UM11071

LPC51U68 User manual

Rev. 1.1 — 17 May 2018 User manual

Document information

Info	Content
Keywords	ARM Cortex-M0+, microcontroller, USB FS device
Abstract	LPC51U68 User Manual

Revision history

Rev	Date	Description
1.1	20180517	LPC51U68 User manual.
Modifications:	• Updated Section 1.2 "Features". Added text: USB 2.0 full-speed host or device controller with on-chip PHY and dedicated DMA controller supporting crystal-less operation in device mode using software library. See Technical note TN00035 for more details.	
1.0	20180222	Initial revision. LPC51U68 User manual.

Contact information

For more information, please visit: **http://www.nxp.com**

For sales office addresses, please send an email to: **salesaddresses@nxp.com**

UM11071

Chapter 1: LPC51U68 Introductory information

Rev. 1.1 — 17 May 2018 **User manual**

1.1 Introduction

The LPC51U68 are ARM Cortex-M0+ based microcontrollers for embedded applications. These devices include 96 KB of on-chip SRAM, 256 KB on-chip flash, full-speed USB device interface, and I2S, three general-purpose timers, one versatile timer with PWM and many other capabilities (SCTimer/PWM), one RTC/alarm timer, one 24-bit Multi-Rate Timer (MRT), a Windowed Watchdog Timer (WWDT), eight flexible serial communication peripherals (each of which can be a USART, SPIs, or I^{2}C interface), and one 12-bit 5.0 Msps ADC, and a temperature sensor.

The ARM Cortex-M0+ coprocessor available on some devices is an energy-efficient and easy-to-use 32-bit core which is code- and tool-compatible with the core. The Cortex-M0+ coprocessor offers up to 100 MHz performance with a simple instruction set and reduced code size.

Refer to LPC51U68 data sheets for complete details on specific products and configurations. LPC51U68 devices are essentially pin-function compatible with LPC5410x and LPC5411x devices in the same package/pinout versions.

1.2 Features

- ARM Cortex-M0+ processor, running at a frequency of up to 100 MHz.
- Single cycle multiplier.
- ARM Cortex-M0+ built-in Nested Vectored Interrupt Controller (NVIC).
- Non-maskable Interrupt (NMI) with a selection of sources.
- Serial Wire Debug (SWD) with 4 breakpoints and 2 watchpoints.
- System tick timer.

- On-Chip memory:
  – 256 KB on-chip flash programming memory with flash accelerator and 256 Byte page write and erase.
  – Up to 96 KB total SRAM composed of up to 64 KB main SRAM, plus an additional 32 KB SRAM.
- ROM API support:
  – Flash In-Application Programming (IAP) and In-System Programming (ISP).
  – ROM-based USB drivers (HID, CDC, MSC, DFU). Flash updates via USB.
  – Booting from valid user code in flash, USART, SPI, and I^{2}C.
  – Legacy, Single, and Dual image boot.
- Serial interfaces:
  – Eight Flexcomm Interface serial peripherals. Each can be selected by software to be a USART, SPI, or I^{2}C interface. Two Flexcomm Interfaces also include an I2S interface, for a total of 2 channel pairs. Each Flexcomm Interface includes a FIFO

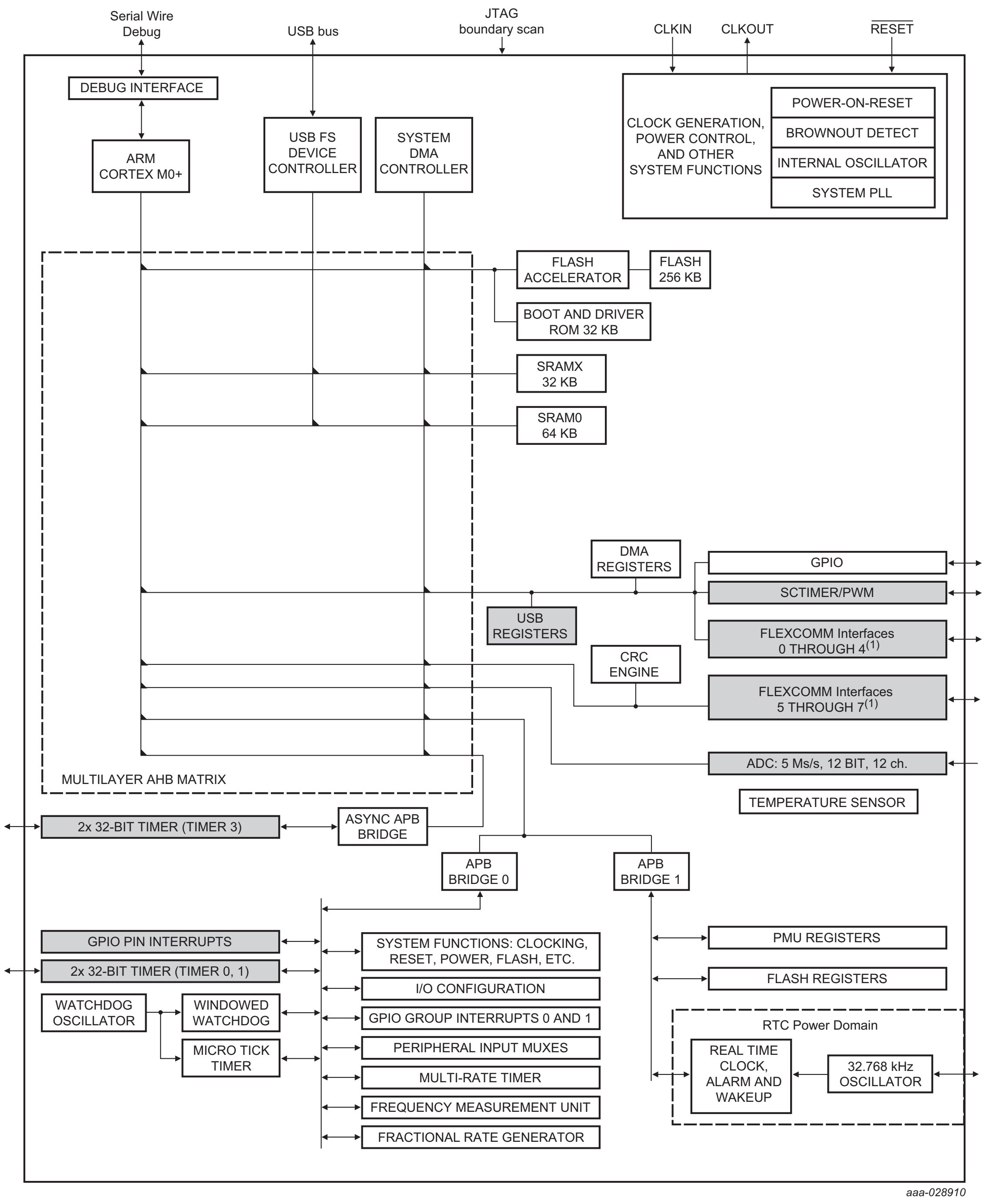
that supports USART, SPI, and I^2S if supported by that Flexcomm Interface. A variety of clocking options are available to each Flexcomm Interface, and include a shared Fractional Rate Generator.

- I^2C supports Fast mode and Fast-mode Plus with data rates of up to 1 Mbit/s and with multiple address recognition and monitor mode. Two sets of true open drain I^2C pins also support High Speed Mode (up to 3.4 Mbit/s) as a slave.
- USB 2.0 full-speed host or device controller with on-chip PHY and dedicated DMA controller supporting crystal-less operation in device mode using software library. See Technical note TN00035 for more details.

- Digital peripherals:
  - DMA controller with 18 channels and 16 programmable triggers, able to access all memories and DMA-capable peripherals.
  - Up to 48 General-Purpose I/O (GPIO) pins. Most GPIOs have configurable pull-up/pull-down resistors, open-drain mode, and input inverter.
  - GPIO registers are located on AHB for fast access.
  - Up to four GPIOs can be selected as pin interrupts (PINT), triggered by rising, falling or both input edges.
  - Two GPIO grouped interrupts (GINT) enable an interrupt based on a logical (AND/OR) combination of input states.
  - CRC engine.
- Analog peripherals:
  - 12-bit ADC with 12 input channels and with multiple internal and external trigger inputs and sample rates of up to 5.0 Msps. The ADC supports two independent conversion sequences.
  - Integrated temperature sensor connected to the ADC.
- Timers
  - Three standard general purpose timers/counters, four of which support up to 4 capture inputs and 4 compare outputs, PWM mode, and external count input. Specific timer events can be selected to generate DMA requests.
  - One SCTimer/PWM (SCT) 8 input and 8 output functions (including capture and match). Inputs and outputs can be routed to/from external pins and internally to/from selected peripherals. Internally, the SCT supports 10 captures/matches, 10 events and 10 states.
  - 32-bit Real-time clock (RTC) with 1 s resolution running in the always-on power domain. A timer in the RTC can be used for wake-up from all low power modes including deep power-down, with 1 ms resolution.
  - Multiple-channel multi-rate 24-bit timer (MRT) for repetitive interrupt generation at up to four programmable, fixed rates.
  - Windowed Watchdog timer (WWDT).
  - Ultra-low power Micro-tick Timer, running from the Watchdog oscillator, that can be used to wake up the device from most low power modes.
- Clock generation:

  - Internal FRO oscillator, factory trimmed for accuracy, that can optionally be used as a system clock as well as other purposes. This oscillator provides a selectable 48 MHz or 96 MHz output, and a 12 MHz output (divided down from the selected higher frequency) that can optionally be used as a system clock as well as other purposes.
  - External clock input for up to 25 MHz.
  - Watchdog oscillator with a frequency range of 6 kHz to 1.5 MHz.
  - 32 kHz low-power RTC oscillator.
  - System PLL allows CPU operation up to the maximum CPU rate without the need for a high-frequency external clock. May be run from the internal FRO 12 MHz output, the external clock input CLKIN, or the RTC oscillator.
  - Clock output function with divider that can reflect many internal clocks.
  - Frequency measurement unit for measuring the frequency of any on-chip or off-chip clock signal.
- Power control:
  - Integrated PMU (Power Management Unit) to minimize power consumption.
  - Reduced power modes: sleep mode, deep-sleep mode, and deep power-down mode.
  - Wake-up from deep-sleep mode on activity on USART, SPI, and I^{2}C peripherals when operating as slaves.
  - Wake-up from sleep, deep-sleep and deep power-down modes from the RTC alarm.
  - The Micro-tick Timer can wake-up the device from most reduced power modes by using the watchdog oscillator when no other on-chip resources are running, for ultra-low power wake-up.
  - Power-On Reset (POR).
  - Brownout detect.
- JTAG boundary scan supported.
- Unique device serial number for identification.
- Single power supply 1.62 V to 3.6 V.
- Operating temperature range of -40°C to +105°C.
- Available as LQFP64 and LQFP48 packages.

1.3 Block diagram



Notes: Each Flexcomm Interface includes USART, SPI, and I2C functions. Flexcomm Interfaces 6 and 7 each also provide an I2S function.

Grey-shaded blocks indicate peripherals that provide DMA requests or are otherwise able to trigger DMA transfers.

Fig 1. Block diagram

1.4 ARM Cortex-M0+ processor

The Cortex-M0+ is a general purpose 32-bit microprocessor with extremely low power consumption. The Cortex-M0+ includes the bulk of the Thumb instruction set and a small subset of Thumb-2 Instructions. The Cortex-M0+ has a 2-stage pipeline in order to decrease power consumption, and includes a single cycle multiplier.

Information about Cortex-M0+ configuration options can be found in Chapter 34.

A multilayer AHB matrix connects the CPU buses and other bus masters to the peripherals in a flexible manner that optimizes performance by allowing peripherals on different slaves ports of the matrix to be accessed simultaneously by different bus masters. More information on the multilayer matrix can be found in Section 2.1.3. Connections in the multilayer matrix are shown in Figure 1. Note that while the AHB bus itself supports word, halfword, and byte accesses, not all AHB peripherals need or provide that support.

APB peripherals are connected to the AHB matrix via two APB buses using separate slave ports from the multilayer AHB matrix. It allows for better performance by reducing collisions between the CPU and the DMA controller, and also for peripherals on the asynchronous bridge to have a fixed clock that does not track the system clock. Note that APB, by definition, does not directly support byte or halfword accesses.

UM11071

Chapter 2: LPC51U68 Memory map

Rev. 1.1 — 17 May 2018 User manual

2.1 General description

The LPC51U68 incorporates several distinct memory regions. Figure 2 shows the overall map of the entire address space from the user program viewpoint following reset.

The APB peripheral area (detailed in Figure 3) is divided into fixed 4 KB slots to simplify addressing.

The registers incorporated into the CPU, such as NVIC, SysTick, and sleep mode control, are located on the private peripheral bus.

2.1.1 Main SRAM

The Main SRAM is comprised of up to a total 96 KB of on-chip static RAM memory. Each SRAM has a separate clock control and power switch, see Section 6.5.16 "AHB Clock Control register 0" and Section 6.5.48 "Power configuration register 0".

Table 1. SRAM configuration

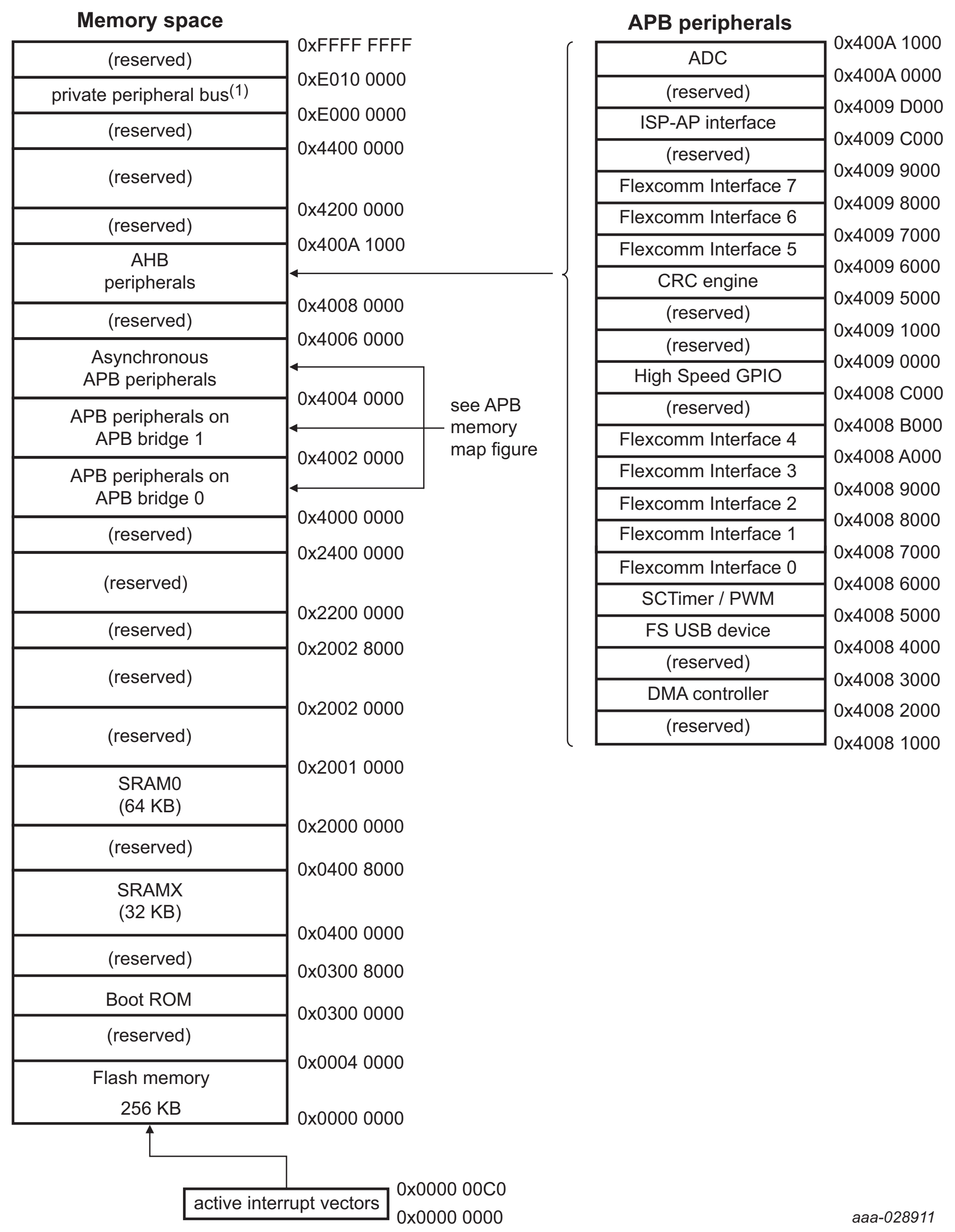
Attributes	SRAM0	SRAMX
(total main SRAM = up to 96 KB)		
Size	Up to 64 KB	Up to 32KB
Address range	Begins at 0x2000 0000	Begins at 0x0400 0000

2.1.1.1 SRAMX

An additional on-chip static RAM memory is available that is not contiguous to the main SRAM. This RAM is called SRAMX, on the main bus of the Cortex-M0+. This RAM can be used, for example, as the location for the program stack, common data, or any other use where a separate access away from the Main SRAM has an advantage. SRAMX can be disabled or enabled in the SYSCON block to save power. See Section 6.5.48 "Power configuration register 0".

2.1.2 Memory mapping

The overall memory map is shown in Figure 2 "Main memory map". Details of APB peripheral mapping are shown in Figure 3 "APB memory map".



1) The private peripheral bus includes CPU peripherals such as the NVIC, SysTick, and the core control registers.

Fig 2. Main memory map

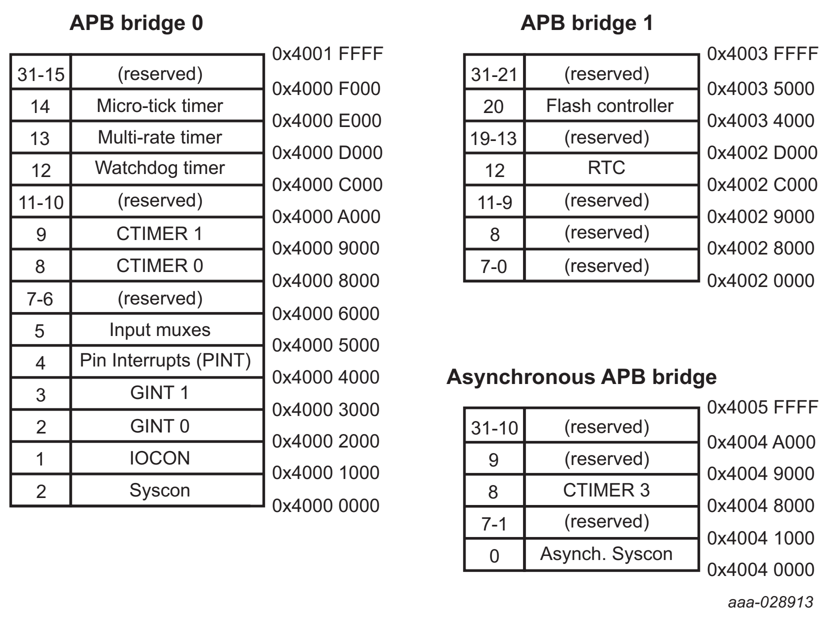


Fig 3. APB memory map

2.1.3 AHB multilayer matrix

The LPC51U68 uses a multi-layer AHB matrix to connect the CPU buses and other bus masters to peripherals in a flexible manner that optimizes performance by allowing peripherals that are on different slave ports of the matrix to be accessed simultaneously by different bus masters. Figure 1 shows details of the potential matrix connections.

UM11071

Chapter 3: LPC51U68 Boot process

Rev. 1.1 — 17 May 2018 **User manual**

3.1 Features

- On-chip boot ROM.
- Contains the boot loader with In-System Programming (ISP) facility and the following APIs:
  – In-Application Programming (IAP) of flash memory.
  – USB ROM based drivers (HID, CDC, MSC, DFU). Supports flash updates via USB ISP mode.
  – Booting from valid user code in flash, USART, SPI, and I^{2}C.
  – Supports Legacy, Single, and Dual image boot.

3.2 Pin description

The parts support ISP via USART, I^{2}C, SPI, and USB. The ISP mode is determined by the state of the ISP0, ISP1, and VBUS pins at power-up or external reset:

Table 2. ISP modes

Boot source	ISP0 (PI00_31)	ISP1 (PIO0_4)	VBUS (PIO1_6)	Description
Flash, no ISP	1	x	x	ISP is bypassed. The device boots from flash if valid user code is detected.
I2C / SPI	0	0	x	The first valid probe message on I^{2}C of Flexcomm Interface 1 or SPI of Flexcomm Interface 3 chooses that interface.
USART	0	1	0	Part enters ISP via the USART of Flexcomm Interface 0.
USB	0	1	1	Allow programming flash as USB mass storage device class (MSC).

The USART ISP interface is implemented on the following pins:

- PIO0_0 for receive
- PIO0_1 for transmit

The USB interface is implemented on the following pins:

- PIO1_6 for VBUS
- USB0_DP for USB D+
- USB0_DM for USB D-

The I2C ISP interface is implemented on the following pins:

- ISP1 (PIO0_4) for I2C/SPI IRQ pin to and from the host system
- PIO0_23 for I2C SCL (clock) signal
- PIO0_24 for I2C SDA (data) signal

The SPI ISP interface is implemented on the following pins:

- PIO0_11 for SPI CLK (clock)

- PIO0_12 for SPI MOSI (master out, slave in) data signal
- PIO0_13 for SPI MISO (master in, slave out) data signal
- PIO0_14 for SPI SSEL (select) signal
- PIO0_4 for I2C/SPI IRQ pin to and from the host system

3.3 General description

The boot loader controls initial operation after reset and also provides the means to program the flash memory. This could be initial programming of a blank device, erasure and re-programming of a previously programmed device, or programming of the flash memory by the application program in a running system.

The boot loader code is executed every time the part is powered on or reset (see Figure 4). The loader can execute the ISP command handler or the user application code.

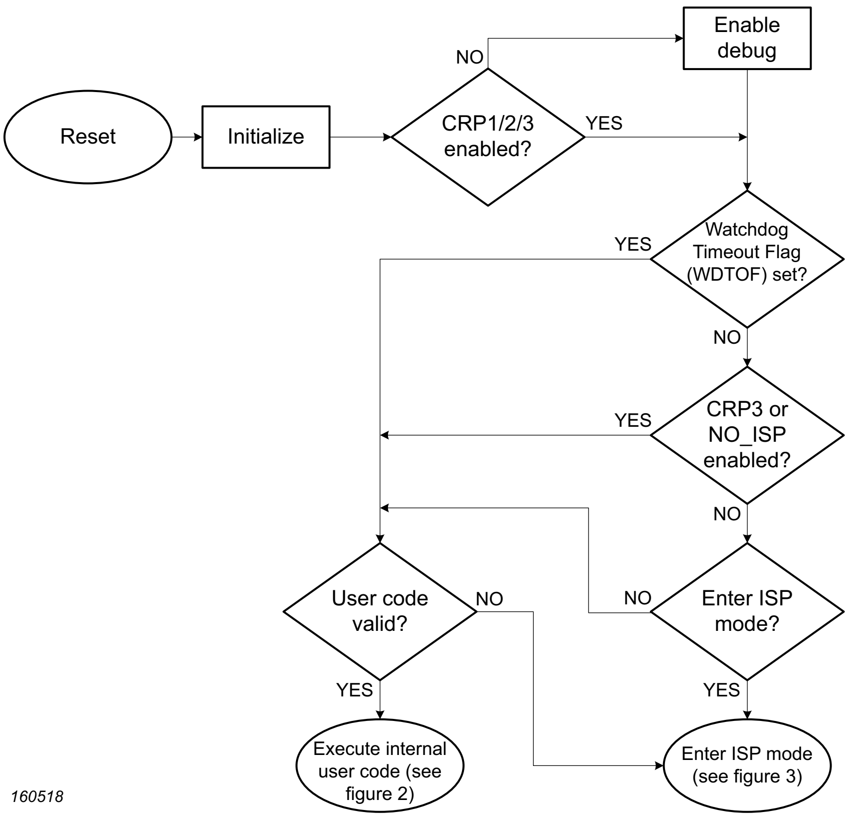
The boot loader version can be read by ISP/IAP calls (see Table 22 or Table 34).

Assuming that power supply pins are at their nominal levels when the rising edge on $\overline{\text{RESET}}$ pin is generated, it may take up to 3 ms before the boot pins are sampled and the decision whether to continue with user code or ISP handler is made. If the boot pins select ISP and the watchdog timeout flag (WDTOF) is set, the external hardware request to start the ISP command handler is ignored. If there is no request for ISP command handler execution, a search is made for a valid user program. If a valid user program is found (see Section 4.3.4 “Criteria for Valid User Code”), then execution is transferred to it. If a valid user program is not found, the selected ISP routine is invoked.

See Section 6.6 “Functional description” for more details of reset, startup behavior, and more. See Chapter 4 “LPC51U68 ISP and IAP” for ISP and IAP commands.

3.4 Boot process

The following figures show the ROM's boot flow process of the LPC51U68. This includes ISP selection, CRP checks, and image type detection. (DE0 = Dual Enhanced image at sector 0, DEn- Dual Enhanced image at sector n (where n > 0), SE = Single Enhanced image).



For DE images, selected CRP is the more restrictive of both images (if available)

Fig 4. Simplified boot flow, figure 1

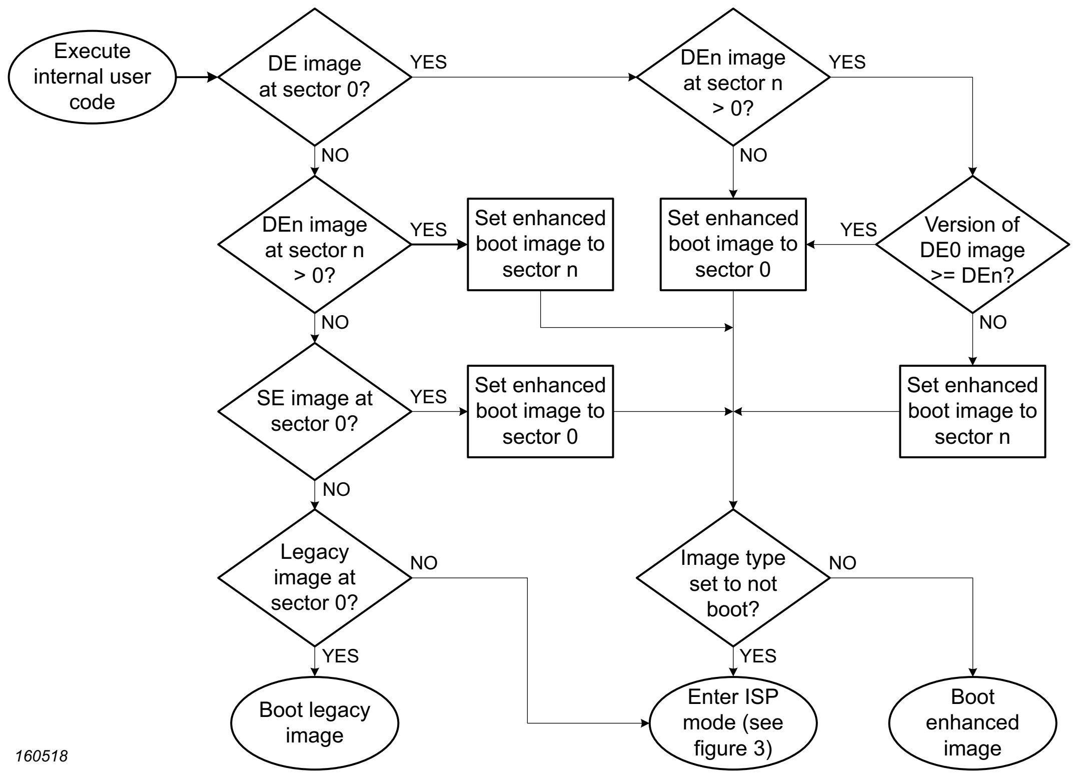

Execute internal user code
DE image at sector 0?
YES
NO
DEn image at sector n > 0?
YES
NO
Version of DE0 image >= DEn?
YES
NO
DEn image at sector n > 0?
YES
NO
Set enhanced boot image to sector n
Set enhanced boot image to sector 0
SE image at sector 0?
YES
NO
Set enhanced boot image to sector 0
Set enhanced boot image to sector n
Legacy image at sector 0?
NO
YES
Image type set to not boot?
NO
YES
Boot legacy image
Enter ISP mode (see figure 3)
Boot enhanced image
160518

Fig 5. Simplified boot flow, figure 2

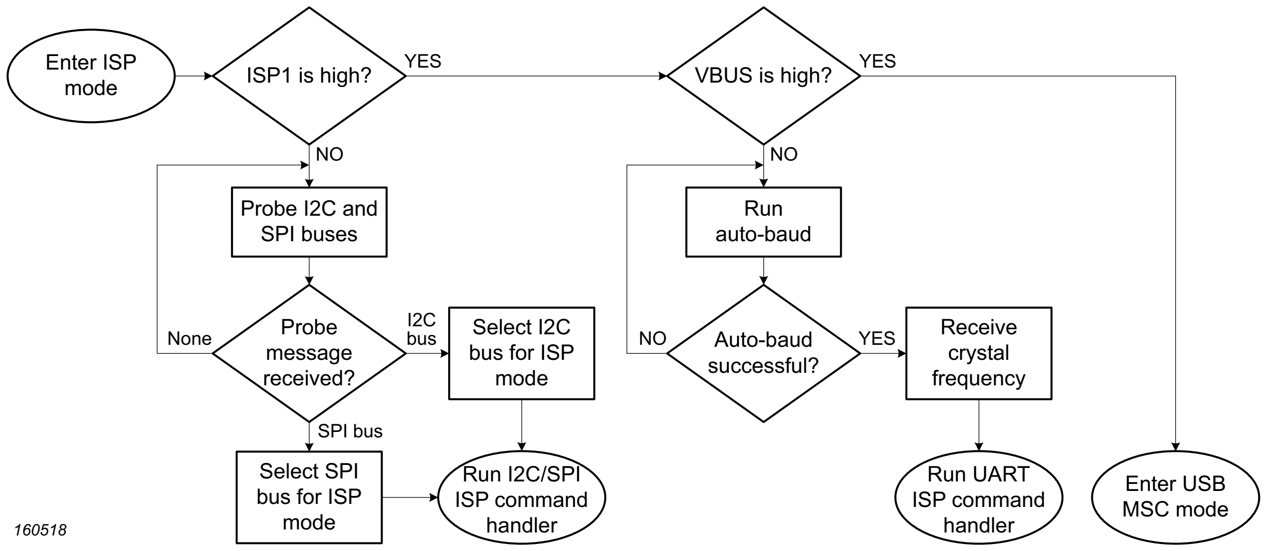


Fig 6. Simplified boot flow, figure 3

3.5 Image boot support

The LPC51U68 supports booting executable images from internal flash located at sector 0, but also supports enhanced images that can boot from sectors greater than 0, even if sector 0 is blank.

The LPC51U68 supports 3 types of images: legacy images, Single Enhanced (SE) images, and Dual Enhanced (DE) images. Selection of the image type is based on several fields defined in the application itself.

3.5.1 Legacy images

A legacy image is located in sector 0 and does not contain an enhanced image marker or an image header structure. The only requirement for a legacy image is that the checksum of the first 8 32-bit words of the image must add to 0x00000000.

3.5.2 Enhanced images

Single Enhanced (SE) and Dual Enhanced (DE) images differ from a legacy image in that they add an enhanced marker value at offset 0x24 in the image and an image header that optionally enables the image CRC capability and adds an image version number. Offset 0x28 in the image must point to the location of a valid image header in the image.

3.5.2.1 Single Enhanced (SE) images

The Single Enhanced image can only be located at sector 0. For an image to be Single Enhanced, it needs to have the Single Enhanced image marker value (0xEDDC9494) at offset 0x24. It must also have a valid image header in the image pointed to at offset 0x28. The image header is explained in Section 3.5.7 "Enhanced Image Boot Block Header".

3.5.2.2 Dual Enhanced (DE) images

The Dual Enhanced image type is similar to the Single Enhanced image type except that it adds the ability to select and boot 1 of 2 images in flash based on the highest version number in the image header. Up to 2 Dual Enhanced images may be located in flash. If only 1 Dual Enhanced image is located in flash, it may be located at the start of any sector. If 2 dual enhanced images are in flash, one of the images must be at sector 0 and the other must start at any sector greater than 0. If 2 images exist, the version number is checked on both images. If the version numbers are the same, the image at sector 0 is executed. If the version numbers are not the same, the image with the highest version number is executed. If more than 2 Dual Enhanced images exist, only the first 2 are checked.

Dual Enhanced images give the ability to run one image from flash, while programming another image located elsewhere in flash. The Dual Enhanced image facility may also be used to locate a boot loader at sector 0 and an application at another sector. If the application is ever erased and the chip is reset, the image at sector 0 will execute following reset.

For an image to be Dual Enhanced, it needs to have the Dual Enhanced image marker value (0x0FFEB6B6) at offset 0x24. It must also have a valid image header in the image pointed to at offset 0x28. The image header is explained in Section 3.5.7 "Enhanced Image Boot Block Header".

For Dual Enhanced images not at sector 0, the image must be linked to run at the sector start address where the image is programmed and must be aligned on a sector boundary.

3.5.3 When to use legacy, single enhanced or dual enhanced images

Each image type has its own advantages and disadvantages. The table below outlines the advantages and disadvantages of each image type and the best use cases for them.

Table 3. Image type considerations

Image type	Advantage	Disadvantage	Best use
Legacy	Simple to create, works with all tool chains without special processing.	Slightly longer boot time than Single Enhanced images.	When debugging an application.
Single Enhanced	Fastest boot time (without CRC), Supports optional CRC checking, more secure.	More complex to build than legacy images, CRC generation can be complex, may increase boot time if CRC is used.	Production systems that benefit from fastest boot time (no CRC).
Dual Enhanced	Single Enhanced advantages plus up to 2 selectable images based on version, more secure.	More complex to build than legacy images, CRC generation can be complex, may increase boot time.	Systems that require boot image cycling based on version.

3.5.4 Default operations when image boot fails

If an image – legacy or enhanced – fails to boot or an image isn’t available to boot, ISP mode is entered based on the states of the ISP1 and USB0 VBUS pins only regardless of the ISP0 pin state.

3.5.5 Enhanced image boot failures

For an image to be enhanced, it must contain a valid enhanced marker at offset 0x24 and a valid image header that is pointed to at offset 0x28. The image header also has requirements that must be met to be considered valid.

Prior to an enhanced image booting, the LPC51U68 performs multiple checks on the image to verify that it is valid or that the image is allowed to boot. If all the image checks are valid, the enhanced image is booted. If any of the checks fail, the enhanced image is not booted and the system enters the ISP mode selected by the ISP0 and VBUS pin states. The following cases on an enhanced image check will prevent the application from booting:

- First 8 words (checksum) do not add up to 0x00000000.
- Image header is invalid.
- Invalid header marker in image header.
- CRC check is invalid (For images where optional CRC check is enabled).
- ISP0 pin is asserted (enters ISP mode) and NO_ISP image type is not used.
- Invalid ‘img_type’ value in the image header.

