 | – | Ω |
| | | GPIO_CLK, GPIO_AD | – | 0 | – | Ω |

Notes: 1. $V_{DDIO}$ voltage must be equal to $V_{DDIN}$ voltage.

2. Current injection may lead to performance degradation or functional failures.

**Table 66-6: I/O Switching Frequency**

| GPIO Type | Drive | VDD | | | Unit |
|---|---|---|---|---|---|
| | $C_{Load}$ = 30pF | 1.8V | 2.5V | 3.3V | |
| GPIO_IO | Low | 8 | 12 | 15 | MHz |
| | Medium | 60 | 80 | 90 | |
| | High | 80 | 110 | 110 | |
| GPIO_CLK | Low | 10 | 15 | 18 | |
| | Medium | 90 | 100 | 120 | |
| | High | – | – | – | |
| GPIO_AD | Low | 10 | 15 | 18 | |
| | Medium | 90 | 100 | 120 | |
| | High | – | – | – | |
| GPIO | Low | 7 | 10 | 12 | |
| | Medium | 40 | 50 | 60 | |
| | High | 50 | 60 | 70 | |

**Table 66-7: QSPI I/O Switching Frequency**

| GPIO Type | Description | Name | Conditions | Min | Max | Unit |
|---|---|---|---|---|---|---|
| GPIO_QSPI | Maximum output frequency | $f_{max}$ | Load = 30 pF | – | 133 | MHz |
| | Output duty cycle | – | Load = 30 pF | 45 | 55 | % |

### 66.7.3 32.768 kHz Crystal Oscillator Characteristics

**Table 66-21: 32.768 kHz Crystal Oscillator Characteristics**

| Symbol | Parameter | Conditions | | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| $f_{OSC}$ | Operating Frequency | Normal mode with crystal | | – | 32.768 | – | kHz |
| $t_{START}$ | Startup Time | $C_m$ > 3fF | | – | – | 1200 | ms |
| $I_{DDON}$ | Current Consumption | ESR < 50k ohm | $C_{CRYSTAL32}$ = 12.5 pF | – | 440 | 900 | nA |
| | | | $C_{CRYSTAL32}$ = 6 pF | | 600 | 900 | |
| | | ESR < 100k ohm | $C_{CRYSTAL32}$ = 12.5 pF | | 800 | 1200 | |
| | | | $C_{CRYSTAL32}$ = 6 pF | | 700 | 1200 | |
| $C_{PARA32}$ | Internal Parasitic Capacitance | Between XIN32 and XOUT32 | | 1.4 | 1.6 | 1.8 | pF |

**Figure 66-4: 32 kHz Oscillator Schematics**

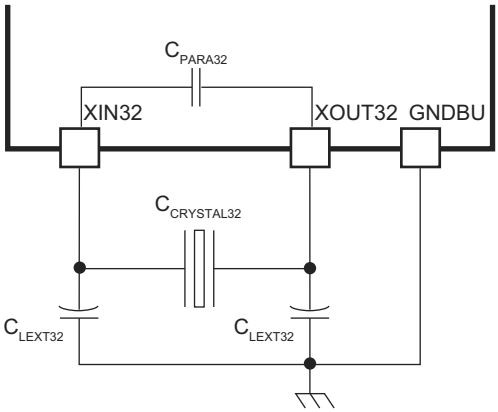



**Table 66-22: Recommended 32.768 kHz Crystal Characteristics**

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| ESR | Equivalent Series Resistor | Crystal at 32.768 kHz | – | – | 100 | kΩ |
| – | Duty Cycle | – | 40 | 50 | 60 | % |
| $C_m$ | Motional Capacitance | Crystal at 32.768 kHz | 3 | – | 8 | fF |
| $C_{SHUNT}$ | Shunt Capacitance | Crystal at 32.768 kHz | 0.6 | – | 2 | pF |
| $C_{CRYSTAL32}$ | Allowed Crystal Capacitance Load(1) | From crystal specification | 6 | – | 12.5 | pF |
| $P_{ON}$ | Drive Level | – | – | – | 0.2 | µW |

Note: 1. The external capacitors value can be determined by using the following formula:

$C_{LEXT32} = (2 \times C_{CRYSTAL32}) - C_{BOARD} - (C_{PARA32} \times 2)$

where:

$C_{LEXT32}$ is the external capacitor value which must be soldered from XIN32 to GND and XOUT32 to GND

$C_{CRYSTAL32}$ is the crystal targeted load

$C_{BOARD}$ is the external board parasitic capacitance (from XIN to GND or XOUT to GND)

$C_{PARA32}$ is the internal parasitic capacitance