3.5.6 Modifications to startup code to enable enhanced boot support

Several modifications need to be made to the startup code to enable enhanced boot support. The value at offset 0x24 in the image must contain an enhanced image marker and the value at offset 0x28 must point to a valid image header in the image. See the box below for an example setup using a Single Enhanced image. Changes are in bold and made to the vector table are of the startup code.

```
; Vector Table Mapped to Address 0 at Reset
           AREA    RESET, DATA, READONLY
           EXPORT  __Vectors
__Vectors  DCD     __initial_sp  ; Top of Stack
           DCD     Reset_Handler ; Reset Handler
           DCD     NMI_Handler
           DCD     HardFault_Handler
           DCD     MemManage_Handler
           DCD     BusFault_Handler
           DCD     UsageFault_Handler
__vector_table_0x1c
           DCD     0            ; Checksum of the first 7 words
           DCD     0
           DCD     0xEDDC9494   ; Enhanced image marker, set to 0x0 for legacy boot
           IMPORT  imageHeader
           DCD     imageHeader  ; Pointer to enhanced image header, use 0x0 for legacy
                                  boot
```

The image header for the startup code must be located somewhere in non-volatile memory. A simple image header that doesn't perform CRC is shown below. All field sizes are 32-bits.

```
/* Image header */
const IMAGEHEADER_T imageHeader = {
         IMAGE_ENH_BLOCK_MARKER, /* Required marker for image header */
                  IMG_NO_CRC,     /* No CRC, makes development easier */
                  0x00000000,     /* crc32_len */
                  0x00000000,     /* crc32_val */
                  0x00000000      /* version */
};
```

3.5.7 Enhanced Image Boot Block Header

This header must be present and be valid for any type of enhanced image [Single/Dual]. A pointer to this header structure must be stored at offset 0x28 of the enhanced image and located in non-volatile memory.

Table 4. Image Header structure

Field	Offset	Size (bytes)	Value	Description
header_marker	0x00	0x4	0xFEEDA5A5	Image header marker must always be 0xFEEDA5A5
img_type	0x04	0x4	-	Image type. 0 - IMG_NORMAL 1 - IMG_NO_CRC (All other values invalid, will not boot)
crc_len	0X08	0X4	-	Length of the image. Applications requiring fast boot time can have partial CRC check by altering this length field. Note, the crc_value field should be calculated for the length of image specified in this field.
crc_value	0x0C	0x4	-	CRC32 of the image excluding this field.
Version	0x10	0x4	-	Image version for Dual Enhanced image support

3.5.7.1 Image Type (img_type)

The “img_type” field in the image header alters how the application boots and enters I2C/SPI ISP mode.

- IMG_NORMAL (0):

For this type of image, the ROM code will check the CRC32 of the image. If the CRC32 computation is valid, the image will boot.

- IMG_NO_CRC (1):

For this type of image, no image checks are made of the image header. The image will always boot.

All other values are invalid.

3.5.7.2 Enhanced Image CRC computation

The enhanced image CRC computation is enabled in an enhanced image if a CRC enabled image type (IMG_NORMAL) is used. The CRC is performed from the start of an image (at a sector boundary) and uses an embedded length value (crc_len) and expected CRC value (crc_value) from the programmed image’s image header. CRC computations are always done on a contiguous range of data, but will skip the embedded CRC value location in the image if the CRC range passes through that address.

The CRC length must be a byte length that is 32-bit aligned (8, 12, 16, etc.). The CRC length can be as small as 4 bytes and as large as the image size minus 4 bytes and adjusted for a 32-bit size. The adjustment of (-4) bytes is needed for the CRC value of the image header.

Because the image header can be embedded anywhere in the programmed image, care must be taken to not use the CRC value field in the CRC computation itself. The CRC length field is used in the CRC computation.

Example: CRC computation length for a full image with image length = 40997 bytes

For an image length of 40997 bytes, the CRC length will be the length of the image adjusted by (-4) for the embedded CRC value field. This adjusts the length to 40993 bytes. The length must also be 32-bit aligned, so the length’s lower 2 bytes are masked off. The final length = (40993 & ~0x3) = 40992 bytes.

CRC generation approach for an enhanced image and CRC parameters

The following basic approach is used to generate the CRC length and value field on an enhanced image.

1. Determine the OFFSET in the image the CRC length and value fields are located.

   This can be done by looking at the image header address in the image at offset 0x28. This address points to an enhanced image header in the image. For images in sector 0, the address is the offset in the file to the image header. For images greater than sector 0, the address minus the sector start address is the offset in the file to the image header.
2. Increment the OFFSET by 16 to get the offset for the CRC length (OFFSET_CRCLEN). Increment the OFFSET by 20 to get the offset for the CRC value (OFFSET_CRCVAL).
3. Prepare for CRC using the following parameters:

   CRC_SEED_VALUE = 0xFFFFFFFF

   CRC_MODE is CRC-32, CRC operations on 32-bit values
4. Determine length of CRC generation (CRC_LEN)

   This must be a minimum of 4 bytes and a maximum of the image size minus 4 bytes. If the CRC generation range goes through the CRC value field, the CRC value field will be skipped. The CRC length must be 32-bit aligned and not longer than the size of the image minus 4 bytes.
5. Update the CRC length value (CRC_LEN) in the image header

   Write the desired CRC length (CRC_LEN) into the image file at offset OFFSET_CRCLEN. This is a 32-bit field.
6. Generate CRC-32 value for the CRC length

   Initialize the CRC-32 algorithm with the correct parameters and use the following algorithm to compute the CRC-32 value starting at the beginning of the file.

```
Unsigned long CRC_VAL = 0;
For (CRCADDR = 0; CRCADDR < CRC_LEN; CRCADDR = CRCADDR + 4) {
          If (CRCADDR != OFFSET_CRCVAL) {
               CRC_VAL = CRC-32(CRC_VAL, 32-bit value at address CRCADDR);
          }
}
```

7. Perform 1’s complement on generated CRC-32 value
8. Update the CRC value (CRC_VAL) in the image header

   Write the generated CRC value (CRC_VAL) into the image file at offset OFFSET_CRCVAL. This is a 32-bit field.

Considerations with enhanced images

Images that enable CRC checks can take considerably longer to boot than images that don’t use CRC. The CRC must be computed and verified prior to image boot for these image types. Using a smaller CRC length (ie, 256 bytes) can reduce this time or avoid using a CRC enabled image type if boot time is important.

3.5.7.3 Dual enhanced Image Versioning

The “Version” field in the image header is only used for Dual Enhanced image checks when 2 Dual Enhanced images are detected in sector 0 and another sector. The LPC51U68 will boot the image with the higher version number.

Considerations with dual-boot images

When the system checks an enhanced image to make sure it’s bootable, it performs the legacy checksum verification, verifies that the image marker is valid, and will perform CRC checks on the image if enabled. When 2 images are available, all of these checks are performed on both images prior to checking the version of the images. This may make boot times considerably longer than normal when CRC is enabled.

UM11071

Chapter 4: LPC51U68 ISP and IAP

Rev. 1.1 — 17 May 2018 **User manual**

4.1 How to read this chapter

All LPC51U68 devices include ROM-based services for programming and reading the flash memory in addition to other functions. In-System Programming works on an unprogrammed or previously programmed device using one from a selection of hardware interfaces. In-Application Programming allows application software to do the same kinds of operations.

See specific device data sheets for different flash configurations.

Remark: In addition to the ISP and IAP commands, the flash configuration register (FLASHCFG) can be accessed in the SYSCON block to configure flash memory access times, see Section 6.5.38.

4.2 Features

- In-System Programming: In-System programming (ISP) is programming or reprogramming the on-chip flash memory, using the boot loader software and USART, I^{2}C, or SPI serial port. This can be done when the part resides in the end-user board.
- In Application Programming: In-Application (IAP) programming is performing erase and write operation on the on-chip flash memory, as directed by the end-user application code.
- Small size (256 byte) page erase programming.

4.3 General description

4.3.1 Boot loader

For the boot loader operation and boot pin, see Chapter 3 "LPC51U68 Boot process".

The boot loader version can be read by ISP/IAP calls (see Section 4.5.13 or Section 4.6.6).

4.3.2 Memory map after any reset

The boot ROM is located in the memory region starting from the address 0x0300 0000. The boot loader is designed to run from this memory area, but both the ISP and IAP software use parts of the on-chip RAM. The RAM usage is described later in Section 4.3.7.

4.3.3 Flash content protection mechanism

The LPC51U68 is equipped with the Error Correction Code (ECC) capable Flash memory. The purpose of an error correction module is twofold. Firstly, it decodes data words read from the memory into output data words. Secondly, it encodes data words to be written to the memory. The error correction capability consists of single bit error correction with Hamming code.

The operation of ECC is transparent to the running application. The ECC content itself is stored in a flash memory not accessible by user's code to either read from it or write into it on its own. A byte of ECC corresponds to every consecutive 128 bits of the user accessible Flash. Consequently, Flash bytes from 0x0000 0000 to 0x0000 000F are protected by the first ECC byte, Flash bytes from 0x0000 0010 to 0x0000 001F are protected by the second ECC byte, etc.

Whenever the CPU requests a read from user's Flash, both 128 bits of raw data containing the specified memory location and the matching ECC byte are evaluated. If the ECC mechanism detects a single error in the fetched data, a correction will be applied before data are provided to the CPU. When a write request into the user's Flash is made, write of user specified content is accompanied by a matching ECC value calculated and stored in the ECC memory.

When a sector of Flash memory is erased, the corresponding ECC bytes are also erased. Once an ECC byte is written, it can not be updated unless it is erased first. Therefore, for the implemented ECC mechanism to perform properly, data must be written into the flash memory in groups of 16 bytes (or multiples of 16), aligned as described above.

4.3.4 Criteria for Valid User Code

The reserved CPU exception vector location 7 (offset 0x0000 001C in the vector table) should contain the 2's complement of the check-sum of table entries 0 through 6. This causes the checksum of the first 8 table entries to be 0. The boot loader code checksums the first 8 locations in sector 0 of the flash. For legacy images, if the result is 0, then execution control is transferred to the user code. For enhanced images, the enhanced image marker, enhanced image header, and optional CRC must also be valid in addition to the checksum result being 0.

If the checksum is not 0 or the enhanced image is not valid, the ISP mode selected by the ISP1 and VBUS pins is selected.

4.3.5 Flash partitions

Some IAP and ISP commands operate on sectors and specify sector numbers. In addition, a page erase command is available. The size of a sector is 32 KB and the size of a page is 256 Byte. One sector contains 128 pages. Sector 0 and page 0 are located at address 0x0000 0000.

Table 5. Flash sectors and pages

Sector number	Sector size	Page numbers	Address range	Total flash (including this sector)
0	32 KB	0 - 127	0x0000 0000 - 0x0000 7FFF	32 KB
1	32 KB	128 - 255	0x0000 8000 - 0x0000 FFFF	64 KB
2	32 KB	256 - 383	0x0001 0000 - 0x0001 7FFF	96 KB
3	32 KB	384 - 511	0x0001 8000 - 0x0001 FFFF	128 KB
4	32 KB	512 - 639	0x0002 0000 - 0x0002 7FFF	160 KB
5	32 KB	640 - 767	0x0002 8000 - 0x0002 FFFF	192 KB
6	32 KB	768 - 895	0x0003 0000 - 0x0003 7FFF	224 KB
7	32 KB	896 - 1023	0x0003 8000 - 0x0003 FFFF	256 KB

4.3.6 Code Read Protection (CRP)

Code Read Protection is a mechanism that allows the user to enable different levels of security in the system so that access to the on-chip flash and use of the ISP can be restricted. When needed, CRP is invoked by programming a specific pattern in the flash image at offset 0x0000 02FC. IAP commands are not affected by the code read protection.

Basic CRP options in USART and I2C/SPI ISP modes are detailed in the ISP command lists in Table 488. I2C and SPI ISP modes may alter these options slightly or may have slightly different commands and parameter sets. CRP permissions in USART and I2C/SPI ISP modes are detailed in the ISP command lists in the USART and I2C/SPI ISP sections.

Important: any CRP change becomes effective only after the device has gone through a power cycle.

Table 6. Code Read Protection (CRP) options

Name	Pattern programmed in 0x0000 02FC	Description
NO_ISP	0x4E69 7370	Prevents sampling of the pins for entering ISP mode. ISP sampling pin is available for other applications.
CRP1	0x1234 5678	Access to chip via the SWD pins is disabled. This mode allows partial flash update using the following ISP commands and restrictions: • Write to RAM command cannot access RAM below 0x2000 0300. • Copy RAM to flash command can not write to Sector 0. • Erase command can erase Sector 0 only when all sectors are selected for erase. • Compare command is disabled. • Read Memory command is disabled. This mode is useful when CRP is required and flash field updates are needed but all sectors can not be erased. Since compare command is disabled in case of partial updates the secondary loader should implement checksum mechanism to verify the integrity of the flash.
CRP2	0x8765 4321	Access to chip via the SWD pins is disabled. The following ISP commands are disabled: • Read Memory • Write to RAM • Go • Copy RAM to flash • Compare When CRP2 is enabled the ISP erase command only allows erasure of all user sectors.
CRP3	0x4321 8765	Access to chip via the SWD pins is disabled. ISP entry selected via the ISP entry pin is disabled if a valid user code is present in flash sector 0. This mode effectively disables ISP override using the entry pin. It is up to the user's application to provide a flash update mechanism using IAP calls or call reinvoke ISP command to enable flash update via USART. **Caution: If CRP3 is selected, no future factory testing can be performed on the device.**

In case a CRP mode is enabled and access to the chip is allowed via the ISP, an unsupported or restricted ISP command will be terminated with return code CODE_READ_PROTECTION_ENABLED.

4.3.6.1 ISP entry protection

In addition to the three CRP modes, the user can prevent the sampling of the pin for entering ISP mode and thereby release the pin for other applications. This is called the NO_ISP mode. The NO_ISP mode can be entered by programming the pattern 0x4E69 7370 at location 0x0000 02FC.

The NO_ISP mode is identical to the CRP3 mode except for SWD access, which is allowed in NO_ISP mode but disabled in CRP3 mode. The NO_ISP mode does not offer any code protection.

4.3.7 ISP interrupt and SRAM use

4.3.7.1 Interrupts during IAP

The on-chip flash memory is not accessible during erase/write operations. When the user application code starts executing, the interrupt vectors from the user flash area are active. Before making any IAP call, either disable the interrupts or ensure that the user interrupt vectors are active in RAM and that the interrupt handlers reside in RAM. The IAP code does not use or disable interrupts.

4.3.7.2 RAM used by ISP command handlers

Memory for the USART and I2C/SPI ISP commands is allocated dynamically.

4.3.7.3 RAM used by IAP command handler

Flash programming commands use the top 32 bytes of on-chip SRAM0 (see Section 2.1.1 for details of the SRAM configuration). This corresponds to addresses 0x2000 FFE0 through 0x2000 FFFF. The maximum stack usage in the user allocated stack space is 128 bytes and grows downwards.

4.4 USART ISP communication protocol

All USART ISP commands should be sent as single ASCII strings. Strings should be terminated with Carriage Return (CR) and/or Line Feed (LF) control characters. Extra <CR> and <LF> characters are ignored. All ISP responses are sent as <CR><LF> terminated ASCII strings. Data is sent and received in plain binary format.

4.4.1 USART ISP initialization

Once the USART ISP mode is entered, the auto-baud routine needs to synchronize with the host via the serial port (USART).

The host should send a ’?’ (0x3F) as a synchronization character and wait for a response. The host side serial port settings should be 8 data bits, 1 stop bit and no parity. The auto-baud routine measures the bit time of the received synchronization character in terms of its own frequency and programs the baud rate generator of the serial port. It also sends an ASCII string ("Synchronized<CR><LF>") to the host. In response to this, the host should send back the same string ("Synchronized<CR><LF>").

The auto-baud routine looks at the received characters to verify synchronization. If synchronization is verified then "OK<CR><LF>" string is sent to the host. The host should respond by sending the crystal frequency (in kHz) at which the part is running. The response is required for backward compatibility of the boot loader code and is ignored. "OK<CR><LF>" string is sent to the host after receiving the crystal frequency. If synchronization is not verified then the auto-baud routine waits again for a synchronization character. For auto-baud to work correctly in case of user invoked ISP, the clock frequency should be greater than or equal to 10 MHz. In USART ISP mode, the part is clocked by the FRO 12 MHz and the crystal frequency is ignored.

Once the crystal frequency is received the part is initialized and the ISP command handler is invoked. For safety reasons an "Unlock" command is required before executing the commands resulting in flash erase/write operations and the "Go" command. The rest of the commands can be executed without the unlock command. The Unlock command is required to be executed once per ISP session. The Unlock command is explained in Section 4.5 “USART ISP commands”.

4.4.2 USART ISP command format

"Command Parameter_0 Parameter_1 ... Parameter_n<CR><LF>" "Data" (Data only for Write commands).

4.4.3 USART ISP response format

"Return_Code<CR><LF>Response_0<CR><LF>Response_1<CR><LF> ... Response_n<CR><LF>" "Data" (Data only for Read commands).

4.4.4 USART ISP data format

The data stream is in plain binary format.

4.5 USART ISP commands

The following commands are accepted by the ISP command handler. Detailed status codes are supported for each command. The command handler sends the return code INVALID_COMMAND when an undefined command is received. Commands and return codes are in ASCII format.

CMD_SUCCESS is sent by ISP command handler only when received ISP command has been completely executed and the new ISP command can be given by the host. Exceptions from this rule are "Set Baud Rate", "Write to RAM", "Read Memory", and "Go" commands.

Table 7. USART ISP command summary

ISP Command	Usage	Section
Unlock	U <Unlock Code>	4.5.1
Set Baud Rate	B <Baud Rate> <stop bit>	4.5.2
Echo	A <setting>	4.5.3
Write to RAM	W <start address> <number of bytes>	4.5.4
Read Memory	R <address> <number of bytes>	4.5.5
Prepare sectors for write operation	P <start sector number> <end sector number>	4.5.6
Copy RAM to flash	C <Flash address> <RAM address> <number of bytes>	4.5.7
Go	G <address> <Mode>	4.5.8

Table 7. USART ISP command summary

ISP Command	Usage	Section
Erase sector(s)	E <start sector number> <end sector number>	4.5.9
Erase page(s)	X <start page number> <end page number>	4.5.10
Blank check sector(s)	I <start sector number> <end sector number>	4.5.11
Read Part ID	J	4.5.12
Read Boot code version	K	4.5.13
Compare	M <address1> <address2> <number of bytes>	4.5.14
ReadUID	N	4.5.15
Read CRC checksum	S <address> <number of bytes>	4.5.16
Read flash signature	Z	4.5.17

Table 8 lists the supported USART ISP commands for each CRP level.

Table 8. ISP commands allowed for different CRP levels

ISP command	CRP1	CRP2	CRP3 (no entry in ISP mode allowed)
Unlock	yes	yes	n/a
Set Baud Rate	yes	yes	n/a
Echo	yes	yes	n/a
Write to RAM	yes; above 0x0200 0300 only	no	n/a
Read Memory	no	no	n/a
Prepare sectors for write operation	yes	yes	n/a
Copy RAM to flash[1]	yes; not to sector 0	no	n/a
Go	no	no	n/a
Erase sector(s)[1]	yes; sector 0 can only be erased when all sectors are erased.	yes; all sectors only	n/a
Erase page(s)[1]	yes; page 0 can only be erased when all pages are erased (not recommended, use Erase Sector).	yes; all pages only	n/a
Blank check sectors	no	no	n/a
Read Part ID	yes	yes	n/a
Read Boot code version	yes	yes	n/a
Compare	no	no	n/a
ReadUID	yes	yes	n/a
Read CRC	no	no	n/a
Read flash signature	yes	yes	n/a

[1] It is not secure to use CRP1 or CRP2 when using Dual Enhanced image types. Use CRP3 instead.

4.5.1 Unlock

Table 9. USART ISP Unlock command

Command	U
Input	Unlock code: 23130_{10}
Return Code	CMD_SUCCESS | INVALID_CODE | PARAM_ERROR
Description	This command is used to unlock Flash Write, Erase, and Go commands.
Example	"U 23130<CR><LF>" unlocks the Flash Write/Erase & Go commands.

4.5.2 Set Baud Rate

Table 10. USART ISP Set Baud Rate command

Command	B
Input	Baud Rate: 9600 | 19200 | 38400 | 57600 | 115200 Stop bit: 1 | 2
Return Code	CMD_SUCCESS | INVALID_BAUD_RATE | INVALID_STOP_BIT | PARAM_ERROR
Description	This command is used to change the baud rate. The new baud rate is effective after the command handler sends the CMD_SUCCESS return code.
Example	"B 57600 1<CR><LF>" sets the serial port to baud rate 57600 bps and 1 stop bit.

4.5.3 Echo

Table 11. USART ISP Echo command

Command	A
Input	Setting: ON = 1 | OFF = 0
Return Code	CMD_SUCCESS | PARAM_ERROR
Description	The default setting for echo command is ON. When ON the ISP command handler sends the received serial data back to the host.
Example	"A 0<CR><LF>" turns echo off.

4.5.4 Write to RAM

The host should send the plain binary code after receiving the CMD_SUCCESS return code. This ISP command handler responds with “OK<CR><LF>” when the transfer has finished.

Table 12. USART ISP Write to RAM command

Command	W
Input	**Start Address:** On-chip RAM address where data bytes are to be written. This address should be a word boundary. **Number of Bytes:** Number of bytes to be written. Count should be a multiple of 4
Return Code	CMD_SUCCESS | ADDR_ERROR (Address not on word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not multiple of 4) | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to download data to on-chip RAM. This command is blocked when code read protection levels 2 or 3 are enabled. Writing
Example	"W 33555200 4<CR><LF>" writes 4 bytes of data to address 0x0200 0300.

4.5.5 Read Memory

Reads the plain binary code of the data stream, followed by the CMD_SUCCESS return code.

Table 13. USART ISP Read Memory command

Command	R
Input	**Start Address:** Address from where data bytes are to be read. This address should be a word boundary. **Number of Bytes:** Number of bytes to be read. Count should be a multiple of 4.
Return Code	CMD_SUCCESS followed by <actual data (plain binary)> | ADDR_ERROR (Address not on word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not a multiple of 4) | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to read data from on-chip RAM or flash memory. This command is blocked when code read protection is enabled.
Example	"R 33554432 4<CR><LF>" reads 4 bytes of data from address 0x0200 0000.

4.5.6 Prepare sectors for write operation

This command makes flash write/erase operation a two-step process.

Table 14. USART ISP Prepare sectors for write operation command

Command	P
Input	**Start Sector Number** **End Sector Number:** Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | BUSY | INVALID_SECTOR | PARAM_ERROR
Description	This command must be executed before executing "Copy RAM to flash" or "Erase Sector(s)" command. Successful execution of the "Copy RAM to flash" or "Erase Sector(s)" command causes relevant sectors to be protected again. To prepare a single sector use the same "Start" and "End" sector numbers.
Example	"P 0 0<CR><LF>" prepares the flash sector 0.

4.5.7 Copy RAM to flash

When writing to the flash, the following limitations apply:

1. The smallest amount of data that can be written to flash by the copy RAM to flash command is 256 byte (equal to one page).
2. One page consists of 16 flash words (lines), and the smallest amount that can be modified per flash write is one flash word (one line). This limitation exists because ECC is applied during the flash write operation, see Section 4.3.3.
3. To avoid write disturbance (a mechanism intrinsic to flash memories), an erase should be performed after 16 consecutive writes inside the same page. Note that the erase operation then erases the entire sector.

   Remark: Once a page has been written to 16 times, it is still possible to write to other pages within the same sector without performing a sector erase (assuming that those pages have been erased previously).

Table 15. USART ISP Copy command

Command	C
Input	**Flash Address(DST):** Destination flash address where data bytes are to be written. The destination address should be a 256 byte boundary. **RAM Address(SRC):** Source on-chip RAM address from where data bytes are to be read. **Number of Bytes:** Number of bytes to be written. Should be 256 | 512 | 1024 | 4096.
Return Code	CMD_SUCCESS | SRC_ADDR_ERROR (Address not on word boundary) | DST_ADDR_ERROR (Address not on correct boundary) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not 256 | 512 | 1024 | 4096) | SECTOR_NOT_PREPARED_FOR WRITE_OPERATION | BUSY | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to program the flash memory. The "Prepare Sector(s) for Write Operation" command should precede this command. The affected sectors are automatically protected again once the copy command is successfully executed. This command is blocked when code read protection is enabled. Also see Section 4.3.3 for the number of bytes that can be written.
Example	"C 0 33556480 512<CR><LF>" copies 512 bytes from the RAM address 0x0200 0800 to the flash address 0.

4.5.8 Go

Table 16. USART ISP Go command

Command	**G**
Input	**Address:** Flash or on-chip RAM address from which the code execution is to be started. This address should be on a word boundary. **Mode:** T (Execute program in Thumb Mode) | A (Execute program in ARM mode).
Return Code	CMD_SUCCESS | ADDR_ERROR | ADDR_NOT_MAPPED | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to execute a program residing in RAM or flash memory. It may not be possible to return to the ISP command handler once this command is successfully executed. This command is blocked when code read protection is enabled.
Example	"G 0 A<CR><LF>" branches to address 0x0000 0000 in ARM mode.

4.5.9 Erase sectors

Table 17. USART ISP Erase sector command

Command	**E**
Input	**Start Sector Number** **End Sector Number:** Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | BUSY | INVALID_SECTOR | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to erase one or more sector(s) of on-chip flash memory. This command only allows erasure of all user sectors when the code read protection is enabled.
Example	"E 2 3<CR><LF>" erases the flash sectors 2 and 3.

4.5.10 Erase pages

Table 18. USART ISP Erase page command

Command	**X**
Input	**Start Page Number** **End Page Number:** Should be greater than or equal to start page number.
Return Code	CMD_SUCCESS | BUSY | INVALID_PAGE | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to erase one or more page(s) of on-chip flash memory.
Example	"X 2 3<CR><LF>" erases the flash pages 2 and 3.

4.5.11 Blank check sectors

Table 19. USART ISP Blank check sector command

Command	**I**
Input	**Start Sector Number:** **End Sector Number:** Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | SECTOR_NOT_BLANK (followed by <Offset of the first non blank word location> <Contents of non blank word location>) | INVALID_SECTOR | PARAM_ERROR
Description	This command is used to blank check one or more sectors of on-chip flash memory. When CRP is enabled, the blank check command returns 0 for the offset and value of sectors which are not blank. Blank sectors are correctly reported irrespective of the CRP setting.
Example	"I 2 3<CR><LF>" blank checks the flash sectors 2 and 3.

4.5.12 Read Part Identification number

Table 20. USART ISP Read Part Identification command

Command	**J**
Input	None.
Return Code	CMD_SUCCESS followed by part identification number (see Table 21).
Description	This command is used to read the part identification number.

Table 21. LPC51U68 device identification numbers

Device	**Hex coding**
LPC51U68J BD64	0x06451B68
LPC51U68J BD48	0x06451A68

4.5.13 Read Boot code version number

Table 22. USART ISP Read Boot Code version number command

Command	**K**
Input	None
Return Code	CMD_SUCCESS followed by 2 bytes of boot code version number in ASCII format. It is to be interpreted as <byte1(Major)>.<byte0(Minor)>.
Description	This command is used to read the boot code version number.

4.5.14 Compare

Table 23. USART ISP Compare command

Command	M
Input	**Address1 (DST):** Starting flash or on-chip RAM address of data bytes to be compared. This address should be a word boundary. **Address2 (SRC):** Starting flash or on-chip RAM address of data bytes to be compared. This address should be a word boundary. **Number of Bytes:** Number of bytes to be compared; should be a multiple of 4.
Return Code	CMD_SUCCESS | (Source and destination data are equal) | COMPARE_ERROR | (Followed by the offset of first mismatch) | COUNT_ERROR (Byte count is not a multiple of 4) | ADDR_ERROR | ADDR_NOT_MAPPED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to compare the memory contents at two locations. **Compare result may not be correct when source or destination address contains any of the first 512 bytes starting from address zero. First 512 bytes are re-mapped to boot ROM**
Example	"M 8192 33587200 4<CR><LF>" compares 4 bytes from the RAM address 0x0200 8000 to the 4 bytes from the flash address 0x2000.

4.5.15 ReadUID

Table 24. USART ReadUID command

Command	N
Input	None
Return Code	CMD_SUCCESS followed by four 32-bit words of a unique serial number in ASCII format. The word sent at the lowest address is sent first.
Description	This command is used to read the unique ID.

4.5.16 Read CRC checksum

Get the CRC checksum of a block of RAM or flash. CMD_SUCCESS followed by 8 bytes of CRC checksum in decimal format.

The checksum is calculated as follows:

CRC-32 polynomial: x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1

Seed Value: 0xFFFF FFFF

Table 25. USART ISP Read CRC checksum command

Command	S
Input	**Address:** The data are read from this address for CRC checksum calculation. This address must be on a word boundary. **Number of Bytes:** Number of bytes to be calculated for the CRC checksum; must be a multiple of 4.
Return Code	CMD_SUCCESS followed by data in decimal format | ADDR_ERROR (address not on word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (byte count is not a multiple of 4) | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to read the CRC checksum of a block of on-chip RAM or flash memory. This command is blocked when code read protection is enabled.
Example	"S 33587200 4<CR><LF>" reads the CRC checksum for 4 bytes of data from address 0x0200 8000. If checksum value is 0xCBF43926, then the host will receive: "3421780262 <CR><LF>"

4.5.17 Read flash signature

Get the signature for the entire flash memory using an internal flash signature generator (see Chapter 33). CMD_SUCCESS followed by the 128-bit flash signature represented in decimal format.

Table 26. USART ISP Read flash signature command

Command	Z
Input	none
Return Code	CMD_SUCCESS followed by data in decimal format | CODE_READ_PROTECTION_ENABLED
Description	This command is used to read the signature of the entire flash memory. This command is blocked when code read protection is enabled.
Example	"Z<CR><LF>" returns the signature for the entire flash memory. If signature value is 0x3BD7, then the host will receive: "15319 <CR><LF>"

4.5.18 UART ISP Error codes

Table 27. USART ISP Error codes

Return Code	Error code	Description
0x0	ERR_ISP_CMD_SUCCESS	Command is executed successfully. Sent by ISP handler only when command given by the host has been completely and successfully executed.
0x1	ERR_ISP_INVALID_COMMAND	Invalid command.
0x2	ERR_ISP_SRC_ADDR_ERROR	Source address is not on word boundary.
0x3	ERR_ISP_DST_ADDR_ERROR	Destination address is not on a correct boundary.
0x4	ERR_ISP_SRC_ADDR_NOT_MAPPED	Source address is not mapped in the memory map. Count value is taken into consideration where applicable.
0x5	ERR_ISP_DST_ADDR_NOT_MAPPED	Destination address is not mapped in the memory map. Count value is taken into consideration where applicable.
0x6	ERR_ISP_COUNT_ERROR	Byte count is not multiple of 4 or is not a permitted value.
0x7	ERR_ISP_INVALID_SECTOR	Sector number is invalid or end sector number is greater than start sector number.
0x8	ERR_ISP_SECTOR_NOT_BLANK	Sector is not blank.
0x9	ERR_ISP_SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	Command to prepare sector for write operation was not executed.
0xA	ERR_ISP_COMPARE_ERROR	Source and destination data not equal.
0xB	ERR_ISP_BUSY	Flash programming hardware interface is busy.
0xC	ERR_ISP_PARAM_ERROR	Insufficient number of parameters or invalid parameter.
0xD	ERR_ISP_ADDR_ERROR	Address is not on word boundary.
0xE	ERR_ISP_ADDR_NOT_MAPPED	Address is not mapped in the memory map. Count value is taken into consideration where applicable.
0xF	ERR_ISP_CMD_LOCKED	Command is locked.
0x10	ERR_ISP_INVALID_CODE	Unlock code is invalid.
0x11	ERR_ISP_INVALID_BAUD_RATE	Invalid baud rate setting.
0x12	ERR_ISP_INVALID_STOP_BIT	Invalid stop bit setting.
0x13	ERR_ISP_CODE_READ_PROTECTION_ENABLED	Code read protection enabled.
0x14	-	Reserved.
0x15	-	Reserved.
0x16	-	Reserved.
0x17	ERR_ISP_FRO_NO_POWER	FRO not turned on in the PDRUNCFG register.
0x18	ERR_ISP_FLASH_NO_POWER	Flash not turned on in the PDRUNCFG register.
0x19	-	Reserved.
0x1A	-	Reserved.
0x1B	ERR_ISP_FLASH_NO_CLOCK	Flash clock disabled in the AHBCLKCTRL register.
0x1C	ERR_ISP_REINVOKE_ISP_CONFIG	Reinvoke ISP not successful.

```
typedef enum
{
 ERR_ISP_BASE = 0x00000000,
 /*0x00000001*/ ERR_ISP_INVALID_COMMAND = ERR_ISP_BASE + 1,
 /*0x00000002*/ ERR_ISP_SRC_ADDR_ERROR,
 /*0x00000003*/ ERR_ISP_DST_ADDR_ERROR,
 /*0x00000004*/ ERR_ISP_SRC_ADDR_NOT_MAPPED,
 /*0x00000005*/ ERR_ISP_DST_ADDR_NOT_MAPPED,
 /*0x00000006*/ ERR_ISP_COUNT_ERROR,
 /*0x00000007*/ ERR_ISP_INVALID_SECTOR,
 /*0x00000008*/ ERR_ISP_SECTOR_NOT_BLANK,
 /*0x00000009*/ ERR_ISP_SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION,
 /*0x0000000A*/ ERR_ISP_COMPARE_ERROR,
 /*0x0000000B*/ ERR_ISP_BUSY, /* Flash programming hardware interface is busy */
 /*0x0000000C*/ ERR_ISP_PARAM_ERROR, /* Insufficient number of parameters */
 /*0x0000000D*/ ERR_ISP_ADDR_ERROR, /* Address not on word boundary */
 /*0x0000000E*/ ERR_ISP_ADDR_NOT_MAPPED,
 /*0x0000000F*/ ERR_ISP_CMD_LOCKED, /* Command is locked */
 /*0x00000010*/ ERR_ISP_INVALID_CODE, /* Unlock code is invalid */
 /*0x00000011*/ ERR_ISP_INVALID_BAUD_RATE,
 /*0x00000012*/ ERR_ISP_INVALID_STOP_BIT,
 /*0x00000013*/ ERR_ISP_CODE_READ_PROTECTION_ENABLED,
 /*0x00000014*/ ERR_ISP_INVALID_FLASH_UNIT, /* reserved */
 /*0x00000015*/ ERR_ISP_USER_CODE_CHECKSUM, /* reserved */
 /*0x00000016*/ ERR_ISP_SETTING_ACTIVE_PARTITION, /* reserved */
 /*0x00000017*/ ERR_ISP_FRO_NO_POWER,
 /*0x00000018*/ ERR_ISP_FLASH_NO_POWER,
 /*0x00000019*/ ERR_ISP_EEPROM_NO_POWER, /* reserved */
 /*0x0000001A*/ ERR_ISP_EEPROM_NO_CLOCK, /* reserved */
 /*0x0000001B*/ ERR_ISP_FLASH_NO_CLOCK,
 /*0x0000001C*/ ERR_ISP_REINVOKE_ISP_CONFIG
} ErrorCode_t;
```

4.6 IAP commands

For in application programming the IAP routine should be called with a word pointer in register r0 pointing to memory (RAM) containing command code and parameters. The result of the IAP command is returned in the result table pointed to by register r1. The user can reuse the command table for result by passing the same pointer in registers r0 and r1. The parameter table should be big enough to hold all the results in case the number of results are more than number of parameters. Parameter passing is illustrated in the Figure 7.

The number of parameters and results vary according to the IAP command. The maximum number of parameters is 5, passed to the "Copy RAM to FLASH" command. The maximum number of results is 5, returned by the "ReadUID" command. The command handler sends the status code INVALID_COMMAND when an undefined command is received. The IAP routine resides at location 0x03000204 and it is thumb code, therefore called as 0x03000205 by the Cortex-M0+ to insure Thumb operation.

The IAP function could be called in the following way using C:

Define the IAP location entry point. Since the least significant bit of the IAP location is set there will be a change to Thumb instruction set if called by the Cortex-M0+.

```
#define IAP_LOCATION 0x03000205
```

Define data structure or pointers to pass IAP command table and result table to the IAP function:

```
unsigned int command_param[5];
unsigned int status_result[5];
```

or

```
unsigned int * command_param;
unsigned int * status_result;
command_param = (unsigned int *) 0x...
status_result =(unsigned int *) 0x...
```

Define pointer to function type, which takes two parameters and returns void. Note the IAP returns the result with the base address of the table residing in R1.

```
typedef void (*IAP)(unsigned int [],unsigned int[]);
IAP iap_entry;
```

Setting the function pointer:

```
#define IAP_LOCATION 0x0300 0205
iap_entry=(IAP) IAP_LOCATION;
```

To call the IAP use the following statement.

```
iap_entry (command_param,status_result);
```

Up to 4 parameters can be passed in the r0, r1, r2 and r3 registers respectively (see the *ARM Thumb Procedure Call Standard SWS ESPC 0002 A-05)*. Additional parameters are passed on the stack. Up to 4 parameters can be returned in the r0, r1, r2 and r3 registers respectively. Additional parameters are returned indirectly via memory. Some of the IAP calls require more than 4 parameters. If the ARM suggested scheme is used for the parameter passing/returning then it might create problems due to difference in the C compiler implementation from different vendors. The suggested parameter passing scheme reduces such risk.

The flash memory is not accessible during a write or erase operation. IAP commands, which results in a flash write/erase operation, use 32 bytes of space in the top portion of the on-chip RAM for execution. The user program should not be use this space if IAP flash programming is permitted in the application.

Table 28. IAP Command Summary

IAP Command	Command code	Section
Prepare sector(s) for write operation	50 (decimal)	4.6.1
Copy RAM to flash	51 (decimal)	4.6.2
Erase sector(s)	52 (decimal)	4.6.3
Blank check sector(s)	53 (decimal)	4.6.4
Read Part ID	54 (decimal)	4.6.5
Read Boot code version	55 (decimal)	4.6.6
Compare	56 (decimal)	4.6.7
Reinvoke ISP	57 (decimal)	4.6.8
Read UID	58 (decimal)	4.6.9
Erase page(s)	59 (decimal)	4.6.10
Read Signature	70 (decimal)	4.6.11

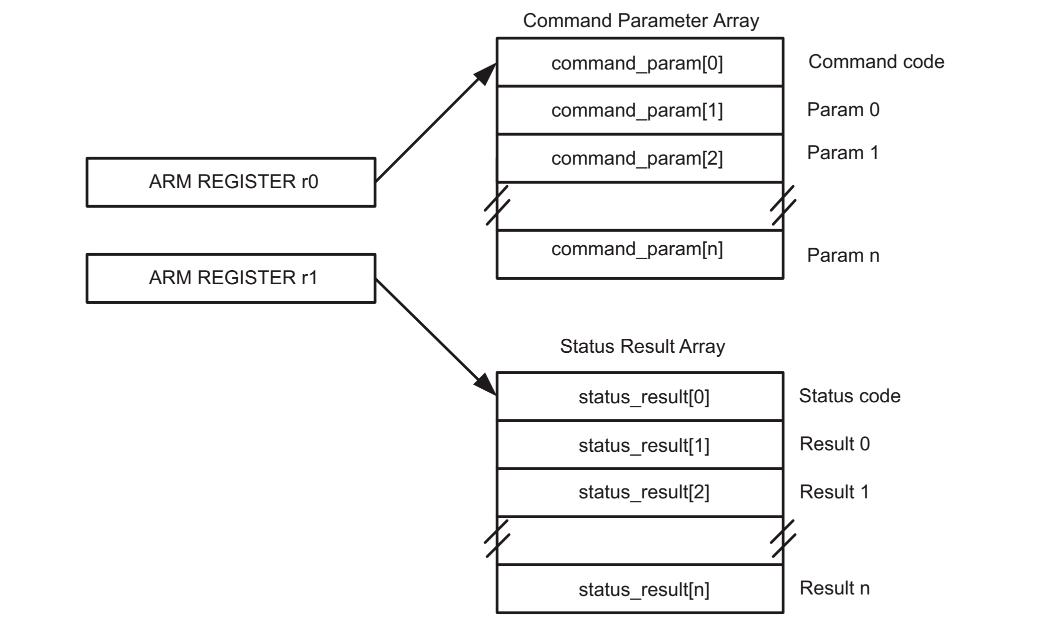


Fig 7. IAP parameter passing

4.6.1 Prepare sector(s) for write operation

This command makes flash write/erase operation a two step process.

Table 29. IAP Prepare sector(s) for write operation command

Command	Prepare sector(s) for write operation
Input	**Command code: 50 (decimal)** **Param0:** Start Sector Number **Param1:** End Sector Number (should be greater than or equal to start sector number).
Status code	CMD_SUCCESS | BUSY | INVALID_SECTOR
Result	None
Description	This command must be executed before executing "Copy RAM to flash" or "Erase Sector(s)" command. Successful execution of the "Copy RAM to flash" or "Erase Sector(s)" command causes relevant sectors to be protected again. To prepare a single sector use the same "Start" and "End" sector numbers.

4.6.2 Copy RAM to flash

See Section 4.5.7 for limitations on the write-to-flash process.

Table 30. IAP Copy RAM to flash command

Command	Copy RAM to flash
Input	**Command code: 51 (decimal)** **Param0(DST):** Destination flash address where data bytes are to be written. This address should be a 256 byte boundary. **Param1(SRC):** Source RAM address from which data bytes are to be read. This address should be a word boundary. **Param2:** Number of bytes to be written. Should be 256 | 512 | 1024 | 4096. **Param3:** System Clock Frequency (CCLK) in kHz.
Status code	CMD_SUCCESS | SRC_ADDR_ERROR (Address not a word boundary) | DST_ADDR_ERROR (Address not on correct boundary) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not 256 | 512 | 1024 | 4096) | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | BUSY
Result	None
Description	This command is used to program the flash memory. The affected sectors should be prepared first by calling "Prepare Sector for Write Operation" command. The affected sectors are automatically protected again once the copy command is successfully executed. Also see Section 4.3.3 for the number of bytes that can be written. **Remark:** All user code must be written in such a way that no master accesses the flash while this command is executed and the flash is programmed.

4.6.3 Erase Sector(s)

Table 31. IAP Erase Sector(s) command

Command	**Erase Sector(s)**
Input	**Command code: 52 (decimal)** **Param0:** Start Sector Number **Param1:** End Sector Number (should be greater than or equal to start sector number). **Param2:** System Clock Frequency (CCLK) in kHz.
Status code	CMD_SUCCESS | BUSY | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | INVALID_SECTOR
Result	None
Description	This command is used to erase a sector or multiple sectors of on-chip flash memory. To erase a single sector use the same "Start" and "End" sector numbers. **Remark:** All user code must be written in such a way that no master accesses the flash while this command is executed and the flash is erased.

4.6.4 Blank check sector(s)

Table 32. IAP Blank check sector(s) command

Command	**Blank check sector(s)**
Input	**Command code: 53 (decimal)** **Param0:** Start Sector Number **Param1:** End Sector Number (should be greater than or equal to start sector number).
Status code	CMD_SUCCESS | BUSY | SECTOR_NOT_BLANK | INVALID_SECTOR
Result	**Result0:** Offset of the first non blank word location if the status code is SECTOR_NOT_BLANK. **Result1:** Contents of non blank word location.
Description	This command is used to blank check a sector or multiple sectors of on-chip flash memory. To blank check a single sector use the same "Start" and "End" sector numbers.

4.6.5 Read Part Identification number

Table 33. IAP Read Part Identification command

Command	**Read part identification number**
Input	**Command code: 54 (decimal)** **Parameters:** None
Status code	CMD_SUCCESS
Result	**Result0:** Part Identification Number.See Table 21 "LPC51U68 device identification numbers".
Description	This command is used to read the part identification number.

4.6.6 Read Boot code version number

Table 34. IAP Read Boot Code version number command

Command	**Read boot code version number**
Input	**Command code: 55 (decimal)** **Parameters:** None
Status code	CMD_SUCCESS
Result	**Result0:** 2 bytes of boot code version number. Read as <byte1(Major)>.<byte0(Minor)>
Description	This command is used to read the boot code version number.

4.6.7 Compare <address1> <address2> <no of bytes>

Table 35. IAP Compare command

Command	**Compare**
Input	**Command code: 56 (decimal)** **Param0(DST):** Starting flash or RAM address of data bytes to be compared; should be a word boundary. **Param1(SRC):** Starting flash or RAM address of data bytes to be compared; should be a word boundary. **Param2:** Number of bytes to be compared; should be a multiple of 4.
Status code	CMD_SUCCESS | COMPARE_ERROR | COUNT_ERROR (Byte count is not a multiple of 4) | ADDR_ERROR | ADDR_NOT_MAPPED
Result	**Result0:** Offset of the first mismatch if the status code is COMPARE_ERROR.
Description	This command is used to compare the memory contents at two locations. **The result may not be correct when the source or destination includes any of the first 512 bytes starting from address zero. The first 512 bytes can be re-mapped to RAM.**

4.6.8 Reinvoke ISP

Table 36. Reinvoke ISP

Command	**Compare**
Input	**Command code: 57 (decimal)** **Param0(mode):** Selected ISP mode. 1=USART, 2=USB, 5=I2C, 6=SPI
Status code	ERR_ISP_REINVOKE_ISP_CONFIG
Result	None.
Description	This command is used to invoke the boot loader in ISP mode. It maps boot vectors and configures the peripherals for ISP. This command may be used when a valid user program is present in the internal flash memory and the ISP entry pin are not accessible to force the ISP mode. If using USART ISP mode, enable the clocks to the default USART RXD and TXD pins before calling this command.

4.6.9 ReadUID

Table 37. IAP ReadUID command

Command	Compare
Input	**Command code: 58 (decimal)**
Status code	CMD_SUCCESS
Result	**Result0:** The first 32-bit word (at the lowest address). **Result1:** The second 32-bit word. **Result2:** The third 32-bit word. **Result3:** The fourth 32-bit word.
Description	This command is used to read the unique ID.

4.6.10 Erase page

Table 38. IAP Erase page command

Command	Erase page
Input	**Command code: 59 (decimal)** **Param0:** Start page number. **Param1:** End page number (should be greater than or equal to start page) **Param2:** System Clock Frequency (CCLK) in kHz.
Status code	CMD_SUCCESS | BUSY | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | INVALID_PAGE
Result	None
Description	This command is used to erase a page or multiple pages of on-chip flash memory. To erase a single page use the same "start" and "end" page numbers. **Remark:** All user code must be written in such a way that no master accesses the flash while this command is executed and the flash is erased.

4.6.11 Read Signature

Table 39. IAP Read Signature command

Command	Read Signature
Input	**Command code: 70 (decimal)**
Status code	CMD_SUCCESS
Result	**Result0:** The 32-bit generated signature.
Description	This command is used to obtain a 32-bit signature value of the entire flash memory. See Section 4.5.17 “Read flash signature” and Chapter 33 for more information. **Remark:** To insure that the flash signature is generated correctly, some setup must first be done as described in paragraph 2 of Section 33.5.1.1 “Signature generation”.

4.6.12 IAP Status Codes

Table 40. IAP Status codes Summary

Status code	Mnemonic	Description
0	CMD_SUCCESS	Command is executed successfully.
1	INVALID_COMMAND	Invalid command.
2	SRC_ADDR_ERROR	Source address is not on a word boundary.
3	DST_ADDR_ERROR	Destination address is not on a correct boundary.
4	SRC_ADDR_NOT_MAPPED	Source address is not mapped in the memory map. Count value is taken into consideration where applicable.
5	DST_ADDR_NOT_MAPPED	Destination address is not mapped in the memory map. Count value is taken into consideration where applicable.
6	COUNT_ERROR	Byte count is not multiple of 4 or is not a permitted value.
7	INVALID_SECTOR	Sector number is invalid.
8	SECTOR_NOT_BLANK	Sector is not blank.
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	Command to prepare sector for write operation was not executed.
10	COMPARE_ERROR	Source and destination data is not same.
11	BUSY	Flash programming hardware interface is busy.

4.7 I2C and SPI ISP commands

The LPC51U68 I2C/SPI ISP allows programming and reprogramming internal flash via a set of commands on the I2C slave or SPI slave buses of the LPC51U68. These need to be connected to a host system that provides the I2C or SPI master connections to the LPC51U68.

For I2C and SPI ISP modes, the LPC51U68 will enable both the I2C and SPI interfaces and select the first one that has a valid probe message.

4.7.1 Dual purpose of the ISP1/ISP1_IRQ pin

The ISP1 pin is a special function pin that is switches function state once I2C/SPI ISP mode is entered and the host interface has been selected. Once the host interface has been selected, the ISP1 pin becomes an output pin used for indicating to the host system that a command can be serviced. During this phase, the pin is called ISP1_IRQ. A low state on this pin indicates to the host that the LPC51U68 needs servicing.

Notes on ISP1 input to ISP1_IRQ output switching

After reset, ISP1 is set to an input. The ISP1 pin switches to the output pin, ISP1_IRQ, only when the host interface (either I2C or SPI) is selected. On entry to the I2C/SPI ISP mode from device reset, the interface is in auto-detection mode and the pin remains an input. Once the host interface sends the first probe command and it's accepted by the LPC51U68, the ISP1 pin switches to an output. When using the 'Reinvoke ISP' command, the host interface is selected as part of the IAP command, so the pin switches to an output right away without a probe message. When the pin switches to an output, it is driven high.

Care must be used that any external signal applied to the ISP1 pin to drive the ISP1 pin low should be done with a pull-down resistor and not tied to ground, so the ISP1_IRQ output doesn't short directly to a low state when driving high. Host systems that drive this pin to selectively enable I2C/SPI ISP mode should drive this pin via a resistor.

4.7.2 Selection of I2C or SPI interface for ISP mode

Only 1 interface – either I2C or SPI – can be active for an I2C/SPI ISP session. The active interface is selected by 2 possible methods: Automatic detection or assigned via the 'Re-invoke ISP mode' IAP command.

Automatic detection is used when the I2C/SPI pin selection hasn't been defined and is used when the device is reset and enters I2C/SPI ISP mode. For automatic detection, the I2C and SPI pins listed in Section 3.2 are monitored for data from the host machine. The first interface to get data (in the form of a probe message) from a host system will become the active interface. Once the interface has been activated, the ISP1 pin will switch to an output (ISP1_IRQ) driven high.

For the 'Re-invoke ISP mode' IAP command, the I2C or SPI interface is selected during the IAP function call. For the IAP call, auto-detection is not possible. The ISP1 pin will switch to an output (ISP1_IRQ) driven high when the IAP call is made.

4.7.3 I2C/SPI ISP mode transaction protocol

This section explains the high-level protocol used with the I2C and SPI interfaces. A typical transaction starts with the host sending a command packet, the LPC51U68 processing the command packet, the LPC51U68 optionally asserting the ISP1_IRQ line low when processing is complete, and then the host system getting the response packet. The LPC51U68 will hold the ISP1_IRQ pin asserted (low) until the host system requests the response packet. The process is shown in Figure 8. Not all commands may follow this protocol – some commands may have no response or may not assert the ISP1_IRQ pin.

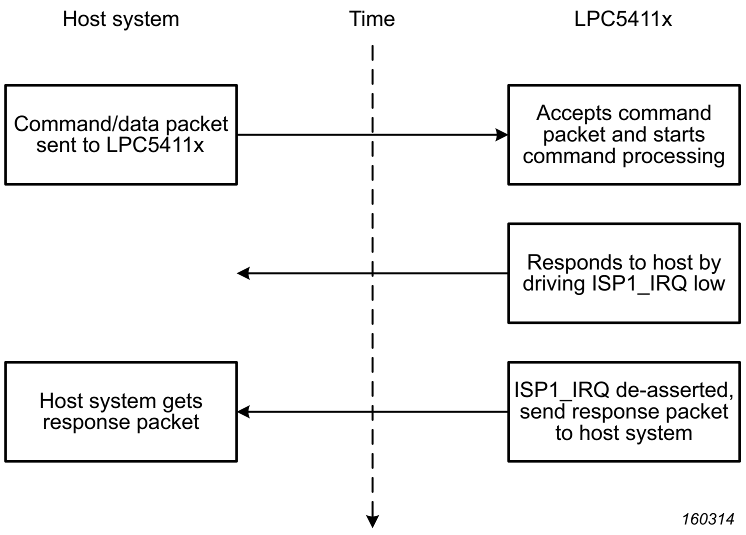

Host system
Time
LPC5411x
Command/data packet sent to LPC5411x
Accepts command packet and starts command processing
Responds to host by driving ISP1_IRQ low
Host system gets response packet
ISP1_IRQ de-asserted, send response packet to host system
160314

Fig 8. Typical host system and LPC51U68 transaction

4.7.4 I2C ISP mode transaction protocol

The LPC51U68 will respond to a host system on I2C addresses 0x18, 0x1C, 0x30, and 0x38. The host system's I2C master clock rate can be as high as 1MHz. The LPC51U68 may extend the I2C clock to delay the I2C master if it needs more time to perform an operation.

When using I2C, the command is given to the LPC51U68 by an I2C write transaction. The entire command packet is accepted by the LPC51U68 and then processing starts. Once the LPC51U68 has completed processing, it may drive the ISP1_IRQ line low until the host system issues an I2C read transaction to get the response.

4.7.5 SPI ISP mode transaction protocol

The LPC51U68 will respond to a host system on the configured SPI interface. A transfer is started once SSEL goes low on the LPC51U68. The SPI clock to the LPC51U68 should not exceed 2MHz. SPI SSEL to first clock timing should not be less than 100uS. SPI transfer configuration should be SPI Mode 0 with 8 data bits.

Note SPI transactions are bi-directional. During the command packet phase, the host system should ignore the read data (send data from LPC51U68). During the response phase, the LPC51U68 will ignore the read data (send data from host).

Although SPI is bi-directional, the command and response packet phases only send data one way for each phase. During the command packet phase, a single SPI transfer occurs where the command and data is sent from the host system. In this phase, SSEL is asserted low, the command packet is sent, and then SSEL is deasserted. The host system then waits for ISP1_IRQ to go low from the LPC51U68. Once ISP1_IRQ goes low, the host system then asserts SSEL low, the response packet is read, and then SSEL is deasserted. The LPC51U68 will deassert ISP1_IRQ at the start of the response packet prior to the response packet completing transfer.

4.7.6 I2C/SPI operations allowed for CRP systems

If CRP is enabled (CRP1, CRP2, or CRP3), then the I2C/SPI commands may be limited in functionality. See Table 41 for limitations of I2C/SPI ISP commands when CRP is enabled.

If NO_ISP mode is used in the application, the I2C/SPI ISP mode cannot be started when the ISP0 pin is asserted. The only way to recover a system when this happens is to erase flash using the SWD interface or via the IAP commands. The application can also use the ‘Re-invoke ISP’ command to get back into the I2C/SPI ISP mode.

Table 41. ISP commands allowed for different CRP levels

ISP command	CRP1	CRP2	CRP3 (no entry in ISP mode allowed)
SH_CMD_GET_VERSION	yes	yes	n/a
SH_CMD_RESET	yes	yes	n/a
SH_CMD_BOOT	no	no	n/a
SH_CMD_CHECK_IMAGE	yes	yes	n/a
SH_CMD_PROBE	yes	yes	n/a
SH_CMD_WRITE_BLOCK	yes; not sector 0	yes; not sector 0	n/a
SH_CMD_READ_BLOCK	no	no	n/a
SH_CMD_SECTOR_ERASE	yes; not sector 0	yes; not sector 0	n/a
SH_CMD_PAGE_ERASE	yes; not sector 0	yes; not sector 0	n/a
SH_CMD_PAGE_WRITE	yes; not sector 0	yes; not sector 0	n/a
SH_CMD_PAGE_READ	no	no	n/a
SH_CMD_WRITE_SUBBLOCK	yes; not sector 0	yes; not sector 0	n/a
SH_CMD_READ_SUBBLOCK	no	no	n/a
SH_CMD_BULK_ERASE	yes; not sector 0	yes; full device only	n/a

4.8 I2C/SPI ISP mode commands, data, and responses

All of the supported commands, associated structures and data formats for those commands, and responses are explained in this section

Table 42. I2C/SPI ISP command summary

ISP Command	Command #	Section
Get Version	0xA1	4.8.1
Reset device	0xA2	4.8.2
Boot image	0xA3	4.8.3
Check image	0xA4	4.8.4
Probe	0xA5	4.8.5
Write block	0xA6	4.8.6
Read block	0xA7	4.8.7
Sector erase	0xA8	4.8.8
Page erase	0xA9	4.8.9
Page write	0xAA	4.8.10
Page read	0xAB	4.8.11
Write sub-block	0xAC	4.8.12
Read sub-block	0xAD	4.8.13
Bulk erase	0xAE	4.8.14

4.8.1 SH_CMD_GET_VERSION (0xA1) command

This command can be used to get the version number of the I2C/SPI ISP processor. This should not be confused with the ‘Read Boot code version’ included as part of the UART ISP and IAP.

Table 43. Command packet

Field	Offset	Size (bytes)	Value	Description
command	0x0	0x1	0xA1	'Get Version' command identifier

Table 44. Response packet

Field	Offset	Size (bytes)	Value	Description
sop	0x0	0x1	0x55	Start of packet identifier
command	0x1	0x1	0xA1	Processed command identifier
length	0x2	0x2	0x2	Length of the response packet
major	0x4	0x1	Version	Major version
minor	0x5	0x1	Version	Minor version

4.8.2 SH_CMD_RESET (0xA2) command

This command can be used to reset the LPC51U68. This command has no response.

Table 45. Command packet

Field	Offset	Size (bytes)	Value	Description
command	0x0	0x1	0xA2	'Reset' command identifier

4.8.3 SH_CMD_BOOT (0xA3) command

This command can be used to boot the application currently programmed into flash. This command only has a response if it cannot boot the application due to CRP level.

Table 46. Command packet

Field	Offset	Size (bytes)	Value	Description
command	0x0	0x1	0xA3	'Boot' command identifier
reserved	0x1	0x3	0x00	Reserved padding
imageAddress	0x4	0x4	Address	Boot address of image

Table 47. Response packet

Field	Offset	Size (bytes)	Value	Description
sop	0x0	0x1	0x55	Start of packet identifier
command	0x1	0x1	0xA3	Processed command identifier
length	0x2	0x2	0x4	Length of the response packet on an error
errorCode	0x4	0x4	Error code	0x00000013 for boot prevention due to CRP level

4.8.4 SH_CMD_CHECK_IMAGE (0xA4) command

This command can be used to check if the application currently programmed into flash has a valid CRC value. The command will compute the CRC32 value of the image at the passed address using the crc_len field in the application image header. If the checksum is valid, a response value of 0 is returned, otherwise the computed CRC32 value is returned.

Table 48. Command packet

Field	Offset	Size (bytes)	Value	Description
command	0x0	0x1	0xA4	'Check image' command identifier
reserved	0x1	0x3	0x00	Reserved padding
imageAddress	0x4	0x4	Address	Boot address of image to check

Table 49. Response packet

Field	Offset	Size (bytes)	Value	Description
sop	0x0	0x1	0x55	Start of packet identifier
command	0x1	0x1	0xA4	Processed command identifier
length	0x2	0x2	0x04	Length of the response packet
errorCode	0x4	0x4	CRC32 value	0 – If checksum value present in image header matches the computed value. 0xYYYYYYYY = computed CRC32. 0xFFFFFFFF = invalid flash address.

4.8.5 SH_CMD_PROBE (0xA5) command

The probe command is used to select either the I2C or SPI interface when in auto-detection mode. This command is required when booting into the I2C/SPI ISP mode from a reset condition. The probe command data will be accepted on the supported I2C or SPI ISP pins. Once the data has been checked, the ISP1_IRQ line is driven low. The interface that was used for the probe command becomes the active interface and the other interface is disabled.

The probe command is optional when the I2C or SPI ISP mode is re-invoked from an application using the ‘Re-invoke ISP’ IAP command. Only the interface selected with the ‘Re-invoke ISP’ IAP command will be active for the optional probe command.

The host system should repeatedly send the probe command to the LCP5410x via one of the supported interfaces until the LPC51U68 asserts the ISP1_IRQ pin low. Once the ISP1_IRQ has been asserted low by the LPC51U68, the probe message can stop. No response will be sent back to the host on the supported interface pins.

Note this command has no response. A low state on the ISP1_IRQ line indicates the probe command was accepted and processed.

Table 50. Command packet

Field	Offset	Size (bytes)	Value	Description
command	0x0	0x1	0xA4	‘Probe’ command identifier
ifSel	0x1	0x1		Host interface port. Must match selected interface used to the LPC51U68. 0 – I2C0 port 1 – SPI0 port
Reserved0	0x2	0x1		Set to 0x00
Reserved1	0x3	0x1		Set to 0x00
Reserved2	0x4	0x1		Set to 0x00
Reserved3	0x5	0x1		Set to 0x00
Reserved4	0x6	0x1		Set to 0x00
checksum	0x7	0x1		XOR of all the 7 bytes above.

4.8.6 SH_CMD_WRITE_BLOCK (0xA6) command

The write block command is used to write a block of data to flash. A block of data is 512 bytes. If a block number crosses a sector boundary, the LPC51U68 automatically erases the sector prior to the write operation.

Table 51. Command packet

Field	Offset	Size (bytes)	Value	Description
command	0x0	0x1	0xA6	‘Write block’ command identifier
crcCheck	0x1	0x1		0 – Do CRC check for this packet. 1 - Ignore CRC field for this packet.
blockNum	0x2	0x2		Flash block number in which the appended data to be programmed. For example to program flash block 0x8000, this parameter should be set to 64.
data	0x4	0x200		Data to be programmed in flash.
checksum	0x204	0x4	CRC32	CRC32 of the packet excluding this field. Set this field to 0 if crcCheck is set to 1.

Table 52. Response packet (success)

Field	Offset	Size (bytes)	Value	Description
sop	0x0	0x1	0x55	Start of packet identifier
command	0x1	0x1	0xA6	Processed command identifier
length	0x2	0x2	0x00	On success this field is set to 0.

Table 53. Response packet (error)

Field	Offset	Size (bytes)	Value	Description
sop	0x0	0x1	0x55	Start of packet identifier
command	0x1	0x1	0xA6	Processed command identifier
length	0x2	0x2	0x00	On error this field is set to 4.
errorCode	0x4	0x4	Error code	Error code specified in error.h

4.8.7 SH_CMD_READ_BLOCK (0xA7) command

The read block command is used to read a block of data from flash. A block of data is 512 bytes.

Table 54. Command packet

Field	Offset	Size (bytes)	Value	Description
command	0x0	0x1	0xA7	‘Read block’ command identifier
reserved	0x1	0x1		Should be zero.
blockNum	0x2	0x2		Flash block number to read. For example to read flash block 0x8000, this parameter should be set to 64.

Table 55. Response packet (success)

Field	Offset	Size (bytes)	Value	Description
sop	0x0	0x1	0x55	Start of packet identifier
command	0x1	0x1	0xA7	Processed command identifier
length	0x2	0x2	0x204	Length of the response packet. 0x204 = success
data	0x4	0x200		Flash block content.
checksum	0x204	0x4	CRC32	CRC32 of the packet excluding this field.

Table 56. Response packet (error)

Field	Offset	Size (bytes)	Value	Description
sop	0x0	0x1	0x55	Start of packet identifier
command	0x1	0x1	0xA7	Processed command identifier
length	0x2	0x2	0x4	Length of the response packet. 0x4 = failure
errorCode	0x4	0x4	Error code	Error code specified in error.h

4.8.8 SH_CMD_SECTOR_ERASE (0xA8) command

The sector erase command is used to erase a sector in flash. A sector size is 32KBytes.

Table 57. Command packet

Field	Offset	Size (bytes)	Value	Description
command	0x0	0x1	0xA8	‘Erase sector’ command identifier
reserved	0x1	0x1		Should be zero.
sectorNum	0x2	0x2		Flash sector number to be erased. For example to erase flash sector at 0x8000, this parameter should be set to 1.

Table 58. Response packet (success)

Field	Offset	Size (bytes)	Value	Description
sop	0x0	0x1	0x55	Start of packet identifier
command	0x1	0x1	0xA8	Processed command identifier
length	0x2	0x2	0x00	On Success this field is set to 0.

Table 59. Response packet (error)

Field	Offset	Size (bytes)	Value	Description
sop	0x0	0x1	0x55	Start of packet identifier
command	0x1	0x1	0xA8	Processed command identifier
length	0x2	0x2	0x4	On error this field is set to 4.
errorCode	0x4	0x4	Error code	Error code specified in error.h

4.8.9 SH_CMD_PAGE_ERASE (0xA9) command

The page erase command is used to erase a page in flash. A page size is 256 bytes.

Table 60. Command packet

Field	Offset	Size (bytes)	Value	Description
command	0x0	0x1	0xA9	‘Page erase’ command identifier
reserved	0x1	0x1		Should be zero.
pageNum	0x2	0x2		Flash page number to be erased. For example to erase flash page at 0x8000, this parameter should be set to 128.

Table 61. Response packet (success)

Field	Offset	Size (bytes)	Value	Description
sop	0x0	0x1	0x55	Start of packet identifier
command	0x1	0x1	0xA9	Processed command identifier
length	0x2	0x2	0x00	Length of the response packet. 0x0 = success

Table 62. Response packet (error)

Field	Offset	Size (bytes)	Value	Description
sop	0x0	0x1	0x55	Start of packet identifier
command	0x1	0x1	0xA9	Processed command identifier
length	0x2	0x2	0x4	Length of the response packet. 0x4 = failure
errorCode	0x4	0x4	Error code	Error code specified in error.h parameters

4.8.10 SH_CMD_PAGE_WRITE (0xAA) command

The page write command is used to write a page in flash. A page size is 256 bytes.

Table 63. Command packet

Field	Offset	Size (bytes)	Value	Description
command	0x0	0x1	0xAA	‘Page write’ command identifier
crcCheck	0x1	0x1		0 – Do CRC check for this packet. 1 - Ignore CRC field for this packet.
pageNum	0x2	0x2		Flash page number in which the appended data to be programmed. For example to program flash page at 0x8000, this parameter should be set to 128.
data	0x4	0x100		Data to be programmed in flash.
checkSum	0x104	0x4	CRC32	CRC32 of the packet excluding this field. Set this field to 0 if crcCheck is set to 1.

Table 64. Response packet (success)

Field	Offset	Size (bytes)	Value	Description
sop	0x0	0x1	0x55	Start of packet identifier
command	0x1	0x1	0xAA	Processed command identifier
length	0x2	0x2	0x00	On Success this field is set to 0.

Table 65. Response packet (error)

Field	Offset	Size (bytes)	Value	Description
sop	0x0	0x1	0x55	Start of packet identifier
command	0x1	0x1	0xAA	Processed command identifier
length	0x2	0x2	0x4	On error this field is set to 4.
errorCode	0x4	0x4	Error code	Error code specified in error.h parameters

4.8.11 SH_CMD_PAGE_READ (0xAB) command

The page read command is used to read a page in flash. A page size is 256 bytes.

Table 66. Command packet

Field	Offset	Size (bytes)	Value	Description
command	0x0	0x1	0xAB	‘Page read’ command identifier
crcCheck	0x1	0x3		0 – Do CRC check for this packet. 1 - Ignore CRC field for this packet.
pageNum	0x4	0x4		Flash page number from which the data to be read. For example to program flash page at 0x8000 this parameter should be set to 128.

Table 67. Response packet (success)

Field	Offset	Size (bytes)	Value	Description
sop	0x0	0x1	0x55	Start of packet identifier
command	0x1	0x1	0xAB	Processed command identifier
length	0x2	0x2	0x00	On Success this field is set to 0x104
data	0x4	0x100	0x104	Flash page content.
checkSum	0x104	0x4	CRC32	CRC32 of the packet excluding this field.

Table 68. Response packet (error)

Field	Offset	Size (bytes)	Value	Description
sop	0x0	0x1	0x55	Start of packet identifier
command	0x1	0x1	0xAB	Processed command identifier
length	0x2	0x2	0x4	On error this field is set to 4.
errorCode	0x4	0x4	Error code	Error code specified in error.h parameters

4.8.12 SH_CMD_WRITE_SUBBLOCK (0xAD) command

The write sub-block command is used for queuing data for a full flash block write. It is used when the host system cannot send the entire data block to the LPC51U68 in a single I2C transfer using the ‘Write block’ command. When using this command, multiple sub-blocks are sent to the LPC51U68 in sequential order for the block. The LPC51U68 will collect all the packets and perform the flash write once the last packet is received. If any other commands are sent between ‘Write sub-block’ commands, the collected buffers are discarded and the operations needs to restart. If a sub-block number crosses a sector boundary, the LPC51U68 will automatically erase the sector prior to the write operation.

Table 69. Command packet

Field	Offset	Size (bytes)	Value	Description
command	0x0	0x1	0xAD	‘Write sub-block’ command identifier
subBlock	0x1	0x1		Bit 0: If set, CRC check is not done for this packet. Bits [5:1]: Specifies the sub-block number. Bits [7:6]: Specifies the sub-block size. 00 – 32 bytes 01 – 64 bytes 10 – 128 bytes 11 – 256 bytes
blockNum	0x2	0x2		Flash block number in which the appended data to be programed. For example to program flash page at 0x8000 this parameter should be set to 64.
data	0x4	Sub-block size		Data to be programmed in flash.
checkSum	Sub-block size + 4	0x4		CRC32 of the packet excluding this field. Set this field to 0 if crcCheck is set to 1.

Table 70. Response packet (success)

Field	Offset	Size (bytes)	Value	Description
sop	0x0	0x1	0x55	Start of packet identifier
command	0x1	0x1	0xAD	Processed command identifier
length	0x2	0x2	0x00	On Success this field is set to 0.

Table 71. Response packet (error)

Field	Offset	Size (bytes)	Value	Description
sop	0x0	0x1	0x55	Start of packet identifier
command	0x1	0x1	0xAD	Processed command identifier
length	0x2	0x2	0x4	On error this field is set to 4
errorCode	0x4	0x4	Error code	Error code specified in error.h parameters

4.8.13 SH_CMD_ READ_SUBBLOCK (0xAE) command

The read sub-block command is used for reading partial data from a full flash block write. It is used when the host system cannot receive the entire data block to the LPC51U68 in a single I2C transfer using the ‘Read block’ command.

Table 72. Command packet

Field	Offset	Size (bytes)	Value	Description
command	0x0	0x1	0xAE	‘Read subblock’ command identifier
subBlock	0x1	0x1		Bit 0: If set, CRC check is not done for this packet. Bits [5:1]: Specifies the sub-block number. Bits [7:6]: Specifies the sub-block size. 00 – 32 bytes 01 – 64 bytes 10 – 128 bytes 11 – 256 bytes
blockNum	0x2	0x2		Flash block number to read. For example to read flash page at 0x8000 this parameter should be set to 64.

Table 73. Response packet (success)

Field	Offset	Size (bytes)	Value	Description
sop	0x0	0x1	0x55	Start of packet identifier
command	0x1	0x1	0xAD	Processed command identifier
length	0x2	0x2	Sub-block size + 4	On Success this field is set to (Sub-block size + 4).
data	0x4	0x2	Sub-block size	Flash data
checksum	Sub-block size + 4	0x4		CRC32 of the packet excluding this field.

Table 74. Response packet (error)

Field	Offset	Size (bytes)	Value	Description
sop	0x0	0x1	0x55	Start of packet identifier
command	0x1	0x1	0xAD	Processed command identifier
length	0x2	0x2	0x4	On error this field is set to 4
errorCode	0x4	0x4	Error code	Error code specified in error.h parameters

4.8.14 SH_CMD_BULK_ERASE (0xAC) command

The bulk erase command is used to erase the entire flash.

Table 75. Command packet

Field	Offset	Size (bytes)	Value	Description
command	0x0	0x1	0xAC	‘Bulk erase’ command identifier
reserved	0x1	0x1		Reserved padding
startSec	0x2	0x1		Starting sector number to erase. Must be >0 for CRP1. Must be 0 for CRP2.
endsec	0x3	0x1		End sector number to erase. Must be last sector for CRP1 and CRP2.

Table 76. Response packet (success)

Field	Offset	Size (bytes)	Value	Description
sop	0x0	0x1	0x55	Start of packet identifier
command	0x1	0x1	0xAC	Processed command identifier
length	0x2	0x2	0x00	On Success this field is set to 0.

Table 77. Response packet (error)

Field	Offset	Size (bytes)	Value	Description
sop	0x0	0x1	0x55	Start of packet identifier
command	0x1	0x1	0xAC	Processed command identifier
length	0x2	0x2	0x4	On error this field is set to 4
errorCode	0x4	0x4	Error code	Error code specified in error.h parameters

4.8.15 I2C/SPI ISP Error codes

These error codes are located in the error.h file.

Table 78. USART ISP Error codes

Return Code	Error code	Description
0x0	ERR_ISP_CMD_SUCCESS	Command is executed successfully. Sent by ISP handler only when command given by the host has been completely and successfully executed.
0x1	ERR_ISP_INVALID_COMMAND	Invalid command.
0x2	ERR_ISP_SRC_ADDR_ERROR	Source address is not on word boundary.
0x3	ERR_ISP_DST_ADDR_ERROR	Destination address is not on a correct boundary.
0x4	ERR_ISP_SRC_ADDR_NOT_MAPPED	Source address is not mapped in the memory map. Count value is taken into consideration where applicable.
0x5	ERR_ISP_DST_ADDR_NOT_MAPPED	Destination address is not mapped in the memory map. Count value is taken into consideration where applicable.
0x6	ERR_ISP_COUNT_ERROR	Byte count is not multiple of 4 or is not a permitted value.
0x7	ERR_ISP_INVALID_SECTOR	Sector number is invalid or end sector number is greater than start sector number.
0x8	ERR_ISP_SECTOR_NOT_BLANK	Sector is not blank.
0x9	ERR_ISP_SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	Command to prepare sector for write operation was not executed.
0xA	ERR_ISP_COMPARE_ERROR	Source and destination data not equal.
0xB	ERR_ISP_BUSY	Flash programming hardware interface is busy.
0xC	ERR_ISP_PARAM_ERROR	Insufficient number of parameters or invalid parameter.
0xD	ERR_ISP_ADDR_ERROR	Address is not on word boundary.
0xE	ERR_ISP_ADDR_NOT_MAPPED	Address is not mapped in the memory map. Count value is taken into consideration where applicable.
0xF	-	Reserved.
0x10	-	Reserved.
0x11	-	Reserved.
0x12	-	-

Table 78. USART ISP Error codes

Return Code	Error code	Description
0x13	ERR_ISP_CODE_READ_ PROTECTION_ENABLED	Code read protection enabled.
0x14	-	Reserved.
0x15	-	Reserved.
0x16	-	Reserved.
0x17	ERR_ISP_FRO_NO_POWER	FRO not turned on in the PDRUNCFG register.
0x18	ERR_ISP_FLASH_NO_POWER	Flash not turned on in the PDRUNCFG register.
0x19	-	Reserved.
0x1A	-	Reserved.
0x1B	ERR_ISP_FLASH_NO_CLOCK	Flash clock disabled in the AHBCLKCTRL register.

4.8.16 I2C/SPI ISP mode protocol software support

This section includes predefined command definitions and software structures used for communication for the I2C/SPI ISP protocol. These may be used with a host application that talks to the I2C/SPI ISP interface via the supported I2C or SPI slave interfaces.

```
/* I2C/SPI IMSP mode protocol commands */
#define SH_CMD_GET_VERSION 0xA1
#define SH_CMD_RESET  0xA2
#define SH_CMD_BOOT   0xA3
#define SH_CMD_CHECK_IMAGE 0xA4
#define SH_CMD_PROBE  0xA5
#define SH_CMD_WRITE_BLOCK 0xA6
#define SH_CMD_READ_BLOCK 0xA7
#define SH_CMD_SECTOR_ERASE 0xA8
#define SH_CMD_PAGE_ERASE 0xA9
#define SH_CMD_PAGE_WRITE 0xAA
#define SH_CMD_PAGE_READ 0xAB
#define SH_CMD_WRITE_SUBBLOCK 0xAC
#define SH_CMD_READ_SUBBLOCK 0xAD
#define SH_CMD_BULK_ERASE 0xAE

/** Structure describing response packet format. */
typedef struct {
     uint8_t sop;    /*!< Start of packet = 0x55 for bootloader */
     uint8_t cmd;    /*!< Response to the Command ID. */
     uint16_t length; /*!< Response data length not including this header. */
} CmdResponse_t;

/** Structure describing Read/Write block command packet format. */
typedef struct {
    uint8_t cmd;      /*!< Command ID */
    uint8_t crc_check; /*!< specifies if we need to do CRC check before processing */
    uint16_t block_nr; /*!< Block number.*/
    uint32_t data[SL_FLASH_BLOCK_SZ/4];    /*!< Data */
    uint32_t crc32;    /*!< CRC32 of command header and data */
} CmdRWBlockParam_t;
```

```
/** Structure describing Read/Write page command packet format. */
typedef struct {
    uint8_t cmd;      /*!< Command ID */
    uint8_t crc_check;/*!< specifies if we need to do CRC check before processing */
    uint16_t page_nr; /*!< page number.*/
    uint32_t data[SL_FLASH_PAGE_SZ/4];     /*!< Data */
    uint32_t crc32;   /*!< CRC32 of command header and data */
} CmdRWPageParam_t;

/** Structure describing Read sub-block command packet format. */
typedef struct {
    uint8_t cmd;          /*!< Command ID */
    uint8_t sub_block_nr; /*!< specifies the sub-block number. Bits below:
                          0 - Skip crc;
                          5-1: sub block nr;
                          7-6: sub-block size. 0 - 32, 1 - 64, 2 - 128, 3 - 256 */
    uint16_t block_nr;    /*!< block number.*/
    uint32_t data[SL_FLASH_BLOCK_SZ/4];    /*!< Data */
} CmdReadSubBlockParam_t;

/** Structure describing Sector erase command packet format. */
typedef struct {
    uint8_t cmd;     /*!< Command ID */
    uint8_t reserved;/*!< Should be zero. */
    uint16_t sec_nr; /*!< Sector number.*/
} CmdEraseSectorParam_t;

/** Structure describing Bulk erase command packet format. */
typedef struct {
    uint8_t cmd;          /*!< Command ID */
    uint8_t reserved;     /*!< Should be zero. */
    uint8_t start_sec;    /*!< Start Sector number.*/
    uint8_t end_sec;      /*!< End Sector number.*/
} CmdBulkEraseParam_t;

/** Structure describing response packet with data. */
typedef struct {
    CmdResponse_t hdr;                 /*!< Response header. */
    uint32_t data[SL_FLASH_BLOCK_SZ/4]; /*!< Data */
    uint32_t crc32;                    /*!< CRC32 of response packet. */
} CmdDataResp_t;
typedef enum
{

ERR_ISP_BASE = 0x00000000,
  /*0x00000001*/ ERR_ISP_INVALID_COMMAND = ERR_ISP_BASE + 1,
  /*0x00000002*/ ERR_ISP_SRC_ADDR_ERROR,
  /*0x00000003*/ ERR_ISP_DST_ADDR_ERROR,
  /*0x00000004*/ ERR_ISP_SRC_ADDR_NOT_MAPPED,
  /*0x00000005*/ ERR_ISP_DST_ADDR_NOT_MAPPED,
  /*0x00000006*/ ERR_ISP_COUNT_ERROR,
  /*0x00000007*/ ERR_ISP_INVALID_SECTOR,
  /*0x00000008*/ ERR_ISP_SECTOR_NOT_BLANK,
  /*0x00000009*/ ERR_ISP_SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION,
  /*0x0000000A*/ ERR_ISP_COMPARE_ERROR,
  /*0x0000000B*/ ERR_ISP_BUSY,  /* Flash programming hardware interface is busy */
  /*0x0000000C*/ ERR_ISP_PARAM_ERROR,    /* Insufficient number of parameters */
  /*0x0000000D*/ ERR_ISP_ADDR_ERROR,     /* Address not on word boundary */
  /*0x0000000E*/ ERR_ISP_ADDR_NOT_MAPPED,
  /*0x0000000F*/ ERR_ISP_CMD_LOCKED,     /* Command is locked */
  /*0x00000010*/ ERR_ISP_INVALID_CODE,   /* Unlock code is invalid */
  /*0x00000011*/ ERR_ISP_INVALID_BAUD_RATE,
  /*0x00000012*/ ERR_ISP_INVALID_STOP_BIT,
  /*0x00000013*/ ERR_ISP_CODE_READ_PROTECTION_ENABLED,
  /*0x00000014*/ ERR_ISP_INVALID_FLASH_UNIT,       /* reserved */
  /*0x00000015*/ ERR_ISP_USER_CODE_CHECKSUM,       /* reserved */
  /*0x00000016*/ ERR_ISP_SETTING_ACTIVE_PARTITION, /* reserved */
  /*0x00000017*/ ERR_ISP_FRO_NO_POWER,
  /*0x00000018*/ ERR_ISP_FLASH_NO_POWER,
  /*0x00000019*/ ERR_ISP_EEPROM_NO_POWER,          /* reserved */
  /*0x0000001A*/ ERR_ISP_EEPROM_NO_CLOCK,          /* reserved */
  /*0x0000001B*/ ERR_ISP_FLASH_NO_CLOCK,
  /*0x0000001C*/ ERR_ISP_REINVOKE_ISP_CONFIG
} ErrorCode_t;
```

UM11071

Chapter 5: LPC51U68 Nested Vectored Interrupt Controller (NVIC)

Rev. 1.1 — 17 May 2018 **User manual**

5.1 How to read this chapter

Available interrupt sources may vary with specific LPC51U68 device type.

5.2 Features

- Nested Vectored Interrupt Controller that is an integral part of each CPU.
- Tightly coupled interrupt controller provides low interrupt latency.
- Controls system exceptions and peripheral interrupts.
- The NVIC of the Cortex- M0+ supports the first 32 interrupts.
  – 32 vectored interrupt slots.
  – 4 programmable interrupt priority levels with hardware priority level masking.
  – Vector table offset register VTOR.
- Support for NMI from any interrupt (see Section 6.5.3).

5.3 General description

The tight coupling to the NVIC to the CPU allows for low interrupt latency and efficient processing of late arriving interrupts.

5.3.1 Interrupt sources

Table 79 lists the interrupt sources for each peripheral function. Each peripheral device may have one or more interrupt lines to the Vectored Interrupt Controller. Each line may represent more than one interrupt source. The interrupt number does not imply any interrupt priority.

See Ref. 1 “Cortex-M0+ TRM” for detailed descriptions of the NVIC and the NVIC registers.

Table 79. Connection of interrupt sources to the NVIC

Interrupt	Name	Interrupt description	Flags
0	WDT, BOD	Windowed watchdog timer, Brownout detect	WARNINT - watchdog warning interrupt BODINTVAL - BOD interrupt level
1	DMA	DMA controller	Interrupt A and interrupt B, error interrupt
2	GINT0	GPIO group 0	Enabled pin interrupts
3	GINT1	GPIO group 1	Enabled pin interrupts
4	PIN_INT0	Pin interrupt 0 or pattern match engine slice 0	PSTAT - pin interrupt status
5	PIN_INT1	Pin interrupt 1or pattern match engine slice 1	PSTAT - pin interrupt status
6	PIN_INT2	Pin interrupt 2 or pattern match engine slice 2	PSTAT - pin interrupt status
7	PIN_INT3	Pin interrupt 3 or pattern match engine slice 3	PSTAT - pin interrupt status
8	UTICK	Micro-tick Timer	INTR
9	MRT	Multi-rate timer	Global MRT interrupts: GFLAG0, 1, 2, 3
10	CTIMER0	Standard counter/timer CTIMER0	Match and Capture interrupts
11	CTIMER1	Standard counter/timer CTIMER1	Match and Capture interrupts
12	SCT0	SCTimer/PWM	EVFLAG SCT event
13	CTIMER3	Standard counter/timer CTIMER3	Match and Capture interrupts
14	Flexcomm0	Flexcomm Interface 0 (USART, SPI, I2C)	See Table 372, Table 394, Table 420.
15	Flexcomm1	Flexcomm Interface 1 (USART, SPI, I2C)	Same as Flexcomm0
16	Flexcomm2	Flexcomm Interface 2 (USART, SPI, I2C)	Same as Flexcomm0
17	Flexcomm3	Flexcomm Interface 3 (USART, SPI, I2C)	Same as Flexcomm0
18	Flexcomm4	Flexcomm Interface 4 (USART, SPI, I2C)	Same as Flexcomm0
19	Flexcomm5	Flexcomm Interface 5 (USART, SPI, I2C)	Same as Flexcomm0
20	Flexcomm6	Flexcomm Interface 6 (USART, SPI, I2C, I2S)	Same as Flexcomm0, plus I^{2}S
21	Flexcomm7	Flexcomm Interface 7 (USART, SPI, I2C, I2S)	Same as Flexcomm0, plus I^{2}S (Table 449)
22	ADC0_SEQA	ADC0 sequence A completion.	See Table 473.
23	ADC0_SEQB	ADC0 sequence B completion.	See Table 473.
24	ADC0_THCMP	ADC0 threshold compare and error.	See Table 473.
25	(reserved)	-	-
26	(reserved)	-	-
27	USB_WAKEUP	USB Activity Interrupt	USB_NEED_CLK, see Chapter 22.
28	USB	USB device	See Table 356.
29	RTC	RTC alarm and wake-up interrupts	See Table 321.
30	(reserved)	-	-
31	(reserved)	-	-

5.4 Register description

The NVIC registers are located on the ARM private peripheral bus.

Table 80. Register overview: NVIC (base address 0xE000 E000)

Name	Access	Offset	Description	Reset value	Section
ISER0	R/W	0x100	Interrupt Set Enable Register 0. This register allows enabling interrupts and reading back the interrupt enables for peripheral functions.	0x0	5.4.2
-	-	0x104	Reserved	0x0	-
ICER0	R/W	0x180	Interrupt Clear Enable Register 0. This register allows disabling interrupts and reading back the interrupt enables for peripheral functions.	0x0	5.4.3
-	-	0x184	Reserved	0x0	-
ISPR0	R/W	0x200	Interrupt Set Pending Register 0. This register allows changing the interrupt state to pending and reading back the interrupt pending state for peripheral functions.	0x0	5.4.4
-	-	0x204	Reserved	0x0	-
ICPR0	R/W	0x280	Interrupt Clear Pending Register 0. This register allows changing the interrupt state to not pending and reading back the interrupt pending state for peripheral functions.	0x0	5.4.5
-	-	0x284	Reserved	0x0	-
IPR0	R/W	0x400	Interrupt Priority Register 0. This register contains the priority fields for interrupts 0 to 3. Two bits for the Cortex-M0+.	0x0	5.4.6
IPR1	R/W	0x404	Interrupt Priority Register 1. This register contains the 3-bit priority fields for interrupts 4 to 7. Two bits for the Cortex-M0+.	0x0	5.4.7
IPR2	R/W	0x408	Interrupt Priority Register 2. This register contains the 3-bit priority fields for interrupts 8 to 11. Two bits for the Cortex-M0+.	0x0	5.4.8
IPR3	R/W	0x40C	Interrupt Priority Register 3. This register contains the 3-bit priority fields for interrupts 12 to 15. Two bits for the Cortex-M0+.	0x0	5.4.9
IPR4	R/W	0x410	Interrupt Priority Register 4. This register contains the 3-bit priority fields for interrupts 16 to 19. Two bits for the Cortex-M0+.	0x0	5.4.10
IPR5	R/W	0x414	Interrupt Priority Register 5. This register contains the 3-bit priority fields for interrupts 20 to 23. Two bits for the Cortex-M0+.	0x0	5.4.11
IPR6	R/W	0x418	Interrupt Priority Register 6. This register contains the 3-bit priority fields for interrupts 24 to 27. Two bits for the Cortex-M0+.	0x0	5.4.12
IPR7	R/W	0x41C	Interrupt Priority Register 7. This register contains the 3-bit priority fields for interrupts 28 to 31. Two bits for the Cortex-M0+.	0x0	5.4.13

5.4.1 Interrupt register bits and fields summary

Table 81 shows the bits or fields in the NVIC that are relevant to each interrupt.

Table 81. Registers related to each Interrupt source

Int #	Name	ISER bit	ICER	ISPR	ICPR	IABR	IPR bits
0	WDT, BOD	ISER0[0]	ICER0[0]	ISPR0[0]	ICPR0[0]	IABR0[0]	IPR0[7:5]
1	DMA	ISER0[1]	ICER0[1]	ISPR0[1]	ICPR0[1]	IABR0[1]	IPR0[15:13]
2	GINT0	ISER0[2]	ICER0[2]	ISPR0[2]	ICPR0[2]	IABR0[2]	IPR0[23:21]
3	GINT1	ISER0[3]	ICER0[3]	ISPR[3]	ICPR[3]	IABR[3]	IPR0[31:29]
4	PIN_INT0	ISER0[4]	ICER0[4]	ISPR[4]	ICPR[4]	IABR[4]	IPR1[7:5]
5	PIN_INT1	ISER0[5]	ICER0[5]	ISPR[5]	ICPR[5]	IABR[5]	IPR1[15:13]
6	PIN_INT2	ISER0[6]	ICER0[6]	ISPR[6]	ICPR[6]	IABR[6]	IPR1[23:21]
7	PIN_INT3	ISER0[7]	ICER0[7]	ISPR[7]	ICPR[7]	IABR[7]	IPR1[31:29]
8	UTICK	ISER0[8]	ICER0[8]	ISPR[8]	ICPR[8]	IABR[8]	IPR2[7:5]
9	MRT	ISER0[9]	ICER0[9]	ISPR[9]	ICPR[9]	IABR[9]	IPR2[15:13]
10	CTIMER0	ISER0[10]	ICER0[10]	ISPR[10]	ICPR[10]	IABR[10]	IPR2[23:21]
11	CTIMER1	ISER0[11]	ICER0[11]	ISPR[11]	ICPR[11]	IABR[11]	IPR2[31:29]
12	SCT0	ISER0[12]	ICER0[12]	ISPR[12]	ICPR[12]	IABR[12]	IPR3[7:5]
13	CTIMER3	ISER0[13]	ICER0[13]	ISPR[13]	ICPR[13]	IABR[13]	IPR3[15:13]
14	Flexcomm0	ISER0[14]	ICER0[14]	ISPR[14]	ICPR[14]	IABR[14]	IPR3[23:21]
15	Flexcomm1	ISER0[15]	ICER0[15]	ISPR[15]	ICPR[15]	IABR[15]	IPR3[31:29]
16	Flexcomm2	ISER0[16]	ICER0[16]	ISPR[16]	ICPR[16]	IABR[16]	IPR4[7:5]
17	Flexcomm3	ISER0[17]	ICER0[17]	ISPR[17]	ICPR[17]	IABR[17]	IPR4[15:13]
18	Flexcomm4	ISER0[18]	ICER0[18]	ISPR[18]	ICPR[18]	IABR[18]	IPR4[23:21]
19	Flexcomm5	ISER0[19]	ICER0[19]	ISPR[19]	ICPR[19]	IABR[19]	IPR4[31:29]
20	Flexcomm6	ISER0[20]	ICER0[20]	ISPR[20]	ICPR[20]	IABR[20]	IPR5[7:5]
21	Flexcomm7	ISER0[21]	ICER0[21]	ISPR[21]	ICPR[21]	IABR[21]	IPR5[15:13]
22	ADC0_SEQA	ISER0[22]	ICER0[22]	ISPR[22]	ICPR[22]	IABR[22]	IPR5[23:21]
23	ADC0_SEQB	ISER0[23]	ICER0[23]	ISPR[23]	ICPR[23]	IABR[23]	IPR5[31:29]
24	ADC0_THCMP	ISER0[24]	ICER0[24]	ISPR[24]	ICPR[24]	IABR[24]	IPR6[7:5]
25	RESERVED	-	-	-	-	-	-
26	RESERVED	-	-	-	-	-	-
27	USB0_WAKEUP	ISER0[27]	ICER0[27]	ISPR[27]	ICPR[27]	IABR[27]	IPR6[31:29]
28	USB0	ISER0[28]	ICER0[28]	ISPR[28]	ICPR[28]	IABR[28]	IPR7[7:5]
29	RTC	ISER0[29]	ICER0[29]	ISPR[29]	ICPR[29]	IABR[29]	IPR7[15:13]
31	RESERVED	-	-	-	-	-	-

5.4.2 Interrupt Set-Enable Register 0 register

The ISER0 register allows enabling the first 32 peripheral interrupts, or for reading the enabled state of those interrupts. Disabling interrupts is done through the ICER0 and ICER1 registers (Section 5.4.3).

Table 82. Interrupt Set-Enable Register 0 register

Bit	Name	Value	Function
0	ISE_WDTBOD	[1]	Watchdog Timer, BOD interrupt enable.
1	ISE_DMA	[1]	DMA interrupt enable.
2	ISE_GINT0	[1]	GPIO group 0 interrupt enable.
3	ISE_GINT1	[1]	GPIO group 1 interrupt enable.
4	ISE_PINT0	[1]	Pin interrupt / pattern match engine slice 0 interrupt enable.
5	ISE_PINT1	[1]	Pin interrupt / pattern match engine slice 1 interrupt enable.
6	ISE_PINT2	[1]	Pin interrupt / pattern match engine slice 2 interrupt enable.
7	ISE_PINT3	[1]	Pin interrupt / pattern match engine slice 3 interrupt enable.
8	ISE_UTICK	[1]	Micro-Tick Timer interrupt enable.
9	ISE_MRT	[1]	Multi-Rate Timer interrupt enable.
10	ISE_CTIMER0	[1]	Standard counter/timer CTIMER0 interrupt enable.
11	ISE_CTIMER1	[1]	Standard counter/timer CTIMER1 interrupt enable.
12	ISE_SCT0	[1]	SCT0 interrupt enable.
13	ISE_CTIMER3	[1]	Standard counter/timer CTIMER3 interrupt enable.
14	ISE_FC0	[1]	Flexcomm Interface 0 interrupt enable.
15	ISE_FC1	[1]	Flexcomm Interface 1 interrupt enable.
16	ISE_FC2	[1]	Flexcomm Interface 2 interrupt enable.
17	ISE_FC3	[1]	Flexcomm Interface 3 interrupt enable.
18	ISE_FC4	[1]	Flexcomm Interface 4 interrupt enable.
19	ISE_FC5	[1]	Flexcomm Interface 5 interrupt enable.
20	ISE_FC6	[1]	Flexcomm Interface 6 interrupt enable.
21	ISE_FC7	[1]	Flexcomm Interface 7 interrupt enable.
22	ISE_ADC0SEQA	[1]	ADC0 sequence A interrupt enable.
23	ISE_ADC0SEQB	[1]	ADC0 sequence B interrupt enable.
24	ISE_ADC0THOV	[1]	ADC0 threshold and error interrupt enable.
25	-	-	Reserved
26	-	-	Reserved
27	ISE_USBACT	[1]	USB activity interrupt enable.
28	ISE_USB	[1]	USB device interrupt enable.
29	ISE_RTC	[1]	Real Time Clock (RTC) interrupt enable.
30	-	-	Reserved. Read value is undefined, only zero should be written.
31	-	-	Reserved

[1] Write: writing 0 has no effect, writing 1 enables the interrupt.
Read: 0 indicates that the interrupt is disabled, 1 indicates that the interrupt is enabled.

5.4.3 Interrupt Clear-Enable Register 0

The ICER0 register allows disabling the first 32 peripheral interrupts, or for reading the enabled state of those interrupts. Enabling interrupts is done through the ISER0 and ISER1 registers (Section 5.4.2).

Table 83. Interrupt Clear-Enable Register 0

Bit	Name	Function
31:0	ICE_...	Peripheral interrupt disables. Bit numbers match ISER0 registers (Table 82). Unused bits are reserved. Write: writing 0 has no effect, writing 1 disables the interrupt. Read: 0 indicates that the interrupt is disabled, 1 indicates that the interrupt is enabled.

5.4.4 Interrupt Set-Pending Register 0 register

The ISPR0 register allows setting the pending state of the first 32 peripheral interrupts, or for reading the pending state of those interrupts. Clearing the pending state of interrupts is done through the ICPR0 and ICPR1 registers (Section 5.4.5).

Table 84. Interrupt Set-Pending Register 0 register

Bit	Name	Function
31:0	ISP_...	Peripheral interrupt pending set. Bit numbers match ISER0 registers (Table 82). Unused bits are reserved. Write: writing 0 has no effect, writing 1 changes the interrupt state to pending. Read: 0 indicates that the interrupt is not pending, 1 indicates that the interrupt is pending.

5.4.5 Interrupt Clear-Pending Register 0 register

The ICPR0 register allows clearing the pending state of the first 32 peripheral interrupts, or for reading the pending state of those interrupts. Setting the pending state of interrupts is done through the ISPR0 and ISPR1 registers (Section 5.4.4).

Table 85. Interrupt Clear-Pending Register 0 register

Bit	Name	Function
31:0	ICP_...	Peripheral interrupt pending clear. Bit numbers match ISER0 registers (Table 82). Unused bits are reserved. Write: writing 0 has no effect, writing 1 changes the interrupt state to not pending. Read: 0 indicates that the interrupt is not pending, 1 indicates that the interrupt is pending.

5.4.6 Interrupt Priority Register 0

The IPR0 register controls the priority of the first 4 peripheral interrupts. Each interrupt can have one of four priorities for the Cortex-M0+. 0 is the highest priority.

Table 86. Interrupt Priority Register 0

Bit	Name	Function
4:0	-	Unused
7:5	IP_WDTBOD	Watchdog Timer and BOD interrupt priority. 0 = highest priority, 7 = lowest priority.
12:8	-	Unused
15:13	IP_DMA	DMA interrupt priority. 0 = highest priority, 7 = lowest priority.
20:16	-	Unused

Table 86. Interrupt Priority Register 0 *…continued*

Bit	Name	Function
23:21	IP_GINT0	GPIO Group 0 interrupt priority. 0 = highest priority, 7 = lowest priority.
28:24	-	Unused
31:29	IP_GINT1	GPIO Group 1 interrupt priority. 0 = highest priority, 7 = lowest priority.

5.4.7 Interrupt Priority Register 1

The IPR1 register controls the priority of the second group of 4 peripheral interrupts. Each interrupt can have one of four priorities for the Cortex-M0+. 0 is the highest priority.

Table 87. Interrupt Priority Register 1

Bit	Name	Function
4:0	-	Unused
7:5	IP_PINT0	Pin interrupt / pattern match engine slice 0 priority. 0 = highest priority, 7 = lowest priority.
12:8	-	Unused
15:13	IP_PINT1	Pin interrupt / pattern match engine slice 1 priority. 0 = highest priority, 7 = lowest priority.
20:16	-	Unused
23:21	IP_PINT2	Pin interrupt / pattern match engine slice 2 priority. 0 = highest priority, 7 = lowest priority.
28:24	-	Unused
31:29	IP_PINT3	Pin interrupt / pattern match engine slice 3 priority. 0 = highest priority, 7 = lowest priority.

5.4.8 Interrupt Priority Register 2

The IPR2 register controls the priority of the third group of 4 peripheral interrupts. Each interrupt can have one of four priorities for the Cortex-M0+. 0 is the highest priority.

Table 88. Interrupt Priority Register 2

Bit	Name	Function
4:0	-	Unused
7:5	IP_UTICK	Micro-Tick Timer interrupt priority. 0 = highest priority, 7 = lowest priority.
12:8	-	Unused
15:13	IP_MRT	Multi-Rate Timer interrupt priority. 0 = highest priority, 7 = lowest priority.
20:16	-	Unused
23:21	IP_CTIMER0	Standard counter/timer CTIMER0 interrupt priority. 0 = highest priority, 7 = lowest priority.
28:24	-	Unused
31:29	IP_CTIMER1	Standard counter/timer CTIMER1 interrupt priority. 0 = highest priority, 7 = lowest priority.

5.4.9 Interrupt Priority Register 3

The IPR3 register controls the priority of the fourth group of 4 peripheral interrupts. Each interrupt can have one of four priorities for the Cortex-M0+. 0 is the highest priority.

Table 89. Interrupt Priority Register 3

Bit	Name	Function
4:0	-	Unused
7:5	IP_SCT0	SCT0 interrupt priority. 0 = highest priority, 7 = lowest priority.
12:8	-	Unused
15:13	IP_CTIMER3	Standard counter/timer CTIMER3 interrupt priority. 0 = highest priority, 7 = lowest priority.
20:16	-	Unused
23:21	IP_FC0	Flexcomm Interface 0 interrupt priority. 0 = highest priority, 7 = lowest priority.
28:24	-	Unused
31:29	IP_FC1	Flexcomm Interface 1 interrupt priority. 0 = highest priority, 7 = lowest priority.

5.4.10 Interrupt Priority Register 4

The IPR4 register controls the priority of the fifth group of 4 peripheral interrupts. Each interrupt can have one of four priorities for the Cortex-M0+. 0 is the highest priority.

Table 90. Interrupt Priority Register 4

Bit	Name	Function
4:0	-	Unused
7:5	IP_FC2	Flexcomm Interface 2 interrupt priority. 0 = highest priority, 7 = lowest priority.
12:8	-	Unused
15:13	IP_FC3	Flexcomm Interface 3 interrupt priority. 0 = highest priority, 7 = lowest priority.
20:16	-	Unused
23:21	IP_FC4	Flexcomm Interface 4 interrupt priority. 0 = highest priority, 7 = lowest priority.
28:24	-	Unused
31:29	IP_FC5	Flexcomm Interface 5 interrupt priority. 0 = highest priority, 7 = lowest priority.

5.4.11 Interrupt Priority Register 5

The IPR5 register controls the priority of the sixth group of 4 peripheral interrupts. Each interrupt can have one of four priorities for the Cortex-M0+. 0 is the highest priority.

Table 91. Interrupt Priority Register 5

Bit	Name	Function
4:0	-	Unused
7:5	IP_FC6	Flexcomm Interface 6 interrupt priority. 0 = highest priority, 7 = lowest priority.
12:8	-	Unused
15:13	IP_FC7	Flexcomm Interface 7 interrupt priority. 0 = highest priority, 7 = lowest priority.
20:16	-	Unused
23:21	IP_ADC0SEQA	ADC 0 sequence A interrupt priority. 0 = highest priority, 7 = lowest priority.
28:24	-	Unused
31:29	IP_ADC0SEQB	ADC 0 sequence B interrupt priority. 0 = highest priority, 7 = lowest priority.

5.4.12 Interrupt Priority Register 6

The IPR6 register controls the priority of the seventh group of 4 peripheral interrupts. Each interrupt can have one of four priorities for the Cortex-M0+. 0 is the highest priority.

Table 92. Interrupt Priority Register 6

Bit	Name	Function
4:0	-	Unused
7:5	IP_ADC0THOV	ADC 0 threshold and error interrupt priority. 0 = highest priority, 7 = lowest priority.
12:8	-	Unused
15:13	-	-
28:16	-	Unused
31:29	IP_USBACT	USB Activity interrupt priority. 0 = highest priority, 7 = lowest priority.

5.4.13 Interrupt Priority Register 7

The IPR7 register controls the priority of the eighth group of 4 peripheral interrupts. Each interrupt can have one of four priorities for the Cortex-M0+. 0 is the highest priority.

Table 93. Interrupt Priority Register 7

Bit	Name	Function
4:0	-	Unused
7:5	IP_USB	USB interrupt enable. 0 = highest priority, 7 = lowest priority.
12:8	-	Unused
15:13	IP_RTC	Real Time clock (RTC) interrupt priority. 0 = highest priority, 7 = lowest priority.
20:16	-	Unused
23:21	-	Reserved
31:24	-	Unused

5.4.14 Software Trigger Interrupt Register

The STIR register provides an alternate way for software to generate an interrupt, in addition to using the ISPR registers. This mechanism can only be used to generate peripheral interrupts, not system exceptions. The STIR register is not available for the Cortex-M0+.

By default, only privileged software can write to the STIR register. Unprivileged software can be given this ability if privileged software sets the USERSETMPEND bit in the CCR register.

The interrupt number to be programmed in this register is listed in Table 79.

Table 94. Software Trigger Interrupt Register (STIR)

Bit	Symbol	Description
8:0	INTID	Writing a value to this field generates an interrupt for the specified the interrupt number.
31:9	-	Reserved. Read value is undefined, only zero should be written.

UM11071

Chapter 6: LPC51U68 System configuration (SYSCON)

Rev. 1.1 — 17 May 2018 **User manual**

6.1 Features

- System and bus configuration.
- Clock select and control.
- PLL configuration
- Reset control.
- Wake-up control.
- BOD configuration.
- High-accuracy frequency measurement function for on-chip and off-chip clocks.
- Uses a selection of on-chip clocks as reference clock.
- Device ID register.

6.2 Basic configuration

Configure the SYSCON block as follows:

- No clock configuration is needed. The clock to the SYSCON block is always enabled. By default, the SYSCON block is clocked by the FRO 12 MHz (fro_12m).
- Target and reference clocks for the frequency measurement function are selected in the input mux block. See Table 200.
- The SYSCON block controls use of the CLKIN and CLKOUT pins which must also be configured through IOCON. See Section 6.3. $\overline{\text{RESET}}$ is a dedicated pin.

6.2.1 Set up the PLL

The PLL creates a stable output clock at a higher frequency than the input clock. If a main clock is needed with a frequency higher than the FRO 12 MHz clock and the FRO 96 or 48 MHz clock (fro_hf) is not appropriate, use the PLL to boost the input frequency. The PLL can be set up by calling an API supplied by NXP Semiconductors. Also see Section 6.6.5 “System PLL functional description”, Section 6.5.46 “PLL registers”, and Chapter 8 “LPC51U68 Power profiles/Power control API”.

6.2.2 Configure the main clock and system clock

The clock source for the registers and memories is derived from main clock. The main clock can be selected from the sources listed in step 1 below.

The main clock, after being optionally divided by the CPU Clock Divider, is called the system clock and clocks the core, the memories, and the peripherals (register interfaces and peripheral clocks).

1. Select the main clock. The following options are available:
   – FRO 12 MHz output (fro_12m) from internal oscillator (default). This clock is divided down from the currently selected FRO oscillator.
   – FRO high speed output (fro_hf), 96 or 48 MHz from internal oscillator.

  - CLKIN.
  - Watchdog oscillator.
  - The output of the system PLL.
  - The RTC 32 kHz oscillator.

  Section 6.5.22 “Main clock source select register A” and Section 6.5.23 “Main clock source select register B”.

2. Select the divider value for the system clock.

   Section 6.5.32 “AHB clock divider register”

3. Select the memories and peripherals that are operating in the application and therefore must have an active clock. The core is always clocked.

   Section 6.5.16 “AHB Clock Control register 0” and Section 6.5.17 “AHB Clock Control register 1”.

6.2.3 Measure the frequency of a clock signal

The frequency of any on-chip or off-chip clock signal can be measured accurately with a selectable reference clock. For example, the frequency measurement function can be used to accurately determine the frequency of the watchdog oscillator which varies over a wide range depending on process and temperature.

The clock frequency to be measured and the reference clock are selected in the input mux block. See Section 10.6.4 “Frequency measure function reference clock select register” and Section 10.6.5 “Frequency measure function target clock select register”.

Details on the accuracy and measurement process are described in Section 6.6.6 “Frequency measure function”.

To start a frequency measurement cycle and read the result, see Table 140.

6.3 Pin description

Table 95. SYSCON pin description

Function	Type	Pin	Description	Reference
CLKOUT	O	PIO0_21	CLKOUT clock output.	Chapter 9
CLKIN	I	PIO0_22	External clock input.	Chapter 9

6.4 General description

6.4.1 Clock generation

The system control block facilitates the clock generation. Many clocking variations are possible. Figure 9 gives an overview of potential clock options. Table 96 describes signals on the clocking diagram. The maximum clock frequency is 100 MHz.

Remark: The indicated clock multiplexers shown in Figure 9 are synchronized. In order to operate, the currently selected clock must be running, and the clock to be switched to must also be running. This is so that the multiplexer can gracefully switch between the two clocks without glitches. Other clock multiplexers are not synchronized. The output divider can be stopped and restarted gracefully during switching if a glitch-free output is needed.

The low-power watchdog oscillator provides a selectable frequency in the range of 6 kHz to 1.5 MHz. The accuracy of this clock is limited to +/- 40% over temperature, voltage, and silicon processing variations. To determine the actual watchdog oscillator output, use the frequency measure block. See Section 6.2.3.

The part contains one system PLL that can be configured to use a number of clock inputs and produce an output clock in the range of 1.2 MHz up to the maximum chip frequency, and can be used to run most on-chip functions. The output of the PLL can be monitored through the CLKOUT pin.

Table 96. Clocking diagram signal name descriptions

Name	Description
32k_clk	The 32 kHz output of the RTC oscillator. The 32 kHz clock must be enabled in the RTCOSCCTRL register (see Section 6.5.45).
clk_in	This is the internal clock that comes from the main CLK_IN pin function. That function must be connected to the pin by selecting it in the IOCON block.
frg_clk	The output of the Fractional Rate Generator. The FRG and its source selection are shown in Figure 9.
fro_12m	12 MHz divided down from the currently selected on-chip FRO oscillator. See Section 6.5.43.
fro_hf	The currently selected FRO high speed output. This may be either 96 MHz or 48 MHz. See Section 6.5.43.
main_clk	The main clock used by the CPU and AHB bus, and potentially many others. The main clock and its source selection are shown in Figure 9.
mclk_in	The MCLK input function, when it is connected to a pin by selecting it in the IOCON block.
pll_clk	The output of the PLL. The PLL and its source selection are shown in Figure 9.
wdt_clk	The output of the watchdog oscillator, which has a selectable target frequency (see Section 6.5.44). It must also be enabled in the PDRINCFG0 register (see Section 6.5.48).
“none”	A tied-off source that should be selected to save power when the output of the related multiplexer is not used.

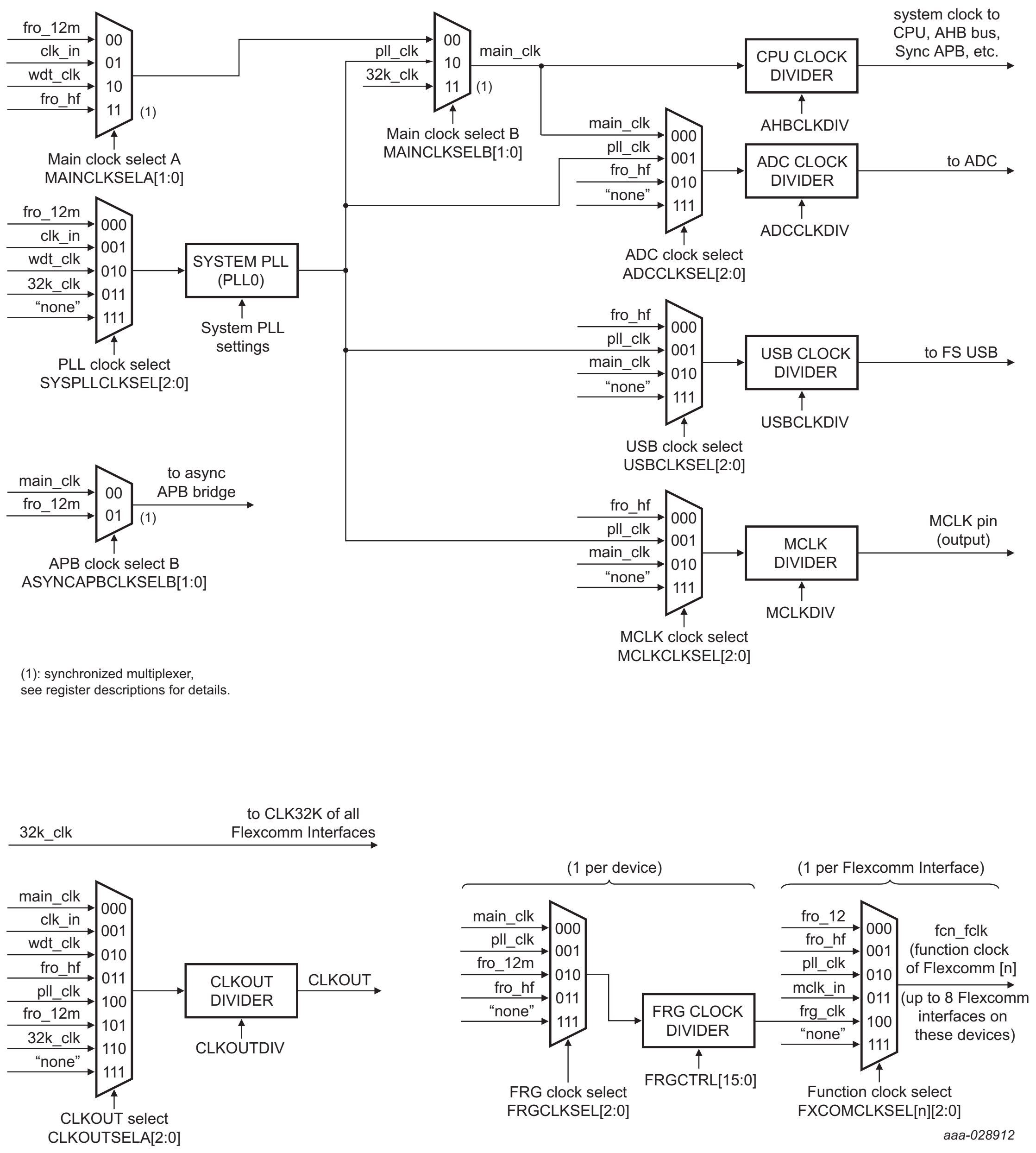


Fig 9. Clock generation

6.5 Register description

All system control block registers reside on word address boundaries. Details of the registers appear in the description of each function. System configuration functions are divided into 3 groups:

- Main system configuration at base address 0x4000 0000 (see Table 97).
- Asynchronous system configuration at base address 0x4004 0000 (see Table 98).
- Other system registers at base address 0x4002 0000 (see Table 99).

All address offsets not shown in the tables are reserved and should not be written to.

Remark: The reset value column shows the reset value seen when the boot loader executes and the flash contains valid user code. During code development, a different value may be seen if a debugger is used to halt execution prior to boot completion.

Table 97. Register overview: Main system configuration (base address 0x4000 0000)

Name	Access	Offset	Description	Reset value [1]	Section
AHBMATPRIO	R/W	0x010	AHB multilayer matrix priority control	0x0	6.5.1
SYSTCKCAL	R/W	0x040	System tick counter calibration	0x0	6.5.2
NMISRC	R/W	0x048	NMI Source Select	0x0	6.5.3
ASYNCAPBCTRL	R/W	0x04C	Asynchronous APB Control	0x1	6.5.4
PIOPORCAP0	RO	0x0C0	POR captured value of port 0	Note [2]	6.5.5
PIOPORCAP1	RO	0x0C4	POR captured value of port 1	Note [2]	6.5.6
PIORESCAP0	RO	0x0D0	Reset captured value of port 0	Note [3]	6.5.7
PIORESCAP1	RO	0x0D4	Reset captured value of port 1	Note [3]	6.5.8
PRESETCTRL0	R/W	0x100	Peripheral reset control 0	0x0	6.5.9
PRESETCTRL1	R/W	0x104	Peripheral reset control 1	0x0	6.5.10
PRESETCTRLSET0	WO	0x120	Set bits in PRESETCTRL0	-	6.5.11
PRESETCTRLSET1	WO	0x124	Set bits in PRESETCTRL1	-	6.5.12
PRESETCTRLCLR0	WO	0x140	Clear bits in PRESETCTRL0	-	6.5.13
PRESETCTRLCLR1	WO	0x144	Clear bits in PRESETCTRL1	-	6.5.14
SYSRSTSTAT	R/W	0x1F0	System reset status register	Note [4]	6.5.15
AHBCLKCTRL0	R/W	0x200	AHB Clock control 0	0x18B	6.5.16
AHBCLKCTRL1	R/W	0x204	AHB Clock control 1	0x0	6.5.17
AHBCLKCTRLSET0	WO	0x220	Set bits in AHBCLKCTRL0	-	6.5.18
AHBCLKCTRLSET1	WO	0x224	Set bits in AHBCLKCTRL1	-	6.5.19
AHBCLKCTRLCLR0	WO	0x240	Clear bits in AHBCLKCTRL0	-	6.5.20
AHBCLKCTRLCLR1	WO	0x244	Clear bits in AHBCLKCTRL1	-	6.5.21
MAINCLKSELA	R/W	0x280	Main clock source select A	0x0	6.5.22
MAINCLKSELB	R/W	0x284	Main clock source select B	0x0	6.5.23
CLKOUTSELA	R/W	0x288	CLKOUT clock source select A	0x7	6.5.24
SYSPLLCLKSEL	R/W	0x290	PLL clock source select	0x7	6.5.25
ADCCLKSEL	R/W	0x2A4	ADC clock source select	0x7	6.5.26
USBCLKSEL	R/W	0x2A8	USB clock source select	0x7	6.5.27
FCLKSEL0	R/W	0x2B0	Flexcomm Interface 0 clock source select	0x7	6.5.28

Table 97. Register overview: Main system configuration (base address 0x4000 0000) *…continued*

Name	Access	Offset	Description	Reset value [1]	Section
FCLKSEL1	R/W	0x2B4	Flexcomm Interface 1 clock source select	0x7	6.5.28
FCLKSEL2	R/W	0x2B8	Flexcomm Interface 2 clock source select	0x7	6.5.28
FCLKSEL3	R/W	0x2BC	Flexcomm Interface 3 clock source select	0x7	6.5.28
FCLKSEL4	R/W	0x2C0	Flexcomm Interface 4 clock source select	0x7	6.5.28
FCLKSEL5	R/W	0x2C4	Flexcomm Interface 5 clock source select	0x7	6.5.28
FCLKSEL6	R/W	0x2C8	Flexcomm Interface 6 clock source select	0x7	6.5.28
FCLKSEL7	R/W	0x2CC	Flexcomm Interface 7 clock source select	0x7	6.5.28
MCLKCLKSEL	R/W	0x2E0	MCLK clock source select	0x7	6.5.29
FRGCLKSEL	R/W	0x2E8	Fractional Rate Generator clock source select	0x7	6.5.30
RESERVED	-	0x2EC	-	-	-
SYSTICKCLKDIV	R/W	0x300	SYSTICK clock divider	0x4000 0000	6.5.31
RESERVED	-	0x304	-	-	-
AHBCLKDIV	R/W	0x380	AHB clock divider	0x0	6.5.32
CLKOUTDIV	R/W	0x384	CLKOUT clock divider	0x4000 0000	6.5.33
ADCCLKDIV	R/W	0x394	ADC clock divider	0x4000 0000	6.5.34
USBCLKDIV	R/W	0x398	USB clock divider	0x4000 0000	6.5.35
FRGCTRL	R/W	0x3A0	Fractional rate divider	0xFF	6.5.36
RESERVED	-	0x3A8	-	-	-
I2SMCLKDIV	R/W	0x3AC	I2S MCLK clock divider	0x4000 0000	6.5.37
FLASHCFG	R/W	0x400	Flash wait states configuration	0x001A	6.5.38
USBCLKCTRL	R/W	0x40C	USB clock control	0x0	6.5.39
USBCLKSTAT	R/W	0x410	USB clock status	0x0	6.5.40
FREQMECTRL	R/W	0x418	Frequency measure register	0x0	6.5.41
MCLKIO	R/W	0x420	MCLK input/output control	0x0	6.5.42
FROCTRL	R/W	0x500	FRO oscillator control	Note [5]	6.5.43
WDTOSCCTRL	R/W	0x508	Watchdog oscillator control	0xA0	6.5.44
RTCOSCCTRL	R/W	0x50C	RTC oscillator 32 kHz output control	0x1	6.5.45
SYSPLLCTRL	R/W	0x580	PLL control	0x0	6.5.46.1
SYSPLLSTAT	RO	0x584	PLL status	0x0	6.5.46.2
SYSPLLNDEC	R/W	0x588	PLL N decoder	0x0	6.5.46.2
SYSPLLPDEC	R/W	0x58C	PLL P decoder	0x0	6.5.46.3
SYSPLLSSCTRL0	R/W	0x590	PLL spread spectrum control 0	0x000 40000	6.5.46.5
SYSPLLSSCTRL1	R/W	0x594	PLL spread spectrum control 1	0x1000 0000	6.5.46.5
PDSLEEPCFG0	R/W	0x600	Sleep configuration register 0	0x00F8 0540	6.5.47
PDRUNCFG0	R/W	0x610	Power configuration register 0	0x00F8 0540	6.5.48
PDRUNCFGSET0	WO	0x620	Set bits in PDRUNCFG0	-	6.5.49
PDRUNCFGCLR0	WO	0x630	Clear bits in PDRUNCFG0	-	6.5.50
STARTER0	R/W	0x680	Start logic 0 wake-up enable register	0x0	6.5.51
RESERVED	-	0x684	-	-	-
STARTERSET0	WO	0x6A0	Set bits in STARTER0	-	6.5.52
RESERVED	-	0x6A4	-	-	-

Table 97. Register overview: Main system configuration (base address 0x4000 0000) *…continued*

Name	Access	Offset	Description	Reset value [1]	Section
STARTERCLR0	WO	0x6C0	Clear bits in STARTER0	-	6.5.53
RESERVED	-	0x6C4	-	-	-
HWWAKE	R/W	0x780	Configures special cases of hardware wake-up	0x0	6.5.54
RESERVED	-	0x800	-	-	-
RESERVED	-	0x804	-	-	-
RESERVED	-	0x808	-	-	-
RESERVED	-	0x80C	-	-	-
RESERVED	-	0xE04	-	-	-
JTAGIDCODE	RO	0xFF4	JTAG ID code register	see table	6.5.55
DEVICE_ID0	RO	0xFF8	Part ID register	Note [5]	6.5.56
DEVICE_ID1	RO	0xFFC	Boot ROM and die revision register	Note [5]	6.5.57

[1] Reset Value reflects the data stored in defined bits only. Reserved bits assumed to be 0.

[2] Determined by the voltage levels on device pins upon power-on reset.

[3] Determined by the voltage levels on device pins when a reset other than power-on reset occurs.

[4] Depends on the source of the most recent reset.

[5] Part dependent.

Table 98. Register overview: Asynchronous system configuration (base address 0x4004 0000)

Name	Access	Offset	Description	Reset value [1]	Section
ASYNCPRESETCTRL	R/W	0x000	Async peripheral reset control	0x0	6.5.58
ASYNCPRESETCTRLSET	WO	0x004	Set bits in ASYNCPRESETCTRL	-	6.5.59
ASYNCPRESETCTRLCLR	WO	0x008	Clear bits in ASYNCPRESETCTRL	-	6.5.60
ASYNCAPBCLKCTRL	R/W	0x010	Async peripheral clock control	0x0	6.5.61
ASYNCAPBCLKCTRLSET	WO	0x014	Set bits in ASYNCAPBCLKCTRL	-	6.5.62
ASYNCAPBCLKCTRLCLR	WO	0x018	Clear bits in ASYNCAPBCLKCTRL	-	6.5.63
ASYNCAPBCLKSELA	R/W	0x020	Async APB clock source select A	0x0	6.5.64

[1] Reset Value reflects the data stored in defined bits only. Reserved bits assumed to be 0.

Table 99. Register overview: Other system configuration (base address 0x4002 0000)

Name	Access	Offset	Description	Reset value [1]	Section
BODCTRL	R/W	0x44	Brown-Out Detect control	0x0	6.5.65

[1] Reset Value reflects the data stored in defined bits only. Reserved bits assumed to be 0.

6.5.1 AHB matrix priority register

The Multilayer AHB Matrix arbitrates between several masters, only if they attempt to access the same matrix slave port at the same time. Care should be taken if the value in this register is changed, improper settings can seriously degrade performance.

Priority values are 3 = highest, 0 = lowest. When the priority is the same, the master with the lower master number is given priority.

Table 100. AHB matrix priority register 0 (AHBMATPRIO, main syscon: offset 0x010) bit description

Bit	Symbol	Master number	Description	Reset value
5:0	-	-	Reserved. Read value is undefined, only zero should be written	-
7:6	PRI_M0	3	Cortex-M0+ bus priority	0x0
9:8	PRI_USB	4	USB interface priority.	0x0
11:10	PRI_DMA	5	DMA controller priority.	0x0
31:12	-	-	Reserved. Read value is undefined, only zero should be written.	-

6.5.2 System tick counter calibration register

This register allows software to set up a default value for the SYST_CALIB register in the System Tick Timer of each CPU. See Chapter 20.

Table 101. System tick timer calibration register (SYSTCKCAL, main syscon: offset 0x040) bit description

Bit	Symbol	Description	Reset value
23:0	CAL	System tick timer calibration value.	0x0
24	SKEW	Initial value for the Systick timer.	
25	NOREF	Initial value for the Systick timer.	
31:26	-	Reserved	-

6.5.3 NMI source selection register

The NMI source selection register selects a peripheral interrupts as source for the NMI interrupt of both CPUs. For a list of all peripheral interrupts and their IRQ numbers see Table 79. For a description of the NMI functionality, see Chapter 35.

Remark: To change the interrupt source for the NMI, the NMI source must first be disabled by writing 0 to the NMIEN bit. Then change the source by updating the IRQN bits and re-enable the NMI source by setting NMIEN.

Table 102. NMI source selection register (NMISRC, main syscon: offset 0x048) bit description

Bit	Symbol	Description	Reset value
7:0	-	Reserved. Read value is undefined, only zero should be written.	-
13:8	IRQM0	The IRQ number of the interrupt that acts as the Non-Maskable Interrupt (NMI) for the Cortex-M0+, if enabled by NMIENM0. Present on selected devices.	0x0
29:14	-	Reserved. Read value is undefined, only zero should be written.	-
30	NMIENM0	Write a 1 to this bit to enable the Non-Maskable Interrupt (NMI) source selected by IRQM0. Present on selected devices.	0x0
31	NMIENM4	Write a 1 to this bit to enable the Non-Maskable Interrupt (NMI) source selected by IRQM4.	0x0

Remark: If the NMISRC register is used to select an interrupt as the source of Non-Maskable interrupts, and the selected interrupt is enabled, one interrupt request can result in both a Non-Maskable and a normal interrupt. This can be avoided by disabling the normal interrupt in the NVIC.

6.5.4 Asynchronous APB Control register

ASYNCAPBCTRL contains a global enable bit for the asynchronous APB bridge and subsystem, allowing connection to the associated peripherals.

Table 103. Asynchronous APB Control register (ASYNCAPBCTRL, main syscon: offset 0x04C) bit description

Bit	Symbol	Value	Description	Reset value
0	ENABLE		Enables the asynchronous APB bridge and subsystem.	0x1
		0	Disabled. Asynchronous APB bridge is disabled.	
		1	Enabled. Asynchronous APB bridge is enabled.	
31:1	-	-	Reserved. Read value is undefined, only zero should be written.	-

6.5.5 POR captured value of port 0

The PIOPORCAP0 register captures the state of GPIO port 0 at power-on-reset. Each bit represents the power-on reset state of one GPIO pin. This register is a read-only register.

Table 104. POR captured PIO status register 0 (PIOPORCAP0, main syscon: offset 0x0C0) bit description

Bit	Symbol	Description	Reset value
31:0	PIOPORCAP	State of PIO0_31 through PIO0_0 at power-on reset	Depends on external circuitry

6.5.6 POR captured value of port 1

The PIOPORCAP1 register captures the state of GPIO port 1 at power-on-reset. Each bit represents the power-on reset state of one GPIO pin. This register is a read-only register.

Table 105. POR captured PIO status register 1 (PIOPORCAP1, main syscon: offset 0x0C4) bit description

Bit	Symbol	Description	Reset value
31:0	PIOPORCAP	State of PIO1_31 through PIO1_0 at power-on reset	Depends on external circuitry

6.5.7 Reset captured value of port 0

The PIORESCAP0 register captures the state of GPIO port 0 when a reset other than a power-on reset occurs. Each bit represents the reset state of one GPIO pin. This register is a read-only register.

Table 106. Reset captured PIO status register 0 (PIORESCAP0, main syscon: offset 0x0D0) bit description

Bit	Symbol	Description	Reset value
31:0	PIORESCAP	State of PIO0_31 through PIO0_0 for resets other than POR.	Depends on external circuitry

6.5.8 Reset captured value of port 1

The PIORESCAP0 register captures the state of GPIO port 1 when a reset other than a power-on reset occurs. Each bit represents the reset state of one GPIO pin. This register is a read-only register.

Table 107. Reset captured PIO status register 1 (PIORESCAP1, main syscon: offset 0x0D4) bit description

Bit	Symbol	Description	Reset value
31:0	PIORESCAP	State of PIO1_31 through PIO1_0 for resets other than POR.	Depends on external circuitry

6.5.9 Peripheral reset control register 0

The PRESETCTRL0 register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset.

Remark: It is recommended that changes to the PRESETCTRL registers be accomplished by using the related PRESETCTRLSET and PRESETCTRLCLR registers. This avoids any unintentional setting or clearing of other bits.

Table 108. Peripheral reset control register 0 (PRESETCTRL0, main syscon: offset 0x100) bit description

Bit	Symbol	Description	Reset value
6:0	-	Reserved. Read value is undefined, only zero should be written.	0x0
7	FLASH_RST	Flash controller reset control. Note that this reset must not be asserted when code is or will be executing from the flash. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
8	FMC_RST	Flash accelerator reset control. Note that the FMC must not be reset while executing from flash, and must be reconfigured after reset. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
10:9	-	Reserved. Read value is undefined, only zero should be written.	0x0
11	MUX_RST	Input mux reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
12	-	Reserved. Read value is undefined, only zero should be written.	0x0
13	IOCON_RST	IOCON reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
14	GPIO0_RST	GPIO0 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
15	GPIO1_RST	GPIO1 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
17:16	-	Reserved. Read value is undefined, only zero should be written.	0x0
18	PINT_RST	Pin interrupt (PINT) reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
19	GINT_RST	Grouped interrupt (GINT) reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
20	DMA_RST	DMA reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
21	CRC_RST	CRC generator reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
22	WWDT_RST	Watchdog timer reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
26:23	-	Reserved. Read value is undefined, only zero should be written.	0x0
27	ADC0_RST	ADC0 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
31:28	-	Reserved. Read value is undefined, only zero should be written.	-

6.5.10 Peripheral reset control register 1

The PRESETCTRL1 register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset.

Remark: It is recommended that changes to the PRESETCTRL registers be accomplished by using the related PRESETCTRLSET and PRESETCTRLCLR registers. This avoids any unintentional setting or clearing of other bits.

Table 109. Peripheral reset control register 1 (PRESETCTRL1, main syscon: offset 0x104) bit description

Bit	Symbol	Description	Reset value
0	MRT_RST	Multi-rate timer (MRT) reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
1	-	Reserved. Read value is undefined, only zero should be written.	-
2	SCT0_RST	SCTimer/PWM 0 (SCT0) reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
9:3	-	Reserved. Read value is undefined, only zero should be written.	0x0
10	UTICK_RST	Micro-tick Timer reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
11	FC0_RST	Flexcomm Interface 0 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
12	FC1_RST	Flexcomm Interface 1 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
13	FC2_RST	Flexcomm Interface 2 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
14	FC3_RST	Flexcomm Interface 3 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
15	FC4_RST	Flexcomm Interface 4 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
16	FC5_RST	Flexcomm Interface 5 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
17	FC6_RST	Flexcomm Interface 6 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
18	FC7_RST	Flexcomm Interface 7 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
22:19	-	Reserved. Read value is undefined, only zero should be written.	0x0
24:23	-	Reserved. Read value is undefined, only zero should be written.	0x0
25	USB_RST	USB reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
26	CTIMER0_RST	CTIMER0 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
27	CTIMER1_RST	CTIMER1 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
31:28	-	Reserved. Read value is undefined, only zero should be written.	-

6.5.11 Peripheral reset control set register 0

Writing a 1 to a bit position in PRESETCTRLSET0 sets the corresponding position in PRESETCTRL0. This is a write-only register. For bit assignments, see Table 108.

Table 110. Peripheral reset control set register 0 (PRESETCTRLSET0, main syscon: offset 0x120) bit description

Bit	Symbol	Description	Reset value
31:0	RST_SET0	Writing ones to this register sets the corresponding bit or bits in the PRESETCTRL0 register, if they are implemented. Bits that do not correspond to defined bits in PRESETCTRL0 are reserved and only zeroes should be written to them.	-

6.5.12 Peripheral reset control set register 1

Writing a 1 to a bit position in PRESETCTRLSET1 sets the corresponding position in PRESETCTRL1. This is a write-only register. For bit assignments, see Table 109.

Table 111. Peripheral reset control set register 1 (PRESETCTRLSET1, main syscon: offset 0x124) bit description

Bit	Symbol	Description	Reset value
31:0	RST_SET1	Writing ones to this register sets the corresponding bit or bits in the PRESETCTRL1 register, if they are implemented. Bits that do not correspond to defined bits in PRESETCTRL1 are reserved and only zeroes should be written to them.	-

6.5.13 Peripheral reset control clear register 0

Writing a 1 to a bit position in PRESETCTRLCLR0 clears the corresponding position in PRESETCTRL0. This is a write-only register. For bit assignments, see Table 108.

Table 112. Peripheral reset control clear register 0 (PRESETCTRLCLR0, main syscon: offset 0x140) bit description

Bit	Symbol	Description	Reset value
31:0	RST_CLR0	Writing ones to this register clears the corresponding bit or bits in the PRESETCTRL0 register, if they are implemented. Bits that do not correspond to defined bits in PRESETCTRL0 are reserved and only zeroes should be written to them.	-

6.5.14 Peripheral reset control clear register 1

Writing a 1 to a bit position in PRESETCTRLCLR1 clears the corresponding position in PRESETCTRL1. This is a write-only register. For bit assignments, see Table 109.

Table 113. Peripheral reset control clear register 1 (PRESETCTRLCLR1, main syscon: offset 0x144) bit description

Bit	Symbol	Description	Reset value
31:0	RST_CLR1	Writing ones to this register clears the corresponding bit or bits in the PRESETCTRL1 register, if they are implemented. Bits that do not correspond to defined bits in PRESETCTRL1 are reserved and only zeroes should be written to them.	-

6.5.15 System reset status register

The SYSRSTSTAT register shows the source of the latest reset event. The bits are cleared by writing a one to any of the bits. The POR event clears all other bits in this register. If another reset signal - for example the external $\overline{\text{RESET}}$ pin - remains asserted after the POR signal is negated, then its bit is set to detected.

Table 114. System reset status register (SYSRSTSTAT, main syscon: offset 0x01F0) bit description

Bit	Symbol	Value	Description
0	POR		POR reset status. Assertion of the POR signal sets this bit, and clears all of the other bits in this register. But if another Reset signal (e.g., External Reset) remains asserted after the POR signal is negated, then its bit is set. This bit is not affected by any of the other sources of Reset.
		0	No POR detected.
		1	POR detected. Writing a one clears this flag.
1	EXTRST		Status of the external $\overline{\text{RESET}}$ pin. External reset status. Assertion of the external RESET signal sets this bit.
		0	No reset event detected.
		1	Reset detected. This bit is cleared by software writing a one to this bit, and by POR.
2	WDT		Status of the Watchdog reset. This bit is set when the Watchdog Timer times out and the WDTRESET bit in the Watchdog Mode Register is 1.
		0	No WDT reset detected.
		1	WDT reset detected. Writing a one clears this flag. This bit is cleared by software writing a one to this bit, and by POR.
3	BOD		Status of the Brown-out detect reset. This bit is set when the VDD voltage reaches a level below the BOD reset trip level. If the VDD voltage dips from the normal operating range to below the BOD reset trip level and recovers, the BOD bit will be set to 1. If the VDD voltage dips from the normal operating range to below the BOD reset trip level and continues to decline to the level at which POR is asserted, the BOD bit is cleared. If the VDD voltage rises continuously from the POR assertion level to a level above the BOD reset trip level, the BOD bit will be set to 1. This bit is cleared by software writing a one to this bit, and by POR. Note: Only in the case where a reset occurs and the POR = 0, the BODR bit indicates if the VDD voltage was below the BOD reset trip level.
		0	No BOD reset detected.
		1	BOD reset detected. Writing a one clears this flag.
4	SYSRST		Status of the software system reset. This bit is set if the processor has been reset due to a system reset request. Setting the SYSRESETREQ bit in the Cortex-M0+ AIRCR register causes a chip reset. This bit is cleared by software writing a one to this bit, and by POR.
		0	No System reset detected.
		1	System reset detected. Writing a one clears this flag.
31:5	-	-	Reserved

6.5.16 AHB Clock Control register 0

The AHBCLKCTRL0 register enables the clocks to individual system and peripheral blocks. The system clock (bit 0) provides the clock for the AHB, the APB bridge, the CPU, the SYSCON block, and the PMU. This clock cannot be disabled.

Remark: It is strongly recommended that changes to the AHBCLKCTRL registers be accomplished by using the related AHBCLKCTRLSET and AHBCLKCTRLCLR registers. This avoids any unintentional setting or clearing of other bits, which could have detrimental effects.

Table 115. AHB Clock Control register 0 (AHBCLKCTRL0, main syscon: offset 0x200) bit description

Bit	Symbol	Description	Reset value after boot
0	-	Reserved. This read-only bit cannot be cleared.	0x1
1	ROM	Enables the clock for the Boot ROM. 0 = Disable; 1 = Enable.	0x1
6:2	-	Reserved	-
7	FLASH	Enables the clock for the flash controller. 0 = Disable; 1 = Enable. This clock is needed for flash programming, not for flash read.	0x1
8	FMC	Enables the clock for the Flash accelerator. 0 = Disable; 1 = Enable. This clock is needed if the flash is being read.	0x1
10:9	-	Reserved	-
11	INPUTMUX	Enables the clock for the input muxes. 0 = Disable; 1 = Enable.	0x0
12	-	Reserved	0x0
13	IOCON	Enables the clock for the IOCON block. 0 = Disable; 1 = Enable.	0x0
14	GPIO0	Enables the clock for the GPIO0 port registers. 0 = Disable; 1 = Enable.	0x0
15	GPIO1	Enables the clock for the GPIO1 port registers. 0 = Disable; 1 = Enable.	0x0
17:16	-	Reserved	0x0
18	PINT	Enables the clock for the pin interrupt block.0 = Disable; 1 = Enable.	0x0
19	GINT	Enables the clock for the grouped pin interrupt block. 0 = Disable; 1 = Enable.	0x0
20	DMA	Enables the clock for the DMA controller. 0 = Disable; 1 = Enable.	0x0
21	CRC	Enables the clock for the CRC engine. 0 = Disable; 1 = Enable.	0x0
22	WWDT	Enables the clock for the Watchdog Timer. 0 = Disable; 1 = Enable.	0x0
23	RTC	Enables the bus clock for the RTC. 0 = Disable; 1 = Enable.	0x0
25:24	-	Reserved	-
26	-	Reserved	-
27	ADC0	Enables the clock for the ADC0 register interface. 0 = Disable; 1 = Enable.	0x0
31:28	-	Reserved	-

6.5.17 AHB Clock Control register 1

The AHBCLKCTRL1 register enables the clocks to individual peripheral blocks.

Table 116. AHB Clock Control register 1 (AHBCLKCTRL1, main syscon: offset 0x204) bit description

Bit	Symbol	Description	Reset value
0	MRT	Enables the clock for the Multi-Rate Timer. 0 = Disable; 1 = Enable.	0x0
1	-	Reserved	-
2	SCT0	Enables the clock for SCT0. 0 = Disable; 1 = Enable.	0x0
9:3	-	Reserved	-
10	UTICK	Enables the clock for the Micro-tick Timer. 0 = Disable; 1 = Enable.	0x0
11	FLEXCOMM0	Enables the clock for Flexcomm Interface 0. 0 = Disable; 1 = Enable.	0x0
12	FLEXCOMM1	Enables the clock for Flexcomm Interface 1. 0 = Disable; 1 = Enable.	0x0
13	FLEXCOMM2	Enables the clock for Flexcomm Interface 2. 0 = Disable; 1 = Enable.	0x0
14	FLEXCOMM3	Enables the clock for Flexcomm Interface 3. 0 = Disable; 1 = Enable.	0x0
15	FLEXCOMM4	Enables the clock for Flexcomm Interface 4. 0 = Disable; 1 = Enable.	0x0
16	FLEXCOMM5	Enables the clock for Flexcomm Interface 5. 0 = Disable; 1 = Enable.	0x0
17	FLEXCOMM6	Enables the clock for Flexcomm Interface 6. 0 = Disable; 1 = Enable.	0x0
18	FLEXCOMM7	Enables the clock for Flexcomm Interface 7. 0 = Disable; 1 = Enable.	0x0
24:19	-	Reserved	-
25	USB	Enables the clock for the USB interface. 0 = Disable; 1 = Enable.	0x0
26	CTIMER0	Enables the clock for timer CTIMER0. 0 = Disable; 1 = Enable.	0x0
27	CTIMER1	Enables the clock for timer CTIMER1. 0 = Disable; 1 = Enable.	0x0
31:28	-	Reserved	-

6.5.18 AHB Clock Control Set register 0

Writing a 1 to a bit position in AHBCLKCTRLSET0 sets the corresponding position in AHBCLKCTRL0. This is a write-only register. For bit assignments, see Table 115.

Table 117. Clock control set register 0 (AHBCLKCTRLSET0, main syscon: offset 0x220) bit description

Bit	Symbol	Description	Reset value
31:0	CLK_SET0	Writing ones to this register sets the corresponding bit or bits in the AHBCLKCTRL0 register, if they are implemented. Bits that do not correspond to defined bits in AHBCLKCTRL0 are reserved and only zeroes should be written to them.	-

6.5.19 AHB Clock Control Set register 1

Writing a 1 to a bit position in AHBCLKCTRLSET1 sets the corresponding position in AHBCLKCTRL1. This is a write-only register. For bit assignments, see Table 116.

Table 118. Clock control set register 1 (AHBCLKCTRLSET1, main syscon: offset 0x224) bit description

Bit	Symbol	Description	Reset value
31:0	CLK_SET1	Writing ones to this register sets the corresponding bit or bits in the AHBCLKCTRL1 register, if they are implemented. Bits that do not correspond to defined bits in AHBCLKCTRL1 are reserved and only zeroes should be written to them.	-

6.5.20 AHB Clock Control Clear register 0

Writing a 1 to a bit position in AHBCLKCTRLCLR0 clears the corresponding position in AHBCLKCTRL0. This is a write-only register. For bit assignments, see Table 115.

Table 119. Clock control clear register 0 (AHBCLKCTRLCLR0, main syscon: offset 0x240) bit description

Bit	Symbol	Description	Reset value
31:0	CLK_CLR0	Writing ones to this register clears the corresponding bit or bits in the AHBCLKCTRL0 register, if they are implemented. Bits that do not correspond to defined bits in AHBCLKCTRL0 are reserved and only zeroes should be written to them.	-

6.5.21 AHB Clock Control Clear register 1

Writing a 1 to a bit position in AHBCLKCTRLCLR1 clears the corresponding position in AHBCLKCTRL1. This is a write-only register. For bit assignments, see Table 116.

Table 120. Clock control clear register 1 (AHBCLKCTRLCLR1, main syscon: offset 0x244) bit description

Bit	Symbol	Description	Reset value
31:0	CLK_CLR1	Writing ones to this register clears the corresponding bit or bits in the AHBCLKCTRL1 register, if they are implemented. Bits that do not correspond to defined bits in AHBCLKCTRL1 are reserved and only zeroes should be written to them.	-

6.5.22 Main clock source select register A

This register selects one of the internal oscillators, FRO, system oscillator, or watchdog oscillator. The oscillator selected is then one of the inputs to the main clock source select register B (see Table 122), which selects the clock source for the main clock. All clocks to the core, memories, and peripherals on the synchronous APB bus are derived from the main clock.

Remark: This selection is internally synchronized: the clock being switched from and the clock being switched to must both be running and have occurred in specific states before the selection actually changes.

Table 121. Main clock source select register A (MAINCLKSELA, main syscon: offset 0x280) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SEL		Clock source for main clock source selector A	0x0
		0x0	FRO 12 MHz (fro_12m)	
		0x1	CLKIN (clk_in)	
		0x2	Watchdog oscillator (wdt_clk)	
		0x3	FRO 96 or 48 MHz (fro_hf)	
31:2	-	-	Reserved	-

6.5.23 Main clock source select register B

This register selects the clock source for the main clock. All clocks to the core, memories, and peripherals are derived from the main clock.

One input to this register is the main clock source select register A (see Table 121), which selects one of the three internal oscillators, FRO, system oscillator, or watchdog oscillator.

Remark: This selection is internally synchronized: the clock being switched from and the clock being switched to must both be running and have occurred in specific states before the selection actually changes.

Table 122. Main clock source select register B (MAINCLKSELB, main syscon: offset 0x284) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SEL		Clock source for main clock source selector B. Selects the clock source for the main clock.	0x0
		0x0	MAINCLKSELA. Use the clock source selected in MAINCLKSELA register.	
		0x1	Reserved setting	
		0x2	System PLL output (pll_clk)	
		0x3	RTC oscillator 32 kHz output (32k_clk)	
31:2	-	-	Reserved	-

6.5.24 CLKOUT clock source select register A

This register pre-selects one of the internal oscillators for the clock sources visible on the CLKOUT pin.

Table 123. CLKOUT clock source select register (CLKOUTSELA, main syscon: offset 0x288) bit description

Bit	Symbol	Value	Description	Reset value
2:0	SEL		CLKOUT clock source selection	0x7
		0x0	Main clock (main_clk)	
		0x1	CLKIN (clk_in)	
		0x2	Watchdog oscillator (wdt_clk)	
		0x3	FRO 96 or 48 MHz (fro_hf)	
		0x4	PLL output (pll_clk)	
		0x5	FRO 12 MHz (fro_12m)	
		0x6	RTC oscillator 32 kHz output (32k_clk)	
		0x7	None, this may be selected in order to reduce power when no output is needed.	
31:3	-	-	Reserved	-

6.5.25 System PLL clock source select register

This register selects the clock source for the system PLL.

Table 124. System PLL clock source select register (SYSPLLCLKSEL, main syscon: offset 0x290) bit description

Bit	Symbol	Value	Description	Reset value
2:0	SEL		System PLL clock source selection	0x7
		0x0	FRO 12 MHz (fro_12m)	
		0x1	CLKIN (clk_in)	
		0x3	RTC 32 kHz clock (32k_clk)	
		0x7	None, this may be selected in order to reduce power when no output is needed.	
			Other values are reserved settings.	
31:3	-	-	Reserved	-

6.5.26 ADC clock source select register

This register selects a clock source for the 12-bit ADCs that is to the system clock. To use a clock other than the Main clock, select the asynchronous clock mode in the ADC control register.

Table 125. ADC clock source select (ADCCLKSEL, main syscon: offset 0x2A4) bit description

Bit	Symbol	Value	Description	Reset value
2:0	SEL		ADC clock source selection	0x7
		0x0	Main clock (main_clk)	
		0x1	System PLL output (pll_clk)	
		0x2	FRO 96 or 48 MHz (fro_hf)	
		0x7	None, this may be selected in order to reduce power when no output is needed.	
			Other values are reserved settings.	
31:3	-	-	Reserved	-

6.5.27 USB clock source select register

This register selects a clock source for the USB device.

Table 126. USB clock source select register (USBCLKSEL, main syscon: offset 0x2A8) bit description

Bit	Symbol	Value	Description	Reset value
2:0	SEL		USB device clock source selection	0x7
		0x0	FRO 96 or 48 MHz (fro_hf)	
		0x1	System PLL output (pll_clk)	
		0x2	Main clock (main_clk)	
		0x7	None, this may be selected in order to reduce power when no output is needed.	
			Other values are reserved settings.	
31:3	-	-	Reserved	-

6.5.28 Flexcomm Interface clock source select registers

These registers select the clock source for each Flexcomm Interface serial peripheral. Each Flexcomm Interface has its own clock source selection.

Table 127. Flexcomm Interface clock source select registers (FCLKSEL[0:7], main syscon: offsets 0x2B0 through 2CC) bit description

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Flexcomm Interface clock source selection. One per Flexcomm Interface.	0x7
		0x0	FRO 12 MHz (fro_12m)	
		0x1	FRO 96 or 48 MHz (fro_hf)	
		0x2	System PLL output (pll_clk)	
		0x3	MCLK pin input, when selected in IOCON (mclk_in)	
		0x4	FRG clock, the output of the fractional rate generator (frg_clk)	
		0x7	None, this may be selected in order to reduce power when no output is needed.	
			Other values are reserved settings.	
31:3	-	-	Reserved	-

6.5.29 MCLK clock source select register

This register selects a clock to provide to the MCLK output function. In a system using I^{2}S and/or digital microphone, this should be related to the clock used by those functions.

Table 128. MCLK clock source select register (MCLKCLKSEL, main syscon: offset 0x2E0) bit description

Bit	Symbol	Value	Description	Reset value
2:0	SEL		MCLK source select. This may be used by Flexcomm Interfaces that support I2S, and/or by the digital microphone subsystem.	0x7
		0x0	FRO 96 or 48 MHz (fro_hf)	
		0x1	System PLL output (pll_clk)	
		0x2	Main clock (main_clk)	
		0x7	None, this may be selected in order to reduce power when no output is needed.	
			Other values are reserved settings.	
31:3	-	-	Reserved. Read value is undefined, only zero should be written.	-

6.5.30 FRG clock source select register

This register selects a clock source for the Fractional Rate Generator.

Table 129. FRG clock source select register (FRGCLKSEL, main syscon: offset 0x2E8) bit description

Bit	Symbol	Value	Description	Reset value
2:0	SEL		Fractional Rate Generator clock source select.	0x7
		0x0	Main clock (main_clk)	
		0x1	System PLL output (pll_clk)	
		0x2	FRO 12 MHz (fro_12m)	
		0x3	FRO 96 or 48 MHz (fro_hf)	
		0x7	None, this may be selected in order to reduce power when no output is needed.	
			Other values are reserved settings.	
31:3	-	-	Reserved. Read value is undefined, only zero should be written.	-

6.5.31 SYSTICK clock divider register

This register configures the SYSTICK peripheral clock.

Table 130. SYSTICK clock divider (SYSTICKCLKDIV, main syscon: offset 0x300) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0x0
31:8	-	Reserved. Read value is undefined, only zero should be written.	-

6.5.32 AHB clock divider register

This register controls how the main clock is divided to provide the system clock to the AHB bus, CPU, and memories.

Table 131. System clock divider register (AHBCLKDIV, main syscon: offset 0x380) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0x0
31:8	-	Reserved. Read value is undefined, only zero should be written.	-

6.5.33 CLKOUT clock divider register

This register determines the divider value for the clock signal on the CLKOUT pin.

Table 132. CLKOUT clock divider register (CLKOUTDIV, main syscon: offset 0x384) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0x0
31:8	-	Reserved. Read value is undefined, only zero should be written.	-

6.5.34 ADC clock source divider register

This register divides the clock to the ADC.

Table 133. ADC clock source divider (ADCCLKDIV, main syscon: offset 0x394) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0x0
31:8	-	Reserved. Read value is undefined, only zero should be written.	-

6.5.35 USB clock divider register

This register determines the divider value for the USB function clock.

Table 134. USB clock divider register (USBCLKDIV, main syscon: offset 0x398) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0x0
31:8	-	Reserved. Read value is undefined, only zero should be written.	-

6.5.36 Fractional baud rate generator register

All Flexcomm Interfaces have, as one of their possible clock sources, a common clock (see Figure 9), that can be adjusted by a fractional divider. This is intended primarily to create a base baud rate clock for USART functions, but may potentially be used for other purposes. This register sets the MULT and DIV values for the fractional rate generator.

Remark: When the FRG is used to create a clock for use by one or more Flexcomm Interfaces (the typical use of the FRG), the FRG output frequency should not be higher than 48 MHz.

The output rate is:

Flexcomm Interface function clock = (clock selected via FRGCLKSEL) / (1 + MULT / DIV)

The clock used by the fractional rate generator is selected via the FRGSEL register (see Section 6.5.30).

Remark: To use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

See also Section 24.3.1 “Configure the Flexcomm Interface clock and USART baud rate” and Section 24.7.2 “Clocking and baud rates”.

Table 135. Fractional baud rate generator register (FRGCTRL, main syscon: offset 0x3A0) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Denominator of the fractional divider. DIV is equal to the programmed value +1. Always set to 0xFF to use with the fractional baud rate generator.	0xFF
15:8	MULT	Numerator of the fractional divider. MULT is equal to the programmed value.	0x0
31:16	-	Reserved	-

6.5.37 MCLK clock divider register

This register determines the divider value for the MCLK output, if used by the application.

Table 136. MCLK clock divider register (MCLKDIV, main syscon: offset 0x3AC) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock divider value. 0: Divide by 1. … 255: Divide by 256.	0x0
31:8	-	Reserved. Read value is undefined, only zero should be written.	-

6.5.38 Flash configuration register

Depending on the system clock frequency, access to the flash memory can be configured with various access times by writing to the FLASHCFG register. It is recommended to use the power mode entry API (see Chapter 8 "LPC51U68 Power profiles/Power control API") to configure device operation in order to achieve lower power operation. However, flash timing can also be set up by user software as shown in Table 137.

Enabling buffering, acceleration, and prefetch will substantially improve performance. Buffering saves power by allowing previously accessed information to be reused without a flash read. Acceleration saves power by reducing CPU stalls. Prefetch typically has a small power cost due to some flash reads being performed that ultimately are not needed. Additional information about the flash accelerator may be found in Section 6.6.4.

Remark: Improper setting of this register may result in incorrect operation of the flash memory. Do not change the flash access time when using the power API in low-power mode.

Table 137. Flash configuration register (FLASHCFG, main syscon: offset 0x400) bit description

Bit	Symbol	Value	Description	Reset value
1:0	FETCHCFG		Instruction fetch configuration. This field determines how flash accelerator buffers are used for instruction fetches.	0x2
		0x0	Instruction fetches from flash are not buffered. Every fetch request from the CPU results in a read of the flash memory. This setting may use significantly more power than when buffering is enabled.	
		0x1	One buffer is used for all instruction fetches.	
		0x2	All buffers may be used for instruction fetches.	
		0x3	Reserved setting, do not use.	
3:2	DATACFG		Data read configuration. This field determines how flash accelerator buffers are used for data accesses.	0x2
		0x0	Data accesses from flash are not buffered. Every data access from the CPU results in a read of the flash memory.	
		0x1	One buffer is used for all data accesses.	
		0x2	All buffers may be used for data accesses.	
		0x3	Reserved setting, do not use.	
4	ACCEL		Acceleration enable.	0x1
		0	Flash acceleration is disabled. Every flash read (including those fulfilled from a buffer) takes FLASHTIM + 1 system clocks. This allows more determinism at a cost of performance.	
		1	Flash acceleration is enabled. Performance is enhanced, dependent on other FLASHCFG settings.	
5	PREFEN		Prefetch enable.	0x0
		0	No instruction prefetch is performed.	
		1	If the FETCHCFG field is not 0, the next flash line following the current execution address is automatically prefetched if it is not already buffered.	

Table 137. Flash configuration register (FLASHCFG, main syscon: offset 0x400) bit description

Bit	Symbol	Value	Description	Reset value
6	PREFOVR		Prefetch override. This bit only applies when PREFEN = 1 and a buffered instruction is completing for which the next flash line is not already buffered or being prefetched.	0x0
		0	Any previously initiated prefetch will be completed.	
		1	Any previously initiated prefetch will be aborted, and the next flash line following the current execution address will be prefetched if not already buffered.	
11:7	-	-	Reserved	-
15:12	FLASHTIM		Flash memory access time. The number of system clocks used for flash accesses is equal to FLASHTIM +1.	0x0
		0x0	1 system clock flash access time (for system clock rates up to 12 MHz).	
		0x1	2 system clocks flash access time (for system clock rates up to 24 MHz).	
		0x2	3 system clocks flash access time (for system clock rates up to 48 MHz).	
		0x3	4 system clocks flash access time (for system clock rates up to 72 MHz).	
		0x4	5 system clocks flash access time (for system clock rates up to 84 MHz).	
		0x5	6 system clocks flash access time (for system clock rates up to 100 MHz).	
			Other values give: “FLASHTIM” + 1 system clocks flash access time.	
31:16	-	-	Reserved	-

6.5.39 USB clock control register

This register controls the polarity of the USB_NEED_CLK signal for triggering the USB wake-up interrupt. For details of how to use the USB_NEED_CLK signal for waking up the part from deep-sleep mode, see Section 22.7.6.

Table 138. USB clock control register (USBCLKCTRL, main syscon: offset 0x40C) bit description

Bit	Symbol	Value	Description	Reset value
0	-	-	Reserved	-
1	POL_CLK		USB_NEED_CLK polarity for triggering the USB wake-up interrupt.	0x0
		0	Falling edge of the USB_NEED_CLK triggers the USB wake-up (default).	
		1	Rising edge of the USB_NEED_CLK triggers the USB wake-up.	
31:2	-	-	Reserved	-

6.5.40 USB clock status register

This register is read-only and returns the status of the USB_NEED_CLK signal. For details of how to use the USB_NEED_CLK signal for waking up the part from deep-sleep mode, see Section 22.7.6.

Table 139. USB clock status register (USBCLKSTAT, main syscon: offset 0x410) bit description

Bit	Symbol	Value	Description	Reset value
0	NEED_CLKST		USB_NEED_CLK signal status	0x0
		0	Low	
		1	High	
31:1	-	-	Reserved	-

6.5.41 Frequency measure function control register

This register starts the frequency measurement function and stores the result in the CAPVAL field. The target frequency can be calculated as follows with the frequencies given in MHz:

$F_{target} = (CAPVAL - 2) \times F_{reference}/2^{14}$

Select the reference and target frequencies using the FREQMEAS_REF and FREQMEAS_TARGET before starting a frequency measurement by setting the PROG bit in FREQMECTRL.

Table 140. Frequency measure function control register (FREQMECTRL, main syscon: offset 0x418) bit description

Bit	Symbol	Description	Reset value
13:0	CAPVAL	Stores the capture result which is used to calculate the frequency of the target clock. This field is read-only.	0x0
30:14	-	Reserved. Read value is undefined, only zero should be written.	-
31	PROG	Set this bit to one to initiate a frequency measurement cycle. Hardware clears this bit when the measurement cycle has completed and there is valid capture data in the CAPVAL field (bits 13:0).	0x0

Also see:

- Section 6.2.3 “Measure the frequency of a clock signal”
- Section 6.6.6 “Frequency measure function”
- Frequency reference clock select register (FREQMEAS_REF) - Section 10.6.4
- Frequency target clock select register (FREQMEAS_TARGET) - Section 10.6.5

6.5.42 MCLK input/output control register

This register selects the direction of the pin associated with MCLK when MCLK is the elected function on that pin.

Table 141. MCLK input/output control register (MCLKIO, main syscon: offset 0x420) bit description

Bit	Symbol	Value	Description	Reset value
0	DIR		MCLK direction control.	0x0
		0	The MCLK function is an input.	
		1	The MCLK function is an output.	
31:1	-	-	Reserved, only zero should be written.	-

6.5.43 FRO Control register

This register is used to select the on-chip FRO oscillator for the higher frequency clock, as well as configuration for the automatic USB rate adjustment mode. The trim value is factory-preset for the 48 MHz oscillator and written by the boot code on start-up.

The following procedure must be followed to select desired higher frequency:

1. Switch the main clock to any clock other than FRO_HF (e.g. fro_12m, 32k_clk, clk_in, wdt_clk).
2. Call the FRO high frequency output setup API. See Section 8.4.5 for selecting 48 MHz or 96 MHz with trim.

3. Set bit 30 (HSPDCLK) in FROCTRL register to 1.
4. Switch the main clock to fro_hf clock.

Table 142. FRO control register (FROCTRL, main syscon: offset 0x500) bit description

Bit	Symbol	Value	Description	Reset value
13:0	-	-	Reserved, only zero should be written.	-
14	SEL	-	fro_hf_output frequency status bit. If read as 0, fro_hf is 48 MHz. If read as 1, fro_hf is 96 MHz.	0x1
15	-	-	Reserved, only zero should be written.	-
23:16	FREQTRIM	-	Frequency trim. Boot code configures this to a device-specific factory trim value for the 48 MHz FRO. If USBCLKADJ = 1, this field is read-only and provides the value resulting from USB rate adjustment. See the USBMODCFG flag regarding reading this field. Application code may adjust this field when USBCLKADJ = 0. A single step of FREQTRIM is roughly equivalent to 0.1% of the selected FRO frequency.	see description
24	USBCLKADJ		USB clock adjust mode.	0x0
		0	Normal operation.	
		1	Automatic USB rate adjustment mode. If the USB FS device peripheral is enabled and connected to a USB host, it provides clock adjustment information to the FRO based on SOF packets. USB rate adjustment requires a number of cycles to take place. the USBMODCHG bit (see below) indicates when initial adjustment is complete, and when later adjustments are in progress. **Remark:** Software must not alter TRIM and FREQTRIM while USBCLKADJ = 1. **Remark:** See USBCLKADJ usage notes below this table.	
25	USBMODCHG	-	USB Mode value Change flag. When 1, indicates that the USB trim is currently being updated (or is still starting up) and software should wait to read FREQTRIM. Update occurs at most once per millisecond.	0x0
29:26	-	-	Reserved, only zero should be written.	-
30	HSPDCLK		High speed clock enable. Allows disabling the highs-speed FRO output if it is not needed.	0x1
		0	The high-speed FRO output is disabled.	
		1	The selected high-speed FRO output (48 MHz or 96 MHz) is enabled.	
31	-	-	Reserved, only zero should be written.	0x0

Notes on using USBCLKADJ

When turning on USBCLKADJ, the current FREQTRIM value will be used as the starting value. From then on, the adjusted value will be used as long as enabled (whether USB is active or not).

If USBCLKADJ is turned off, the application may take one of two actions:

1. Read the register to pick up the adjusted FREQTRIM and then write back with the USBADJ cleared. The FRO will continue to use the adjusted value.
2. If software saved the original factory trimmed value of FREQTRIM, it can be written back as above.

6.5.44 Watchdog oscillator control register

This register controls the frequency of the watchdog oscillator, in the range of 6 kHz to 1.5 MHz. This oscillator is connected to the watchdog timer and the Micro-tick Timer. The low-power nature of this oscillator limits its accuracy to +/- 40% over temperature, voltage, and silicon processing variations. The actual frequency may be measured using the frequency measure block. See Section 6.2.3.

Table 143. Watchdog oscillator control register (WDTOSCCTRL, main syscon: offset 0x508) bit description

Bit	Symbol	Description	Reset value
4:0	DIVSEL	Divider select. Selects the value of the divider that adjusts the output of the oscillator. 0x00 = divide by 2 0x01 = divide by 4 0x02 = divide by 6 … 0x1E = divide by 62 0x1F = divide by 64	0x0
9:5	FREQSEL	Frequency select. Selects the frequency of the oscillator. 0x00 = invalid setting when watchdog oscillator is running 0x01 = 0.4 MHz 0x02 = 0.6 MHz 0x03 = 0.75 MHz 0x04 = 0.9 MHz 0x05 = 1.0 MHz 0x06 = 1.2 MHz 0x07 = 1.3 MHz 0x08 = 1.4 MHz 0x09 = 1.5 MHz 0x0A = 1.6 MHz 0x0B = 1.7 MHz 0x0C = 1.8 MHz 0x0D = 1.9 MHz 0x0E = 2.0 MHz 0x0F = 2.05 MHz 0x10 = 2.1 MHz 0x11 = 2.2 MHz 0x12 = 2.25 MHz 0x13 = 2.3 MHz 0x14 = 2.4 MHz 0x15 = 2.45 MHz 0x16 = 2.5 MHz 0x17 = 2.6 MHz 0x18 = 2.65 MHz 0x19 = 2.7 MHz 0x1A = 2.8 MHz 0x1B = 2.85 MHz 0x1C = 2.9 MHz 0x1D = 2.95 MHz 0x1E = 3.0 MHz 0x1F = 3.05 MHz	0x5
31:10	-	Reserved	-

6.5.45 RTC oscillator control register

This register enables the 32 kHz output of the RTC oscillator (32k_clk). This clock can be used to create the main clock when the PLL input or output is selected as the clock source to the main clock.

Table 144. RTC oscillator control register (RTCOSCCTRL, main syscon: offset 0x50C) bit description

Bit	Symbol	Value	Description	Reset value
0	EN		RTC 32 kHz clock enable.	0x1
		0	Disabled. RTC clock off.	
		1	Enabled. RTC clock on.	
31:1	-	-	Reserved	0x0

6.5.46 PLL registers

The PLL provides a wide range of frequencies and can potentially be used for many on-chip functions. the PLL can be used with or without a spread spectrum clock generator. See Section 6.6.5 “System PLL functional description” for additional details of PLL operation.

6.5.46.1 System PLL control register

The SYSPLLCTRL register provides most of the control over basic selections of PLL modes and operating details.

Table 145. System PLL control register (SYSPLLCTRL, main syscon: offset 0x580 bit description

Bit	Symbol	Value	Description	Reset value
3:0	SELR	-	Bandwidth select R value. See Section 6.5.46.6 regarding values for this field.	0x0
9:4	SELI	-	Bandwidth select I value. See Section 6.5.46.6 regarding values for this field.	0x0
14:10	SELP	-	Bandwidth select P value. See Section 6.5.46.6 regarding values for this field.	0x0
15	BYPASS		PLL bypass control.	0x0
		0	Bypass disabled. PLL CCO is sent to the PLL post-dividers.	
		1	Bypass enabled. PLL input clock is sent directly to the PLL output (default).	
16	BYPASSCCODIV2		Bypass feedback clock divide by 2.	0x0
		0	Divide by 2. The CCO feedback clock is divided by 2 in addition to the programmed M divide.	
		1	Bypass. The CCO feedback clock is divided only by the programmed M divide.	
17	UPLIMOFF		Disable upper frequency limiter. For spread spectrum mode: SEL_EXT = 0, BANDSEL = 0, and UPLIMOFF = 1.	0x0
		0	Normal mode.	
		1	Upper frequency limiter disabled.	

Table 145. System PLL control register (SYSPLLCTRL, main syscon: offset 0x580 bit description *…continued*

Bit	Symbol	Value	Description	Reset value
18	BANDSEL		PLL filter control. Set this bit to one when the spread spectrum controller is disabled or at low frequencies. For spread spectrum mode: SEL_EXT = 0, BANDSEL = 0, and UPLIMOFF = 1.	0x0
		0	SSCG control. The PLL filter uses the parameters derived from the spread spectrum controller.	
		1	MDEC control. The PLL filter uses the programmable fields SELP, SELR, and SELI in this register to control the filter constants.	
19	DIRECTI		PLL direct input enable.	0x0
		0	Disabled. The PLL input divider (N divider) output is used to drive the PLL CCO.	
		1	Enabled. The PLL input divider (N divider) is bypassed. the PLL input clock is used directly to drive the PLL CCO input.	
20	DIRECTO		PLL direct output enable.	0x0
		0	Disabled. The PLL output divider (P divider) is used to create the PLL output.	
		1	Enabled. The PLL output divider (P divider) is bypassed, the PLL CCO output is used as the PLL output.	
31:21	-	-	Reserved. Read value is undefined, only zero should be written.	-

6.5.46.2 System PLL status register

The read-only SYSPLLSTAT register provides the PLL lock status

Remark: The lock status does not reliably indicate the PLL status for the following two configurations: spread-spectrum mode or fractional enabled or low input clock frequencies such as 32 kHz. In these cases, refer to the PLL lock times listed in the specific device data sheet to obtain appropriate wait times for the PLL to lock.

Table 146. System PLL status register (SYSPLLSTAT, main syscon: offset 0x584) bit description

Bit	Symbol	Description	Reset value
0	LOCK	PLL lock indicator.	0x0
31:1	-	Reserved.	-

6.5.46.3 System PLL N-divider register

The SYSPLLNDEC controls operation of the PLL pre-divider.

Table 147. System PLL N-divider register (SYSPLLNDEC, main syscon: offset 0x588) bit description

Bit	Symbol	Description	Reset value
9:0	NDEC	Decoded N-divider coefficient value. This field is encoded, see Section 6.5.46.6 for information values for this field.	0x0
10	NREQ	NDEC reload request. When a 1 is written to this bit, the NDEC value is loaded into the PLL. Must be cleared by software for any subsequent load, or the PLL can be powered down and back up via the PDEN_SYS_PLL bit in the PDRUNCFG0 register if the NDEC value is changed.	0x0
31:11	-	Reserved. Read value is undefined, only zero should be written.	-

6.5.46.4 System PLL P-divider register

The SYSPLLPDEC controls operation of the PLL post-divider.

Table 148. System PLL P-divider register (SYSPLLPDEC, main syscon: offset 0x58C) bit description

Bit	Symbol	Description	Reset value
6:0	PDEC	Decoded P-divider coefficient value. This field is encoded, see Section 6.5.46.6 for information values for this field.	0x0
7	PREQ	PDEC reload request. When a 1 is written to this bit, the PDEC value is loaded into the PLL. Must be cleared by software for any subsequent load, or the PLL can be powered down and back up via the PDEN_SYS_PLL bit in the PDRUNCFG0 register if the PDEC value is changed.	0x0
31:8	-	Reserved. Read value is undefined, only zero should be written.	-

6.5.46.5 Spread spectrum control with the System PLL

The spread spectrum functionality can be used to modulate the PLL output frequency. This can decrease electromagnetic interference (EMI) in an application.

The Spread Spectrum Clock Generator can be used in several ways:

- It can encode M-divider values between 1 and 255 to produce the MDEC value used directly by the PLL, saving the need for executing encoding algorithm code, or hard-coding predetermined values into an application.
- It can provide a fractional rate feature to the PLL.
- It can be set up to automatically alter the PLL CCO frequency on an ongoing basis to decrease electromagnetic interference (EMI).

If the spread spectrum mode is enabled, choose N to ensure 2 MHz < Fin/N < 4 MHz. Spread spectrum mode cannot be used when Fin = 32 kHz.

When the modulation (MR) is set to zero, the PLL becomes a fractional PLL.

6.5.46.5.1 System PLL spread spectrum control register 0

Table 149. System PLL spread spectrum control register 0 (SYSPLLSSCTRL0, main syscon: offset 0x590) bit description

Bit	Symbol	Value	Description	Reset value
16:0	MDEC	-	Decoded M-divider coefficient value. This field is encoded, see Section 6.5.46.6 for information values for this field.	0x0
17	MREQ	-	MDEC reload request. When a 1 is written to this bit, the MDEC value is loaded into the PLL. Must be cleared by software for any subsequent load, or the PLL can be powered down and back up via the PDEN_SYS_PLL bit in the PDRUNCFG0 register if the MDEC value is changed.	0x0
18	SEL_EXT		Select spread spectrum mode. Selects the source of the feedback divider value. For normal mode, this must be the value from the MDEC field in this register. For spread spectrum mode: SEL_EXT = 0, BANDSEL = 0, and UPLIMOFF = 1.	0x1
		0	The PLL feedback divider value comes from the spread spectrum controller.	
		1	The PLL feedback divider value comes from the MDEC field in this register.	
31:19	-	-	Reserved. Read value is undefined, only zero should be written.	-

6.5.46.5.2 System PLL spread spectrum control register 1

Table 150. System PLL spread spectrum control register 1 (SYSPLLSSCTRL1, main syscon: offset 0x594) bit description

Bit	Symbol	Value	Description	Reset value
18:0	MD	-	M- divider value with fraction. MD[18:11]: integer portion of the feedback divider value. MD[10:0]: fractional portion of the feedback divider value. In fractional mode, fcco = (2 - BYPASSCCODIV2) x (MD x 2^-11) x Fref	0x0
19	MDREQ	-	MD reload request. When a 1 is written to this bit, the MD value is loaded into the PLL. This bit is cleared when the load is complete.	0x0
22:20	MF	-	Programmable modulation frequency fm = Fref/Nss with Fref = Fin/N 0b000 => Nss = 512 (fm ≈ 3.9 - 7.8 kHz) 0b001 => Nss ≈ 384 (fm ≈ 5.2 - 10.4 kHz) 0b010 => Nss = 256 (fm ≈ 7.8 - 15.6 kHz) 0b011 => Nss = 128 (fm ≈ 15.6 - 31.3 kHz) 0b100 => Nss = 64 (fm ≈ 32.3 - 64.5 kHz) 0b101 => Nss = 32 (fm ≈ 62.5- 125 kHz) 0b110 => Nss ≈ 24 (fm ≈ 83.3- 166.6 kHz) 0b111 => Nss = 16 (fm ≈ 125- 250 kHz)	0x0
25:23	MR	-	Programmable frequency modulation depth. 0 = no spread. δfmodpk-pk = Fref x k/Fcco = k/MDdec 0b000 => k = 0 (no spread spectrum) 0b001 => k ≈ 1 0b010 => k ≈ 1.5 0b011 => k ≈ 2 0b100 => k ≈ 3 0b101 => k ≈ 4 0b110 => k ≈ 6 0b111 => k ≈ 8	0x0
27:26	MC	-	Modulation waveform control. 0 = no compensation. Compensation for low pass filtering of the PLL to get a triangular modulation at the output of the PLL, giving a flat frequency spectrum. 0b00 => no compensation 0b10 => recommended setting 0b11 => max. compensation	0x0
28	PD		Spread spectrum power-down.	0x1
		0	Enabled. Spread spectrum controller is enabled.	
		1	Disabled. Spread spectrum controller is disabled.	
29	DITHER		Select modulation frequency.	0x0
		0	Fixed. Fixed modulation frequency.	
		1	Dither. Randomly dither between two modulation frequencies.	
31:30	-	-	Reserved. Read value is undefined, only zero should be written.	-

6.5.46.6 Calculating NDEC, MDEC, PDEC, SELP, SELI, and SELR values

NDEC, MDEC, and PDEC do not use the direct binary representations of the desired settings directly. Instead, an encoded version of the values is used. SELP, SELI, and SELR values are determined by the MDEC.

6.5.46.6.1 NDEC

The valid range for N is 1 to 2^8. This value is encoded into a 10-bit NDEC value. The value is defined by the following code:

```
N_max=0x00000100, x=0x00000080;
     switch (N) {
          case 0: x = 0xFFFFFFFF;
          case 1: x = 0x00000302;
          case 2: x = 0x00000202;

          default: for (i = N; i <= N_max; i++)
               x = (((x ^ (x>>2) ^ (x>>3) ^ (x>>4)) & 1) << 7) | ((x>>1) & 0x7F); }
     NDEC[9:0] = x;
```

Remark: While the PLL output is in use, do not change the NDEC value. Changing the NDEC value changes the FCCO frequency and can cause the system to fail.

6.5.46.6.2 MDEC

The valid range for M is 1 to 2^15. This value is encoded into a 17-bit MDEC value. The PLL M-divider value (MDEC) can be set directly if the PLL is not used with the spread spectrum clock generator (SSCG). If the SSCG is enabled via the SEL_EXT bit, then the SSCG sets the MDEC value.

The value between M and MDEC is defined by the following code.

```
M_max=0x00008000, x=0x00004000;
     switch (M) {
          case 0: x = 0xFFFFFFFF;
          case 1: x = 0x00018003;
          case 2: x = 0x00010003;

          default: for (i = M; i <= M_max; i++)
               x = (((x ^ (x>>1)) & 1) << 14) | ((x>>1) & 0x3FFF); }
     MDEC[16:0] = x;
```

Remark: While the PLL output is in use, do not change the MDEC value. Changing the MDEC value changes the FCCO frequency and can cause the system to fail.

6.5.46.6.3 PDEC

The valid range for P is from 1 to 2^5. This value is encoded into a 7-bit PDEC value. The value is defined by following code:

```
P_max=0x20, x=0x10;
     switch (P) {
          case 0: x = 0xFFFFFFFF;
          case 1: x = 0x00000062;
          case 2: x = 0x00000042;
          default: for (i = P; i <= P_max; i++)
                x = (((x ^ (x>>2)) & 1) << 4) | ((x>>1) & 0xF); }
     PDEC[6:0] = x;
```

Remark: While the PLL output is in use, do not change the PDEC value. Changing the PDEC value changes the PLL output frequency and can cause the system to fail.

6.5.46.6.4 SELP, SELI, and SELR

The values for SELP, SELI, and SELR depend on the value for M as expressed by the following pseudo-code:

```
if (BYPASSCCODIV2)
     M = M / 2;
     // bandwidth: compute SELP from Multiplier
     SELP = ((550 * M) / (150 * 2)) + 1;
     if (SELP > 0x1F)
          SELP = 0x1F;
     // bandwidth: compute SELI from Multiplier
     if (M > (16384 * 150 / 550))
          SELI = 1;
     else if (M > (8192 * 150 / 550))
          SELI = 2;
     else if (M > (2048 * 150 / 550))
          SELI = 4;
     else if (M >= (501 * 150 / 550))
          SELI = 8;
     else if (M >= (60 * 150 / 550))
     {
          SELI = 4 * (1024 / (((550 * M) / 150) + 9) + 1);
     }
     else
     {
          SELI = 4 * (((550 * M) / (150 * 4)) + 1);
     }
     if (SELI > 0x3F)
          SELI = 0x3F;
     SELR = 0
```

Remark: If the 32 kHz RTC oscillator is used as the reference input to the PLL, then use fixed values SELI=1, SELP=6 and SELR=0, instead of applying the above rules. These values reduce the PLL loop bandwidth to combat the effect of reference oscillator jitter on the PLL output signal.

Remark: The values for SELP, SELI, and SELR are generated by the encoding block when the spread spectrum clock generator is enabled and need not be programmed explicitly.

6.5.47 Sleep configuration register 0

The PDSLEEPCFG0 register controls the power to various analog blocks while the CPU is in the deep sleep reduced power mode. Entering reduced power modes is typically accomplished by calling the power mode entry API. See Section 8.4.3. It is also possible to configure the PDSLEEPCFG0 and PDSLEEPCFG1 directly, set the SLEEPDEEP bit of the Cortex-M0+ SCR register, then execute a WFI instruction to enter reduced power modes.

Table 151. Sleep configuration register (PDSLEEPCFG0, main syscon: offset 0x600) bit description

Bit	Symbol	Description	Reset value
31:0	-	See bit descriptions in the PDRUNCFG0 register.	0x02F8 0540

6.5.48 Power configuration register 0

The PDRUNCFG0 register controls the power to various analog blocks.

Remark: For safety, changes to this register should generally be accomplished by writing to PDRUNCFGSET0 and/or PDRUNCFGCLR0. This avoids the possibility of an interrupt changing the value of PDRUNCFG0 after it is read, but before an altered value is written back. It also avoids accidentally changing bits that may have been altered by an API or another portion of user software. **Reserved bits must not be changed by user software.**

Table 152. Power Configuration register (PDRUNCFG0, main syscon: offset 0x610) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved	-
4	PDEN_FRO	FRO oscillator. 0 = Powered; 1 = Powered down.	0x0
5	-	Reserved	-
6	PDEN_TS	Temp sensor. 0 = Powered; 1 = Powered down.	0x1
7	PDEN_BOD_RST	Brown-out Detect reset. 0 = Powered; 1 = Powered down.	0x0
8	PDEN_BOD_INTR	Brown-out Detect interrupt. 0 = Powered; 1 = Powered down.	0x1
9	-	Reserved	-
10	PDEN_ADC0	ADC0. 0 = Powered; 1 = Powered down.	0x1
12:11	-	Reserved	-
13	PDEN_SRAM0	SRAM0. 0 = Powered; 1 = Powered down.	0x0
15:14	-	Reserved	-
16	PDEN_SRAMX	SRAMX. 0 = Powered; 1 = Powered down.	0x0
17	PDEN_ROM	ROM. 0 = Powered; 1 = Powered down.	0x0
18	-	Reserved	-
19	PDEN_VDDA	VDDA to the ADC, must be enabled for the ADC to work. Also see bit 23. 0 = Powered; 1 = Powered down.	0x1
20	PDEN_WDT_OSC	Watchdog oscillator. 0 = Powered; 1 = Powered down.	0x1
21	PDEN_USB_PHY	USB pin interface. 0 = Powered; 1 = Powered down.	0x1
22	PDEN_SYS_PLL	System PLL. 0 = Powered; 1 = Powered down.	0x1
23	PDEN_VREFP	Vrefp to the ADC, must be enabled for the ADC to work. Also see bit 19. 0 = Powered; 1 = Powered down.	0x1
31:24	-	Reserved	-

6.5.49 Power configuration set register 0

Writing a 1 to a bit position in PDRUNCFGSET0 sets the corresponding position in PDRUNCFG0. This is a write-only register. For bit assignments, see Table 152.

Table 153. Power configuration set registers (PDRUNCFGSET0 main syscon: offset 0x620) bit description

Bit	Symbol	Description	Reset value
31:0	PD_SET	Writing ones to this register sets the corresponding bit or bits in the PDRUNCFG0 register, if they are implemented. Bits that do not correspond to defined bits in PDRUNCFG0 are reserved and only zeroes should be written to them.	-

6.5.50 Power configuration clear register 0

Writing a 1 to a bit position in PDRUNCFGCLR0 clears the corresponding position in PDRUNCFG0. This is a write-only register. For bit assignments, see Table 152.

Table 154. Power configuration clear registers (PDRUNCFGCLR0, main syscon: offset 0x630) bit description

Bit	Symbol	Description	Reset value
31:0	PD_CLR	Writing ones to this register clears the corresponding bit or bits in the PDRUNCFG0 register, if they are implemented. Bits that do not correspond to defined bits in PDRUNCFG0 are reserved and only zeroes should be written to them.	-

6.5.51 Start enable register 0

The STARTER0 register enables an interrupt for wake-up from deep-sleep mode.

GPIO Pin interrupts, GPIO group interrupts, and selected peripherals such as USB, SPI, I2C, USART, WWDT, RTC, Micro-tick Timer, and BOD can be left running to allow wake-up from deep sleep mode.

The pattern match feature of the pin interrupt requires a clock in order to operate, and will not wake up the device from reduced power modes beyond sleep mode.

Remark: It is recommended that changes to the STARTER registers be accomplished by using the related STARTERSET and STARTERCLR registers. This avoids any unintentional setting or clearing of other bits.

Remark: Also enable the corresponding interrupts in the NVIC. See Table 82 "Interrupt Set-Enable Register 0 register".

Table 155. Start enable register 0 (STARTER0, main syscon: offset 0x680) bit description

Bit	Symbol	Description	Reset value
0	WDT, BOD	WWDT and BOD interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
1	DMA	DMA wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
2	GINT0	Group interrupt 0 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	-
3	GINT1	Group interrupt 1 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
4	PIN_INT0	GPIO pin interrupt 0 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Not for pattern match.	0x0
5	PIN_INT1	GPIO pin interrupt 1 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Not for pattern match.	0x0

Table 155. Start enable register 0 (STARTER0, main syscon: offset 0x680) bit description *…continued*

Bit	Symbol	Description	Reset value
6	PIN_INT2	GPIO pin interrupt 2 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Not for pattern match.	0x0
7	PIN_INT3	GPIO pin interrupt 3 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Not for pattern match.	0x0
8	UTICK	Micro-tick Timer wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
13:9	-	Reserved. Read value is undefined, only zero should be written.	-
14	FLEXCOMM0	Flexcomm 0 peripheral interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
15	FLEXCOMM1	Flexcomm 1 peripheral interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
16	FLEXCOMM2	Flexcomm 2 peripheral interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
17	FLEXCOMM3	Flexcomm 3 peripheral interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
18	FLEXCOMM4	Flexcomm 4 peripheral interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
19	FLEXCOMM5	Flexcomm 5 peripheral interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
20	FLEXCOMM6	Flexcomm 6 peripheral interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
21	FLEXCOMM7	Flexcomm 7 peripheral interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
26:22	-	Reserved. Read value is undefined, only zero should be written.	-
27	USB_NEEDCLK	USB activity interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
28	USB	USB function interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
29	RTC	RTC interrupt alarm and wake-up timer. 0 = Wake-up disabled. 1 = Wake-up enabled.	0x0
31:30	-	Reserved. Read value is undefined, only zero should be written.	-

6.5.52 Start enable set register 0

Writing a 1 to a bit position in STARTERSET0 sets the corresponding position in STARTER0. This is a write-only register. For bit assignments, see Table 155.

Table 156. Start enable set register 0 (STARTERSET0, main syscon: offset 0x6A0) bit description

Bit	Symbol	Description	Reset value
31:0	START_SET0	Writing ones to this register sets the corresponding bit or bits in the STARTER0 register, if they are implemented. Bits that do not correspond to defined bits in STARTER0 are reserved and only zeroes should be written to them.	-

6.5.53 Start enable clear register 0

Writing a 1 to a bit position in STARTERCLR0 clears the corresponding position in STARTER0. This is a write-only register. For bit assignments, see Table 155.

Table 157. Start enable clear register 0 (STARTERCLR0, main syscon: offset 0x6C0) bit description

Bit	Symbol	Description	Reset value
31:0	START_CLR0	Writing ones to this register clears the corresponding bit or bits in the STARTER0 register, if they are implemented. Bits that do not correspond to defined bits in STARTER0 are reserved and only zeroes should be written to them.	-

6.5.54 Hardware Wake-up control register

The purpose of the Hardware Wake-up control register is to provide the possibility for some peripherals to have DMA service during deep-sleep mode without waking up the CPU. These wake-ups are based on peripheral FIFO levels, not directly related to peripheral DMA requests and interrupts.

When a peripheral that is able to operate (at least in some modes) during deep-sleep reaches its programmed FIFO threshold, it can cause bus clocks to be temporarily enabled. During that time, DMA can move data out of the peripheral FIFO into memory. When DMA completes, full deep-sleep mode will be resumed.

Table 158. Hardware Wake-up control register (HWWAKE, main syscon: offset 0x780) bit description

Bit	Symbol	Description	Reset value
0	FORCEWAKE	Force peripheral clocking to stay on during deep-sleep mode. When 1, clocking to peripherals is prevented from being shut down when the CPU enters deep-sleep mode. This is intended to allow a coprocessor to continue operating while the main CPU(s) are shut down.	0x0
1	FCWAKE	Wake for Flexcomm Interfaces. When 1, any Flexcomm Interface FIFO reaching the level specified by its own FIFO level will cause peripheral clocking to be enabled temporarily while a Flexcomm Interface FIFO level is at or beyond the specified level. This allows DMA to become active to move data out of the Flexcomm Interface FIFO. Generally, WAKEDMA should also be enabled when FCWAKE is enabled. This feature uses the TXLVL flag in the FIFOSTAT register if the bit WAKETX in the FIFOCFG register is set, or the RXLVL flag in FIFOSTAT if WAKERX in FIFOCFG is set.	0x0
2	-	Reserved. Read value is undefined, only zero should be written.	-
3	WAKEDMA	Wake for DMA. When 1, DMA being busy will cause peripheral clocking to remain running until DMA completes. This is generally used in conjunction with bit 1 and/or 2 in order to prevent peripheral clocking from being shut down as soon as the cause of wake-up is cleared, but before DMA has completed its related activity.	0x0
31:4	-	Reserved. Read value is undefined, only zero should be written.	-

6.5.55 JTAG ID code register

This register contains the JTAG ID code.

Table 159. JTAG ID code register (JTAGIDCODE, main syscon: offset 0xFF4) bit description

Bit	Symbol	Description	Value
31:0	JTAGID	JTAG ID code.	0x1725E02B

6.5.56 Device ID0 register

This register contains the part ID. The part ID can also be obtained using the ISP or IAP ReadPartID commands. See Table 20 and Table 33.

Table 160. Device ID0 register (DEVICE_ID0, main syscon: offset 0xFF8) bit description

Bit	Symbol	Description	Reset value
31:0	PARTID	Part ID	part dependent

Table 161. Device ID0 register values

Part number	Part ID
LPC51U68JBD64	0x06451B68
LPC51U68JBD48	0x06451A68

6.5.57 Device ID1 register

This register contains the boot ROM and die revisions.

Table 162. Device ID1 register (DEVICE_ID1, main syscon: offset 0xFFC) bit description

Bit	Symbol	Description	Value
31:0	REVID	Revision	0x0041725E

6.5.58 Asynchronous peripheral reset control register

The ASYNCPRESETCTRL register allows software to reset specific peripherals attached to the async APB bridge. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset.

Remark: It is recommended that changes to the ASYNCPRESETCTRL registers be accomplished by using the related ASYNCPRESETCTRLSET and ASYNCPRESETCTRLCLR registers. This avoids any unintentional setting or clearing of other bits.

Table 163. Asynchronous peripheral reset control register (ASYNCPRESETCTRL, async syscon: offset 0x000) bit description

Bit	Symbol	Description	Reset value
12:0	-	Reserved	-
13	CTIMER3	Standard counter/timer CTIMER3 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function.	0x0
31:14	-	Reserved	-

6.5.59 Asynchronous peripheral reset control set register

Writing a 1 to a bit position in ASYNCPRESETCTRLSET sets the corresponding position in ASYNCPRESETCTRL. This is a write-only register. For bit assignments, see Table 163.

Table 164. Asynchronous peripheral reset control set register (ASYNCPRESETCTRLSET, async syscon: offset 0x004) bit description

Bit	Symbol	Description	Reset value
31:0	ARST_SET	Writing ones to this register sets the corresponding bit or bits in the ASYNCPRESETCTRL register, if they are implemented. Bits that do not correspond to defined bits in ASYNCPRESETCTRL are reserved and only zeroes should be written to them.	-

6.5.60 Asynchronous peripheral reset control clear register

Writing a 1 to a bit position in ASYNCPRESETCTRLCLR clears the corresponding position in PRESETCTRL0. This is a write-only register. For bit assignments, see Table 163.

Table 165. Asynchronous peripheral reset control clear register (ASYNCPRESETCTRLCLR, async syscon: offset 0x008) bit description

Bit	Symbol	Description	Reset value
31:0	ARST_CLR	Writing ones to this register clears the corresponding bit or bits in the ASYNCPRESETCTRL register, if they are implemented. Bits that do not correspond to defined bits in ASYNCPRESETCTRL are reserved and only zeroes should be written to them.	-

6.5.61 Asynchronous APB clock control register

This register controls how the clock selected for the asynchronous APB peripherals is divided to provide the clock to the asynchronous peripherals. The clock will be stopped if the DIV field is set to zero.

Remark: It is recommended that changes to the ASYNCAPBCLKCTRL registers be accomplished by using the related ASYNCAPBCLKCTRLSET and ASYNCAPBCLKCTRLCLR registers. This avoids any unintentional setting or clearing of other bits.

Table 166. Asynchronous APB clock control register (ASYNCAPBCLKCTRL, async syscon: offset 0x010) bit description

Bit	Symbol	Description	Reset value
12:0	-	Reserved	-
13	CTIMER3	Controls the clock for CTIMER3. 0 = Disable; 1 = Enable.	0x0
31:14	-	Reserved	-

6.5.62 Asynchronous APB clock control set register

Writing a 1 to a bit position in ASYNCAPBCLKCTRLSET sets the corresponding position in ASYNCAPBCLKCTRL. This is a write-only register. For bit assignments, see Table 163.

Table 167. Asynchronous APB clock control set register (ASYNCAPBCLKCTRLSET, async syscon: offset 0x014) bit description

Bit	Symbol	Description	Reset value
31:0	ACLK_SET	Writing ones to this register sets the corresponding bit or bits in the ASYNCAPBCLKCTRL register, if they are implemented. Bits that do not correspond to defined bits in ASYNCPRESETCTRL are reserved and only zeroes should be written to them.	-

6.5.63 Asynchronous APB clock control clear register

Writing a 1 to a bit position in ASYNCAPBCLKCTRLCLR clears the corresponding position in ASYNCAPBCLKCTRL. This is a write-only register. For bit assignments, see Table 163.

Table 168. Asynchronous APB clock control clear register (ASYNCAPBCLKCTRLCLR, async syscon: offset 0x018) bit description

Bit	Symbol	Description	Reset value
31:0	ACLK_CLR	Writing ones to this register clears the corresponding bit or bits in the ASYNCAPBCLKCTRL register, if they are implemented. Bits that do not correspond to defined bits in ASYNCAPBCLKCTRL are reserved and only zeroes should be written to them.	-

6.5.64 Asynchronous clock source select register A

This register selects a potential clock for the asynchronous APB peripherals from among several clock sources.

Remark: This selection is internally synchronized: the clock being switched from and the clock being switched to must both be running and have occurred in specific states before the selection actually changes.

Table 169. Asynchronous clock source select register A (ASYNCAPBCLKSELA, async syscon: offset 0x020) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SEL		Clock source for asynchronous clock source selector A	0x0
		0x0	Main clock	
		0x1	FRO 12 MHz	
		0x2	Reserved setting	
		0x3	Reserved setting	
31:2	-	-	Reserved	-

6.5.65 BOD control register

The BOD control register selects four separate threshold values for sending a BOD interrupt to the NVIC and for forced reset. Reset and interrupt threshold values listed in Table 170 are typical values. More details can be found in specific device data sheets.

Both the BOD interrupt and the BOD reset can wake-up the chip from sleep and deep-sleep, modes if enabled. See Chapter 8 “LPC51U68 Power profiles/Power control API”.

Table 170. BOD control register (BODCTRL, other system registers: offset 0x044) bit description

Bit	Symbol	Value	Description	Reset value
1:0	BODRSTLEV		BOD reset level	0x0
		0x0	Level 0: 1.5 V	
		0x1	Level 1: 1.85 V	
		0x2	Level 2: 2.0 V	
		0x3	Level 3: 2.3 V	
2	BODRSTENA		BOD reset enable	0x0
		0	Disable reset function.	
		1	Enable reset function.	
4:3	BODINTLEV		BOD interrupt level	0x0
		0x0	Level 0: 2.05 V	
		0x1	Level 1: 2.45 V	
		0x2	Level 2: 2.75 V	
		0x3	Level 3: 3.05 V	
5	BODINTENA		BOD interrupt enable	0x0
		0	Disable interrupt function.	
		1	Enable interrupt function.	
6	BODRSTSTAT		BOD reset status. When 1, a BOD reset has occurred. Cleared by writing 1 to this bit.	0x0
7	BODINTSTAT		BOD interrupt status. When 1, a BOD interrupt has occurred. Cleared by writing 1 to this bit.	0x0
31:8	-	-	Reserved	-

6.6 Functional description

6.6.1 Reset

Reset has the following sources:

- The $\overline{\text{RESET}}$ pin.
- Watchdog reset.
- Power-On Reset (POR).
- Brown Out Detect (BOD).
- ARM software reset.
- ISP-AP debug reset.

Assertion of the POR or the BOD reset, once the operating voltage attains a usable level, starts the FRO. After the FRO-start-up time (maximum of 6 μs on power-up), the FRO provides a stable clock output. The reset remains asserted until the external Reset is released, the oscillator is running, and the flash controller has completed its initialization.

On the assertion of any reset source (ARM software reset, POR, BOD reset, External reset, and Watchdog reset), the following processes are initiated:

1. The FRO is enabled or starts up if not running.
2. The flash wake-up starts. This takes approximately 250 μs or less.
3. The boot code in the ROM starts. The boot code performs the boot tasks and may jump to the flash.

When the internal Reset is removed, the processor begins executing at address 0, which is initially the Reset vector mapped from the boot block. At that point, all of the processor and peripheral registers have been initialized to predetermined values.

6.6.2 Start-up behavior

The FRO 12 MHz oscillator provides the default clock at Reset and provides a clean system clock shortly after the supply pins reach operating voltage. See the device data sheet for details of start-up timing.

6.6.3 Brown-out detection

The part includes up to four levels for monitoring the voltage on the V_{DD} pin. If this voltage falls below one of the selected levels, the BOD asserts an interrupt signal to the NVIC or issues a reset, depending on the value of the BODRSTENA bit in the BOD control register (Table 170).

The interrupt signal can be enabled for interrupt in the Interrupt Enable Register in the NVIC (see Table 82) in order to cause a CPU interrupt; if not, software can monitor the signal by reading a dedicated status register.

If the BOD interrupt is enabled in the STARTER0 register and in the NVIC, the BOD interrupt can wake up the chip from reduced power modes, not including deep power-down.

If the BOD reset is enabled, the forced BOD reset can wake up the chip from reduced power modes, not including deep power-down.

6.6.4 Flash accelerator functional description

The flash accelerator block allows maximization of the performance of the CPU when it is running code from flash memory, while also saving power. The flash accelerator also provides speed and power improvements for data accesses to the flash memory.

A description of the flash accelerator configuration register may be found in Section 6.5.38.

The flash accelerator is divided into several functional blocks:

- AHB matrix interface, accessible by all bus masters that have a connection to the matrix slave port used for flash memory.
- An array of eight 128-bit buffers
- Flash accelerator control logic, including address compare and flash control
- A flash memory interface

Figure 10 shows a simplified diagram of the flash accelerator blocks and data paths.

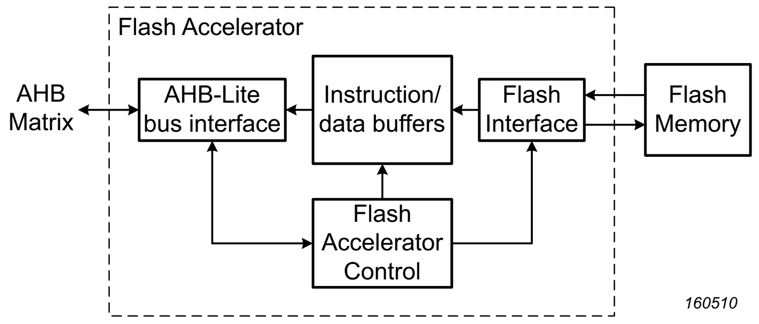


Fig 10. Simplified block diagram of the flash accelerator

In the following descriptions, the term “fetch” applies to an explicit flash read request from the CPU. “Prefetch” is used to denote a flash read of instructions beyond the current processor fetch address.

6.6.4.1 Flash memory bank

Flash programming operations are not controlled by the flash accelerator, but are handled as a separate function. The boot code includes flash programming functions that may be called as part of the application program, as well as loaders that may be used to accomplish initial flash programming.

6.6.4.2 Flash programming constraints

Since the flash memory does not allow accesses during programming and erase operations, it is necessary for the flash accelerator to force the CPU to wait if a memory access to a flash address is requested while the flash memory is busy with a programming operation. Under some conditions, this delay could result in a Watchdog time-out. The user will need to be aware of this possibility and take steps to insure that an unwanted Watchdog reset does not cause a system failure while programming or erasing the flash memory. Application code, especially interrupts, can continue to run from other memories during flash erase/write operations.

In order to preclude the possibility of stale data being read from the flash memory, the flash accelerator buffers are automatically invalidated at the beginning of any flash programming or erase operation. Any subsequent read from a flash address will cause a new fetch to be initiated after the flash operation has completed.

6.6.5 System PLL functional description

The PLL is typically used to create a frequency that is higher than other on-chip clock sources, and used to operate the CPU and/or other on-chip functions. It may also be used to obtain a specific clock that is otherwise not available. For example, a source clock with a frequency of any integer MHz (e.g. the 12 MHz FRO) can be divided down to 1 MHz, then multiplied up to any other integer MHz (e.g. 13, 14, 15, etc.).The PLL can be set up by calling an API supplied by NXP Semiconductors. Also see Section 6.5.46 “PLL registers”, and Section 8.4.1.

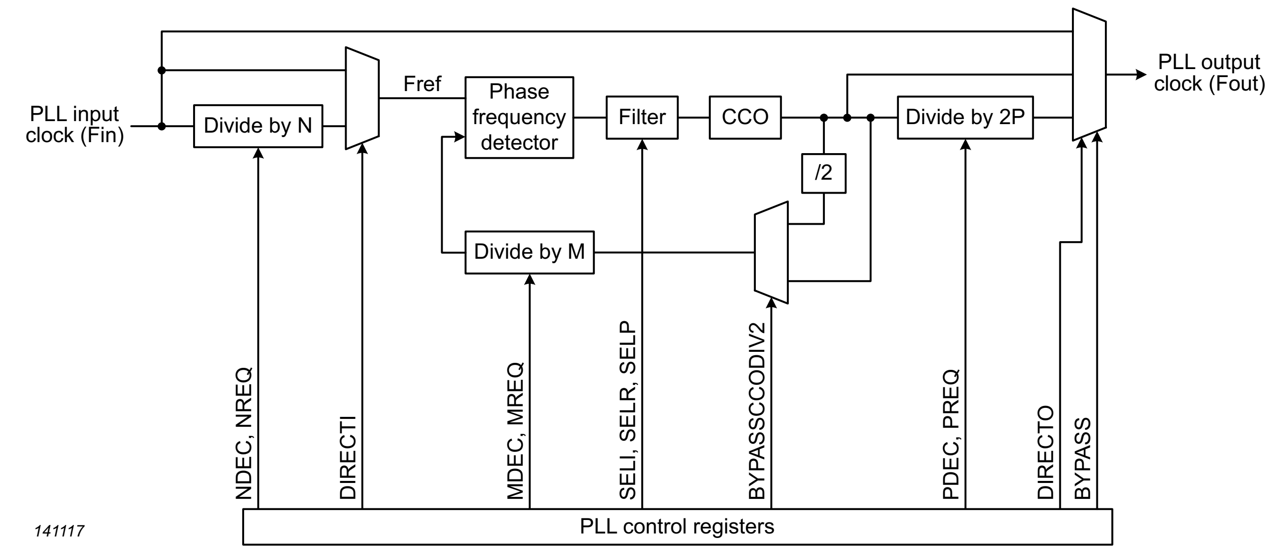


Fig 11. System PLL block diagram showing typical operation

6.6.5.1 PLL Features

- Input frequency: in normal mode, can include the 32 kHz RTC clock and 12 MHz FRO, or up to 25 MHz from the CLKIN pin. In fractional mode, Fref (see Figure 12) between 2 and 4 MHz.
- CCO frequency: 75 MHz to 150 MHz.
- Output clock range: 1.2 MHz to 150 MHz. Note that the upper frequency limit of the PLL exceeds the upper frequency limit of this device.
- Programmable dividers:
  - Pre-divider. Divide by N, where N = 1 to 256
  - Feedback-divider. Divide by M or 2 x M (where M = 1 to 32,768)
  - Post-divider. Divide by 1 or 2 x P, where P = 1 to 32

- Lock detector.
- Power-down mode.
- Fractional divider mode.
- Spread Spectrum mode.

6.6.5.2 PLL description

A number of sources may be used as an input to the PLL, see Figure 9. In addition, a block diagram of the PLL is shown in Figure 11. The PLL input, in the range of 32 kHz to 25 MHz, may initially be divided down by a value "N", which may be in the range of 1 to 256. This input division provides a greater number of possibilities in providing a wide range of output frequencies from the same input frequency.

Following the PLL input divider is the PLL multiplier. The multiplier can multiply the input divider output through the use of a Current Controlled Oscillator (CCO) by a value "M", in the range of 1 through 32,768. The resulting frequency must be in the range of 75 MHz to 150 MHz. The multiplier works by dividing the CCO output by the value of M, then using a phase-frequency detector to compare the divided CCO output to the multiplier input. The error value is filtered and used to adjust the CCO frequency.

The PLL output may further be divided by a value "2P" if desired, where P is value in the range of 1 to 32.

All of the dividers that are part of the PLL use an encoded value, not the binary divide value. The LPCOpen Chip_POWER_SetPLL API (see Section 8.4.1) can adjust the value for the main feedback divider (the M divider), but does not accept pre- and post-divider values. See section Section 6.6.5.3 and Section 6.6.5.5 for information on how to obtain divider values.

There are additional dividers in the clocking system to bring the PLL output frequency down to what is needed for the CPU, USB, and other peripherals. The PLL output dividers are described in the Clock Dividers section following the PLL description.

For PLL register descriptions, see Section 6.5.46.

6.6.5.2.1 Lock detector

The lock detector measures the phase difference between the rising edges of the input and feedback clocks. Only when this difference is smaller than the so called “lock criterion” for more than seven consecutive input clock periods, the lock output switches from low to high. A single too large phase difference immediately resets the counter and causes the lock signal to drop (if it was high). Requiring seven phase measurements in a row to be below a certain figure ensures that the lock detector will not indicate lock until both the phase and frequency of the input and feedback clocks are very well aligned. This effectively prevents false lock indications, and thus ensures a glitch free lock signal.

The PLL lock indicator is not dependable when Fref is below 100 kHz or above 20 MHz. Instead, software should use a 6 ms time interval to insure the PLL will be stable.

In fractional mode and spread spectrum mode, the PLL will generally not lock, software should use a 6 ms time interval to insure the PLL will be stable. See Section 6.6.5.5.1.

6.6.5.2.2 Power-down

To reduce the power consumption when the PLL clock is not needed, a PLL power-down mode has been incorporated. This mode is enabled by setting the PDEN_SYS_PLL bit to one in the power configuration register PDRUNCFG0 (Section 6.5.48). In this mode, the internal current reference will be turned off, the oscillator and the phase-frequency detector will be stopped and the dividers will enter a reset state. While in PLL power-down mode, the lock output will be low to indicate that the PLL is not in lock.

When the PLL power-down mode is terminated by setting the PDEN_SYS_PLL bit to zero, the PLL will resume its normal operation and will make the lock signal high once it has regained lock on the input clock. While in this state, new divider values may be entered, which will be used when the PLL power-down state is exited by clearing PDEN_SYS_PLL.

6.6.5.3 PLL operating modes

The PLL includes several main operating modes, and a power-down mode. These are summarized in Table 171 and detailed in the following sections.

Table 171. PLL operating mode summary

Mode	PDEN_SYS_PLL bit in PDRUNCFG0	Bits in SYSPLLCTRL:			SEL_EXT bit in SYSYPLLSSCTRL0	PD bit in SYSYPLLSSCTRL1
		BYPASS	UPLIMOFF	BANDSEL		
Normal	0	0	0	1	1	1
Fractional divider	0	0	1	0	0	0
Spread spectrum	0	0	1	0	0	0
Power-down	1	x [1]	x	x	x	1

[1] Use 1 if the PLL output is used even though the PLL is not altering the frequency.

6.6.5.3.1 Normal modes

Typical operation of the PLL includes an optional pre-divide of the PLL input, followed by a frequency multiplication, and finally an optional post-divide to produce the PLL output.

Notations used in the frequency equations:

- Fin = the input to the PLL.
- Fout = the output of the PLL.
- Fref = the PLL reference frequency, the input to the phase frequency detector.
- N = optional pre-divider value.
- M = feedback divider value, which represents the multiplier for the PLL. Note that an additional divide-by-2 may optionally be included in the divider path.
- P = optional post-divider value. An additional divide-by-2 is included in the post-divider path.

A block diagram of the PLL as used in normal modes is shown in Figure 12.

In all variations of normal mode, the following requirements must be met:

- 75 MHz ≤ Fcco ≤ 150 MHz
- 4 kHz ≤ Fin / N ≤ 25 MHz

Normal mode with optional pre-divide

In the equations, use N = 1 when the pre-divider is not used:

When the extra divide by 2 **is** in the feedback divider path (BYPASSCCODIV2 = 0):

Fout = Fcco = 2 x M x Fin / N

When the extra divide by 2 **is not** in the feedback divider path (BYPASSCCODIV2 = 1):

Fout = Fcco = M x Fin / N

Normal mode with post-divide and optional pre-divide

In the equations, use N = 1 when the pre-divider is not used:

When the extra divide by 2 **is** in the feedback divider path (BYPASSCCODIV2 = 0). Use N = 1 when the pre-divider is not used:

Fout = Fcco / (2 x P) = M x Fin / (N x P)

When the extra divide by 2 **is not** in the feedback divider path (BYPASSCCODIV2 = 1):

Fout = Fcco / (2 x P) = M x Fin / (N x 2 x P)

6.6.5.3.2 Fractional divider mode

The PLL includes an fractional divide mode. The fractional mode uses an integer divide value and that value plus 1 in a ratio determined by the fractional part of the divide value in order to obtain an average rate that is a fractional multiple of the PLL reference frequency. The SEL_EXT bit in the SYSPLLSSCTRL0 register determines whether the fractional divider is used (SEL_EXT = 0) or bypassed (SEL_EXT = 1). In the first case, the MD value from the SYSPLLSSCTRL1 register is used to generate the feedback divider values. In the latter case, the MDEC value from the SYSPLLSSCTRL0 register is used directly to control the feedback divider.

When the fractional divider is active, the spread spectrum controller block generates divider values M and M+1 in the correct proportion so that the average CCO frequency is represented by the specified fraction. the average CCO frequency is:

Fcco = (2 - BYPASSCCODIV2) * (MD * 2^{-11}) * Fref

The overall effect of the PLL otherwise the same as normal modes. A block diagram of the PLL as used in fractional mode is shown in Figure 12.

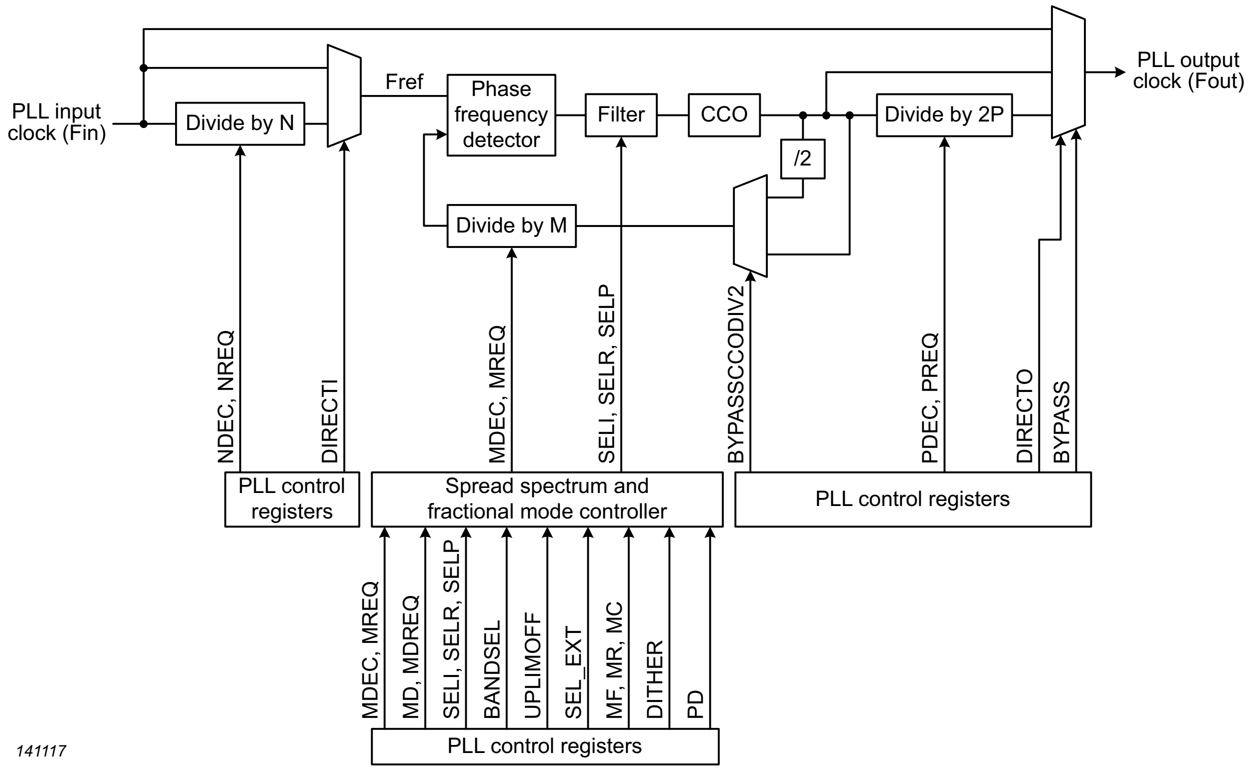


Fig 12. System PLL block diagram showing spread spectrum and fractional divide operation

6.6.5.3.3 Spread Spectrum mode

The spread spectrum functionality can be used to modulate the PLL output frequency automatically, in a programmable manner. This can decrease electromagnetic interference (EMI) in an application.

The Spread Spectrum Clock Generator can be used in several ways:

- It can encode M-divider values between 1 and 255 to produce the MDEC value used directly by the PLL, saving the need for executing encoding algorithm code, or hard-coding predetermined values into an application.
- It can provide a fractional rate feature to the PLL.
- It can be set up to automatically alter the PLL CCO frequency on an ongoing basis to decrease electromagnetic interference (EMI).

A block diagram of the PLL as used in fractional mode is shown in Figure 12.

If the spread spectrum mode is enabled, choose N to ensure 2 MHz < Fin/N < 4 MHz. Spread spectrum mode cannot be used when Fin = 32 kHz.

When the modulation (MR) is set to zero, the PLL becomes a fractional PLL.

Triangular wave modulation: For the center spread triangular waveform modulation with a modulation frequency depth δfmodpk-pk and a modulation frequency fm, the clock cycle displacement and spectral tone reduction ΔP can be calculated. The theoretical maximum clock cycle displacement (peak-to-peak) can be expressed with the following equation below:

if directo$_{PLL}$ = 1:

$$\Delta n_{max;theoretically} = \frac{N_{ss} \times k}{16}$$

if directo$_{PLL}$ = 0, P$_{PLL}$ = 1:

$$\Delta n_{max;theoretically} = \frac{N_{ss} \times k}{32 \times P_{PLL}}$$

In practice, the clock cycle displacement could be larger. So, for safety reasons (buffer overflow) use:

if directo$_{PLL}$ = 1:

$$\Delta n_{max;practically} = \frac{N_{ss} \times k}{8}$$

if directo$_{PLL}$ = 0, P$_{PLL}$ = 1:

$$\Delta n_{max;practically} = \frac{N_{ss} \times k}{16 \times P_{PLL}}$$

The spectral tone reduction/EMI reduction ΔP at F$_{out}$ is approximately:

if directo$_{PLL}$ = 1:

$$\Delta P \approx 10 log \frac{N_{ss} \times k}{2}$$

if directo$_{PLL}$ = 0, P$_{PLL}$ = 1:

$$\Delta P \approx 10 log \frac{N_{ss} \times k}{4 \times P_{PLL}}$$

See Table 172 for the spectral tone reduction and clock cycle displacement for directo$_{PLL}$ = 0 and P$_{PLL}$= 1.

Table 172. Values for different settings, directo$_{PLL}$ = 0, P$_{PLL}$ = 1

Table values are: ΔP Δn_{max}	mf[2:0]=000 N_{SS} = 512		mf[2:0]=001 N_{SS} ≈ 384		mf[2:0]=010 N_{SS} = 256		mf[2:0]=011 N_{SS} = 128		mf[2:0]=100 N_{SS} = 64		mf[2:0]=101 N_{SS} = 32		mf[2:0]=110 N_{SS} ≈ 24		mf[2:0]=111 N_{SS} = 16	
mr[2:0]=000, k≈0	0 dB	0	0 dB	0	0 dB	0	0 dB	0	0 dB	0	0 dB	0	0 dB	0	0 dB	0
mr[2:0]=001, k≈1	21 dB	32	20 dB	24	18 dB	16	15 dB	8	12 dB	4	9 dB	2	8 dB	1.5	6 dB	1
mr[2:0]=010, k≈1.5	23 dB	48	22 dB	32	20 dB	24	17 dB	12	14 dB	6	11 dB	3	10 dB	2.2	8 dB	1.5
mr[2:0]=011, k≈2	24 dB	64	23 dB	48	21 dB	32	18 dB	16	15 dB	8	12 dB	4	11 dB	3	9 dB	2
mr[2:0]=100, k≈3	26 dB	96	25 dB	64	25 dB	48	20 dB	24	17 dB	12	14 dB	6	13 dB	4.5	12 dB	4
mr[2:0]=101, k≈4	27 dB	128	26 dB	96	24 dB	64	21 dB	32	18 dB	16	15 dB	8	14 dB	6	12 dB	4
mr[2:0]=110, k≈6	28 dB	192	28 dB	128	26 dB	96	23 dB	48	20 dB	24	17 dB	12	16 dB	9	14 dB	6
mr[2:0]=111, k≈8	30 dB	256	29 dB	192	27 dB	128	24 dB	64	21 dB	32	18 dB	16	17 dB	12	15 dB	8

6.6.5.3.4 PLL power-down mode

If the PLL is not used, or if it there are cases where it is turned off in a running application, power can be saved by putting the PLL in power-down mode. Before this is done, the CPU and any peripherals that are not meant to stopped as well must be running from some other clock source.

6.6.5.4 PLL related registers

The PLL is controlled by registers described elsewhere in this chapter (see Section 6.5.46), and summarized below.

Table 173. Summary of PLL related registers

Register	Description	Section
SYSPLLCTRL	PLL control	6.5.46.1
SYSPLLSTAT	PLL status	6.5.46.2
SYSPLLNDEC	PLL pre-divider	6.5.46.3
SYSPLLPDEC	PLL post-divider	6.5.46.4
SYSPLLSSCTRL0	PLL spread spectrum control 0	6.5.46.5.1
SYSPLLSSCTRL1	PLL spread spectrum control 1	6.5.46.5.2

6.6.5.5 PLL usage

As previously noted, the PLL divider settings used in the PLL registers are not simple binary values, they are encoded as shown in the PLL register descriptions. The divider values and their encoding can be found by calculation using the information in this document. For simple PLL usage with no pre- or post-divide, the LPCOpen Chip_POWER_SetPLL API can be used (see Section 8.4.1). Also, a PLL setting calculator can be found on the NXP website. The latter two possibilities are recommended in order to avoid PLL setup issues.

6.6.5.5.1 Procedure for determining PLL settings

In general, PLL configuration values may be found as follows:

1. Identify a desired PLL output frequency. This may depend on a specific interface frequency needed or be based on expected CPU performance requirements, and may be limited by system power availability.
2. Determine which clock source to use as the PLL input. This can be influenced by power or accuracy required, or by the potential to obtain the desired PLL output frequency.
3. Identify PLL settings to obtain the desired output from the selected input. The Fcco frequency must be either the actual desired output frequency, or the desired output frequency times 2 x P, where P is from 2 to 32. The Fcco frequency must also be a multiple of the PLL reference frequency, which is either the PLL input, or the PLL input divided by N, where N is from 2 to 256.
4. There may be several ways to obtain the same PLL output frequency. PLL power depends on Fcco (a lower frequency uses less power) and the divider used. Bypassing the input and/or output divider saves power.
5. Check that the selected settings meet all of the PLL requirements:
   – Fin is in the range of 32 kHz to 25 MHz.

- Fcco is in the range of 75 MHz to 150 MHz.
- Fout is in the range of 1.2 MHz to 150 MHz.
- The pre-divider is either bypassed, or N is in the range of 2 to 256.
- The post-divider is either bypassed, or P is in the range of 2 to 32.
- M is in the range of 3 to 32,768.

Also note that PLL startup time becomes longer as Fref drops below 500 kHz. At 500 kHz and above, startup time is up to 500 microseconds. Below 500 kHz, startup time can be estimated as 200 / Fref, or up to 6.1 milliseconds for Fref = 32 kHz. PLL accuracy and jitter is better with higher values of Fref.

6.6.5.5.2 PLL setup sequence

The following sequence should be followed to initialize and connect the PLL:

1. Make sure that the PLL output is disconnected from any downstream functions. If the PLL was previously being used to clock the CPU, and the CPU Clock Divider is being used, it may be set to speed up operation while the PLL is disconnected.
2. Select a PLL input clock source. See Section 6.5.25 “System PLL clock source select register”.
3. Set up the PLL dividers and mode settings. See Section 6.5.46 “PLL registers”.
4. Wait for the PLL output to stabilize. The value of the PLl lock may not be stable when the PLL reference frequency (FREF, the frequency of REFCLK, which is equal to the PLL input frequency divided by the pre-divider value) is less than 100 kHz or greater than 20 MHz. In these cases, the PLL may be assumed to be stable after a start-up time has passed. This time is 500 µs when Fref is 500 kHz or greater and 200 / Fref seconds when FREF is less than 500 kHz.
5. If the PLL will be used to clock the CPU, change the CPU Clock Divider setting for operation with the PLL, if needed. This must be done before connecting the PLL.
6. Connect the PLL to whichever downstream function is will be used with. The structure of the clock dividers may be seen on the right of Figure 9 “Clock generation”.

6.6.6 Frequency measure function

The Frequency Measure circuit is based on two 14-bit counters, one clocked by the reference clock and one by the target clock. Synchronization between the clocks is performed at the start and end of each count sequence.

A measurement cycle is initiated by software setting a control/status bit in the FREQMECTRL register (Table 140). The software can then poll this same measurement-in-progress bit which will be cleared by hardware when the measurement operation is completed.

The measurement cycle terminates when the reference counter rolls-over. At that point the state of the target counter is loaded into a capture field in the FREQMEAS register, and the measure-in-progress bit is cleared. Software can read this capture value and apply to it a specific calculation which will return the precise frequency of the target clock in MHz.

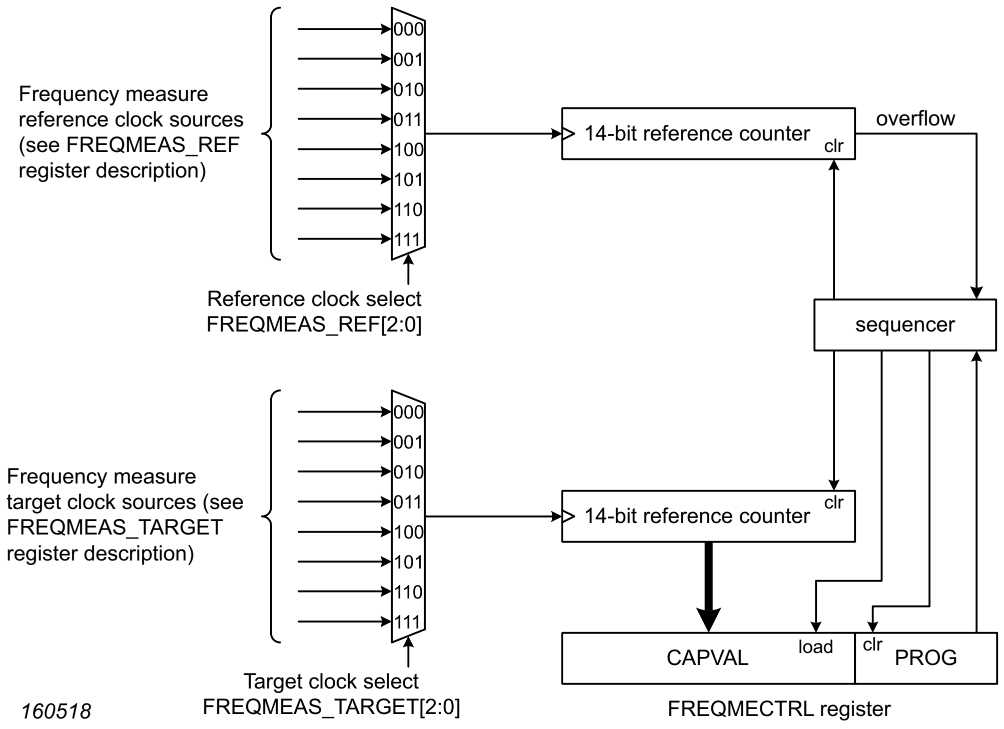

Frequency measure reference clock sources (see FREQMEAS_REF register description)
000
001
010
011
100
101
110
111
Reference clock select FREQMEAS_REF[2:0]
14-bit reference counter
clr
overflow
sequencer
Frequency measure target clock sources (see FREQMEAS_TARGET register description)
Target clock select FREQMEAS_TARGET[2:0]
14-bit reference counter
clr
CAPVAL
load
clr
PROG
FREQMECTRL register
160518

Fig 13. Frequency measure block diagram

Figure 13 shows a block diagram of the frequency measure function. Also see:

- Section 6.2.3 "Measure the frequency of a clock signal"
- Frequency measure control register (FREQMECTRL)- Section 6.5.41
- Frequency reference clock select register (FREQMEAS_REF) - Section 10.6.4
- Frequency target clock select register (FREQMEAS_TARGET) - Section 10.6.5

6.6.6.1 Accuracy

The frequency measurement function can measure the frequency of any on-chip (or off-chip) clock (referred to as the target clock) to a high degree of accuracy using another on-chip clock of known frequency as a reference.

The following constraints apply:

- The frequency of the reference clock must be (somewhat) greater that the frequency of the target clock.
- The system clock used to access the frequency measure function register must also be greater than the frequency of the target clock.

The frequency measurement function circuit is able to measure the target frequency with an error of less than 0.1%, provided the reference frequency is precisely known.

Uncertainty in the reference clock (for example the ±1% accuracy of the FRO) will add to the measurement error of the target clock. In general, though, this additional error is less than the uncertainty of the reference clock.

There can also be a modest loss of accuracy if the reference frequency exceeds the target frequency by a very large margin (25x or more). Accuracy is not a simple function of the magnitude of the frequency difference, however. Nearly identical frequency combinations, still with a spread of about 43x, result in errors of less than 0.05%.

If the target and reference clocks are different by more than a factor of approximately 500, then the accuracy decreases to +/- 4%.

UM11071

Chapter 7: LPC51U68 Power management

Rev. 1.1 — 17 May 2018 **User manual**

7.1 Introduction

This chapter provides an overview of power related information about LPC51U68 devices. These devices include a variety of power switches and clock switches to allow fine tuning power usage to match requirements at different performance levels and reduced power modes.

To turn analog components on or off in active and deep sleep modes, use the PDRUNCFG0 and PDSLEEPCFG0 registers (see Table 152 and Table 151). In deep-sleep mode, the power profile API controls which analog peripherals remain powered up (see Section 8.4.3). There is no register implemented to turn analog peripherals on or off for deep-sleep mode.

7.2 General description

Power to the part is supplied via two power domains. The main power domain is powered by VDD and supplies power to the core, peripheral, memories, inputs and outputs via an on-chip regulator.

A second, always-on power domain is also powered by VDD, and includes the RTC and wake-up timer. This domain always has power as long as sufficient voltage is supplied to VDD.

Power usage is controlled by settings in register within the SYSCON block, regulator settings controlled via a Power API, and the operating mode of a CPU. The following modes are supported in order from highest to lowest power consumption:

1. Active mode:

   The part is in active mode after a Power-On Reset (POR) and when it is fully powered and operational after booting.

2. Sleep mode:

   Sleep mode saves a significant amount of power by stopping CPU execution without affecting peripherals or requiring significant wake-up time. The sleep mode affects the relevant CPU only. The clock to the core is shut off. Peripherals and memories are active and operational.

3. Deep-sleep mode:

   Deep-sleep mode is configurable and can potentially turn off nearly all on-chip power consumption other than the on-chip power supply, with the cost of a longer wake-up time. The deep-sleep mode affects the entire system, the clocks to CPUs are shut down and, if not configured, the peripherals receive no internal clocks. Device registers, and SRAMs that are not shut down maintain their contents. Some device features can be automatically disabled by setting up the value of the PDSLEEPCFG0 register (see Section 6.5.47). Entry to this mode can only be accomplished by the master CPU in devices that support 2 CPUs.

   Through the power profiles API, selected peripherals such as USB, SPI, I2C, USART, WWDT, RTC, Micro-tick Timer, and BOD can be left running in deep sleep mode.

4. Deep power-down mode:

   Deep power-down mode shuts down virtually all on-chip power consumption, but requires a significantly longer wake-up time. For maximal power savings, the entire system (CPUs and all peripherals) is shut down except for the PMU and the RTC. On wake-up, the part reboots. Entry to deep power-down mode can only be accomplished by the master CPU in a device that supports two CPUs.

Table 174. Peripheral configuration in reduced power modes

Peripheral	Reduced power mode		
	Sleep	Deep-sleep	Deep power-down
FRO	Software configured	Software configured	Off
Flash	Software configured	Standby	Off
BOD	Software configured	Software configured	Off
PLL	Software configured	Off	Off
Watchdog osc and WWDT	Software configured	Software configured	Off
Micro-tick Timer	Software configured	Software configured	Off
DMA	Active	Configurable some for operations, see Section 7.3.4	Off
USART	Software configured	Off; but can create a wake-up interrupt in synchronous slave mode or 32 kHz clock mode	Off
SPI, I2C, I2S	Software configured	Off; but can create a wake-up interrupt in slave mode	Off
USB	Software configured	Software configured	Off
Other digital peripherals	Software configured	Off	Off
RTC oscillator	Software configured	Software configured	Software configured

7.2.1 Wake-up process

The part always wakes up to the active mode. To wake up from the reduced power modes, you must configure the wake-up source. Each reduced power mode supports its own wake-up sources and needs to be configured accordingly as shown in Table 175.

Table 175. Wake-up sources for reduced power modes

Power mode	Wake-up source	Conditions
Sleep	Any interrupt	Enable interrupt in NVIC.
	HWWAKE	Certain Flexcomm Interface activity. See Section 6.5.54 “Hardware Wake-up control register”.

Table 175. Wake-up sources for reduced power modes

Power mode	Wake-up source	Conditions
Deep-sleep	Pin interrupts	Enable pin interrupts in NVIC and STARTER0 register.
	BOD interrupt	• Enable interrupt in NVIC and STARTER0 registers. • Enable interrupt in BODCTRL register. • Configure the BOD to keep running in this mode with the power API.
	BOD reset	Enable reset in BODCTRL register.
	Watchdog interrupt	• Enable the watchdog oscillator in the PDRUNCFG0 register. • Enable the watchdog interrupt in NVIC and STARTER0 registers. • Enable the watchdog in the WWDT MOD register and feed. • Enable interrupt in WWDT MOD register. • Configure the WDTOSC to keep running in this mode with the power API.
	Watchdog reset	• Enable the watchdog oscillator in the PDRUNCFG0 register. • Enable the watchdog and watchdog reset in the WWDT MOD register and feed.
	Reset pin	Always available.
	RTC 1 Hz alarm timer	• Enable the RTC 1 Hz oscillator in the RTCOSCCTRL register. • Enable the RTC bus clock in the AHBCLKCTRL0 register. • Start RTC alarm timer by writing a time-out value to the RTC COUNT register. • Enable the RTCALARM interrupt in the STARTER0 register.
	RTC 1 kHz timer time-out and alarm	• Enable the RTC 1 Hz oscillator and the RTC 1 kHz oscillator in the RTC CTRL register. • Start RTC 1 kHz timer by writing a value to the WAKE register of the RTC. • Enable the RTC wake-up interrupt in the STARTER0 register.
	Micro-tick timer (intended for ultra-low power wake-up from deep-sleep mode	• Enable the watchdog oscillator in the PDRUNCFG0 register. • Enable the Micro-tick timer clock by writing to the AHBCLKCTRL1 register. • Start the Micro-tick timer by writing UTICK CTRL register. • Enable the Micro-tick timer interrupt in the STARTER0 register.
	I2C interrupt	Interrupt from I2C in slave mode. See Chapter 26 “LPC51U68 I2C-bus interfaces”.
	SPI interrupt	Interrupt from SPI in slave mode. See Chapter 25 “LPC51U68 Serial Peripheral Interfaces (SPI)”.
	USART interrupt	Interrupt from USART in slave or 32 kHz mode. See Chapter 24 “LPC51U68 USARTs”.
	USB need clock interrupt	Interrupt from USB when activity is detected that requires a clock. See Section 22.7.6 “USB wake-up”.
	DMA interrupt	See Chapter 14 for details of DMA-related interrupts.
	HWWAKE	Certain Flexcomm Interface activity. See Section 6.5.54 “Hardware Wake-up control register”.
Deep power-down	RTC 1 Hz alarm timer	• Enable the RTC 1 Hz oscillator in the RTC CTRL register. • Start RTC alarm timer by writing a time-out value to the RTC COUNT register.
	RTC 1 kHz timer time-out and alarm	• Enable the RTC 1 Hz oscillator and the RTC 1 kHz oscillator in the RTCOSCCTRL register. • Enable the RTC bus clock in the AHBCLKCTRL0 register. • Start RTC 1 kHz timer by writing a value to the WAKE register of the RTC.
	Reset pin	Always available.

7.3 Functional description

7.3.1 Power management

The LPC51U68 support a variety of power control features. In Active mode, when the chip is running, power and clocks to selected peripherals can be optimized for power consumption. In addition, there are three special modes of processor power reduction with different peripherals running: sleep mode, deep-sleep mode, and deep power-down mode, activated by the power mode configure API (see Section 8.4.3).

Remark: The Debug mode is not supported in sleep, deep-sleep, or deep power-down modes.

7.3.2 Active mode

In Active mode, the CPU, memories, and peripherals are clocked by the AHB/CPU clock.

The chip is in Active mode after reset and the default power configuration is determined by the reset values of the PDRUNCFG0, AHBCLKCTRL0, and AHBCLKCTRL1 registers. The power configuration can be changed during run time.

7.3.2.1 Power configuration in Active mode

Power consumption in Active mode is determined by the following configuration choices:

- The AHBCLKCTRL registers control which memories and peripherals are running (Section 6.5.16 “AHB Clock Control register 0” and Section 6.5.17 “AHB Clock Control register 1”). Generally speaking, in order to save power, functions that are not needed by the application should be turned off. If specific times are known when certain functions will not be needed, they can be turned off temporarily and turned back on when they will be needed.
- The power to various analog blocks (RAMs, PLL, oscillators, and the BOD circuit) can be controlled individually through the PDRUNCFG0 register (Table 152). As with clock controls, these blocks should generally be tuned off if not needed by the application. If turned off, time will be needed before these blocks can be used again after being turned on.
- The clock source for the system clock can be selected from the FRO (default), the 32 kHz oscillator, or the watchdog oscillator (see Figure 9 and related registers).
- The system clock frequency can be selected (see Section 6.6.5 “System PLL functional description” and other clocking related sections). Generally speaking, everything uses less power at lower frequencies, so running the CPU and other device features at a frequency sufficient for the application (plus some margin) will save power. If the PLL is not needed, it should be turned off to save power. Also, running the PLL at a lower CCO frequency saves power.
- Several peripherals use individual peripheral clocks with their own clock dividers. The peripheral clocks can be shut down through the corresponding clock divider registers if the base clock is still needed for another function.
- The power library provides an easy way to optimize power consumption depending on CPU load and performance requirements. See Chapter 8 “LPC51U68 Power profiles/Power control API”.

7.3.3 Sleep mode

In sleep mode, the system clock to the CPU is stopped and execution of instructions is suspended until either a reset or an interrupt occurs.

Peripheral functions, if selected to be clocked in the AHBCLKCTRL registers, continue operation during sleep mode and may generate interrupts to cause the processor to resume execution. Sleep mode eliminates dynamic power used by the processor itself, memory systems and related controllers, and internal buses. The processor state and registers, peripheral registers, and internal SRAM values are maintained, and the logic levels of the pins remain static.

As in active mode, the power API provides an easy way to optimize power consumption depending on CPU load and performance requirements in sleep mode. See Chapter 8 "LPC51U68 Power profiles/Power control API".

7.3.3.1 Power configuration in sleep mode

Power consumption in sleep mode is configured by the same settings as in Active mode:

- Enabled clocks remain running.
- The system clock frequency remains the same as in Active mode, but the processor is not clocked.
- Analog and digital peripherals are powered and selected as in Active mode through the PDRUNCFG0, AHBCLKCTRL0, and AHBCLKCTRL1 registers.

7.3.3.2 Programming sleep mode

The following steps must be performed to enter sleep mode:

1. In the NVIC, enable all interrupts that are needed to wake up the part.
2. Alter PDRUNCFG if needed to reflect any functions that should be on or off during sleep mode.
3. Execute the WFI instruction to enter sleep mode

7.3.3.3 Wake-up from sleep mode

Sleep mode is exited automatically when an interrupt enabled by the NVIC arrives at the processor or a reset occurs. After wake-up caused by an interrupt, the device returns to its original power configuration defined by the contents of the PDRUNCFG0 and the AHBCLKCTRL registers. If a reset occurs, the microcontroller enters the default configuration in Active mode.

7.3.4 Deep-sleep mode

In deep-sleep mode, the system clock to the processor is disabled as in sleep mode. Analog blocks are powered down by default but can be selected to keep running through the power API if needed as wake-up sources. The main clock and all peripheral clocks are disabled. The FRO is disabled. The flash memory is put in standby mode.

Deep-sleep mode eliminates power used by analog peripherals and all dynamic power used by the processor itself, memory systems and related controllers, and internal buses. The processor state and registers, peripheral registers, and internal SRAM values are maintained, and the logic levels of the pins remain static.

GPIO Pin Interrupts, GPIO Group Interrupts, and selected peripherals such as USB, SPI, I2C, USART, WWDT, RTC, Micro-tick Timer, and BOD can be left running in deep sleep mode The FRO, RTC oscillator, and the watchdog oscillator can be left running.

In some cases, DMA can operate in deep-sleep mode, see Section 14.5.9 “DMA in reduced power modes” and Section 6.5.54 “Hardware Wake-up control register”.

7.3.4.1 Power configuration in deep-sleep mode

Power consumption in deep-sleep mode is determined primarily by which analog wake-up sources remain enabled. Serial peripherals and pin interrupts configured to wake up the contribute to the power consumption only to the extent that they are clocked by external sources. All wake-up events (other than reset) must be enabled in the STARTER registers and in the NVIC. In addition, any related analog block (for example, the RTC oscillator or the watchdog oscillator) must be explicitly enabled through a power API function See Table 175 and Chapter 8 “LPC51U68 Power profiles/Power control API”.

7.3.4.2 Programming deep-sleep mode

The following steps must be performed to enter deep-sleep mode:

1. Select wake-up sources and enable all selected wake-up events in the STARTER registers (Table 155) and in the NVIC.
2. Select the FRO 12 MHz as the main clock. See Table 121 and Table 122.
3. On power-up, the BOD is enabled. Power API disables BOD in deep-sleep mode. User must disable BOD reset (bit 2 in the BODCTRL register) and clear bit ‘6’ in the BODCTRL register before calling the power API to enter deep-sleep mode.
4. Call the power API with the peripheral parameter to enable the analog peripherals as wake-up sources (see Chapter 8 “LPC51U68 Power profiles/Power control API”). The peripheral parameter is a 32-bit value that corresponds to bits in the PDRUNCFG0 register. Alternatively, set up the PDSLEEPCFG0 register appropriately, set the SLEEPDEEP bit in the Cortex-M0+ SCR register, and execute a WFI instruction.

7.3.4.3 Wake-up from deep-sleep mode

The part can wake up from deep-sleep mode in the following ways:

- Using a signal on one of the eight pin interrupts selected in Section 10.6.1 “Pin interrupt select registers”. Each pin interrupt must also be enabled in the STARTER0 register (Table 155) and in the NVIC.
- Using an interrupt from a block such as the watchdog interrupt or RTC interrupt, when enabled during the reduced power mode via the power API. Also enable the wake-up sources in the STARTER registers (Table 155) and the NVIC.
- Using a reset from the $\overline{\text{RESET}}$ pin, or the BOD or WWDT (if enabled in the power API).
- Using a wake-up signal from any of the serial peripherals that are operating in deep-sleep mode. Also enable the wake-up sources in the STARTER registers (Table 155) and the NVIC.
- GPIO group interrupt signal. The interrupt must also be enabled in the STARTER0 register (Table 155) and in the NVIC.
- RTC alarm signal or wake-up signal. See Chapter 18. Interrupts must also be enabled in the STARTER0 register (Table 155) and in the NVIC.

7.3.5 Deep power-down mode

In deep power-down mode, power and clocks are shut off to the entire chip with the exception of the RTC.

During deep power-down mode, the contents of the SRAM and registers (other than those in the RTC) are not retained. All functional pins are tri-stated in deep power-down mode as long as chip power supplied externally.

7.3.5.1 Power configuration in deep power-down mode

Deep power-down mode has no configuration options. All clocks, the core, and all peripherals are powered down. Only the RTC is powered, as long as power is supplied to the device.

7.3.5.2 Wake-up from deep power-down mode:

Wake-up from deep power-down can be accomplished via the reset pin or the RTC.

7.3.5.3 Programming deep power-down mode using the RTC for wake-up:

The following steps must be performed to enter deep power-down mode when using the RTC for waking up:

1. Set up the RTC high resolution timer. Write to the RTC VAL register. This starts the high res timer if enabled. Another option is to use the 1 Hz alarm timer.
2. Call the power API function, see Chapter 8 “LPC51U68 Power profiles/Power control API”.

7.3.5.4 Wake-up from deep power-down mode using the RTC:

The part goes through the entire reset process when the RTC times out:

- The PMU will turn on the on-chip voltage regulator. When the core voltage reaches the power-on-reset (POR) trip point, a system reset will be triggered and the chip boots.
- All registers will be in their reset state.

UM11071

Chapter 8: LPC51U68 Power profiles/Power control API

Rev. 1.1 — 17 May 2018 **User manual**

8.1 How to read this chapter

The Power profiles and Power control APIs can be implemented using the power library from LPCOpen software package, or the SDK software package available on nxp.com.

8.2 Features

- Simple APIs to control power consumption and wake-up in all power modes.
- Manage power consumption for sleep and active modes
- Prepare the part to enter low power modes (sleep, deep-sleep, and deep power-down).
- Configure wake-up from deep-sleep via functions enabled by bits in the PDRUNCFG0 register.

8.3 General description

PLL setup and control of device power consumption or entry to low power modes can be configured through simple calls to the power profile. APIs exist to:

- Set up the System PLL (API call available in LPCOpen).
- Set up on-chip power based on the expected operating frequency.
- Set up reduced power modes.
- Set up low-frequency low power operation.

Remark: Disable all interrupts before making calls to the power profile API. The interrupts can be re-enabled after the power profile API calls have completed.

8.4 API description

Power APIs provide functions to configure the system clock and set up the system for expected performance requirements. The Power APIs are available in the power library provided with LPCOpen software package (see Table 176 “Power API calls (available in the LPCOpen software package)”) and with the SDK software package (see Table 177 “Power API calls (available in the SDK software package)”).

Table 176. Power API calls (available in the LPCOpen software package)

Function prototype	API description	Section
`uint32_t Chip_POWER_SetPLL (uint32_t multiply_by, uint32_t input_freq);`	Power API PLL configuration routine.This API sets up basic PLL operation.	8.4.1
`uint32_t Chip_POWER_SetVoltage (uint32_t desired_freq);`	Power API internal voltage configuration routine.This API configures the internal regulator for the desired active operating mode and frequency. Also sets up corresponding flash wait states.	8.4.2
`void Chip_POWER_EnterPowerMode (POWER_MODE_T mode, uint32_t peripheral_ctrl);`	Power API power mode configuration routine.This API prepares the chip for reduced power modes: sleep, deep-sleep, or deep power-down mode. Also allows selection of which peripherals are kept alive in the reduced mode, and can therefore wake up the device from that mode.	8.4.3
`void Chip_POWER_SetLowPowerVoltage (uint32_t freq)`	Set voltage levels for low power regulation mode. Voltages are optimized for FRO operation at 12 MHz or 48 MHz.	8.4.4
`void Chip_POWER_SetFROHFRate (uint32_t freq)`	Setup the FRO high frequency output for either 48 MHz or 96 MHz. Updates the correct trim value and settings for high frequency FRO operation.	8.4.5

Table 177. Power API calls (available in the SDK software package)

Function prototype	API description	Section
`uint32_t POWER_SetVoltageForFreq (uint32_t desired_freq);`	Power API internal voltage configuration routine.This API configures the internal regulator for the desired active operating mode and frequency. Also sets up corresponding flash wait states.	8.4.2
`void POWER_EnterPowerMode (power_mode_cfg_t mode, uint64_t exclude_from_pd);`	Power API power mode configuration routine.This API prepares the chip for reduced power modes: sleep, deep-sleep, or deep power-down mode. Also allows selection of which peripherals are kept alive in the reduced mode, and can therefore wake up the device from that mode.	8.4.3
`void POWER_SetLowPowerVoltageForFreq (uint32_t freq)`	Set voltage levels for low power regulation mode. Voltages are optimized for FRO operation at 12 MHz or 48 MHz.	8.4.4
`void CLOCK_SetupFROClocking (uint32_t iFreq);`	Setup the FRO high frequency output for either 48 MHz or 96 MHz. Updates the correct trim value and settings for high frequency FRO operation.	8.4.5

8.4.1 Chip_POWER_SetPLL (available in LPCOpen)

This routine sets up the System PLL given the PLL input frequency and feedback multiplier. Note that this API does not support special PLL operating modes. A library call is available via LPCOpen that supports additional PLL features.

The Chip_POWER_SetPLL works by setting the PLL input pre-divider to 2. It then determines what multiplier and post divider settings to use to get the requested frequency (M * input frequency) while keeping the PLL within its various operating limits.

Table 178. Chip_POWER_SetPLL API routine

Routine	Chip_POWER_SetPLL
LPCOpen Prototype	`uint32_t Chip_POWER_SetPLL (uint32_t multiply_by, uint32_t input_freq);`
Input parameter	**Param0:** multiplier (1 to 16) **Param1:** input_freq
Result	Error code. 0 = no error.
Description	Sets up the PLL for the requested multiplier and input frequency.

8.4.1.1 Param0: multiplier

The input parameter multiplier (Param0) specifies the feedback multiplier for the PLL. The range supported is from 1 to 16.

8.4.1.2 Param1: input_freq

The input frequency is the clock rate of the PLL input. The input frequency times the multiplier must not be greater than 100 MHz.

8.4.1.3 Error or return codes

A return code of zero indicates that the operation was successful.

Table 179. Error codes

Return code	Error code	Description
0x000B 0002	ERR_CLK_INVALID_PARAM	Parameter out of range or requested frequency too high or not possible

8.4.2 Chip_POWER_SetVoltage (LPCOpen) and POWER_SetVoltageForFreq (SDK)

This routine configures the device's internal power control settings according to the calling arguments. The goal is to prepare on-chip regulators to deliver the amount of power needed for the requested performance level, as defined by the CPU operating frequency.

Remark: The Chip_POWER_SetVoltage API and POWER_SetVoltageForFreq APIs should only be used when the system clock divider is 1 (AHBCLKDIV = 1, see Table 131).

Table 180. Chip_POWER_SetVoltage and POWER_SetVoltageForFreq API routines

Routine	Chip_POWER_SetVoltage and POWER_SetVoltageForFreq
LPCOpen Prototype	`uint32_t Chip_POWER_SetVoltage(uint32_t desired_freq);`
SDK Prototype	`uint32_t POWER_SetVoltageForFreq(uint32_t desired_freq);`
Input parameter	**Param0:** desired frequency (in Hz)
Result	Error code. 0 = no error.
Description	Configures the internal device voltage in active mode, as well as setting up the corresponding flash wait states.

8.4.2.1 Param0: frequency

The frequency is the clock rate the CPU will be using during the selected mode. microcontroller uses to source the system and peripheral clocks. This operand must represent an integer from 1 to 100 MHz inclusive.

8.4.2.2 Error or return codes

A return code of zero indicates that the operation was successful.

Table 181. Error codes

Return code	Error code	Description
0x000C 0004	PWR_ERROR_INVALID_CFG	Mode is invalid
0x000B 0002	ERR_CLK_INVALID_PARAM	Frequency is outside the supported range

8.4.3 Chip_POWER_EnterPowerMode (LPCOpen) and POWER_EnterPowerMode (SDK)

The Chip_POWER_EnterPowerMode and POWER_EnterPowerMode APIs prepare the part, then enter any of the low power modes. Specifically for the deep-sleep mode, the API function configures which analog components remain running in those two modes, so that an interrupt from one of the analog peripherals can wake up the part.

Table 182. Chip_POWER_EnterPowerMode and POWER_EnterPowerMode API routines

Routine	Chip_POWER_EnterPowerMode and POWER_EnterPowerMode
LPCOpen Prototype	`void Chip_POWER_EnterPowerMode (POWER_MODE_T mode, uint32_t peripheral_ctrl);`
SDK Prototype	`void POWER_EnterPowerMode (power_mode_cfg_t mode, uint64_t exclude_from_pd);`
Input parameter	**Param0:** mode **Param1:** peripheral
Result	None
Description	Defines the low power mode (either sleep, deep-sleep, or deep power-down modes) and allows controlling which peripherals are powered up in the reduced power mode.

Remark: Aside from the analog peripherals listed with this parameter, the serial peripherals can also wake up the chip from deep-sleep mode from an interrupt triggered by an incoming signal. This wake-up scenario is not configured using the Chip_POWER_EnterPowerMode and POWER_EnterPowerMode APIs. For details, see Section 24.3.2 "Configure the USART for wake-up", Section 25.3.1 "Configure the SPI for wake-up", or Section 26.4.3 "Configure the I^{2}C for wake-up".

8.4.3.1 Param0: mode

The mode parameter defines the low power mode and prepares the chip to enter the selected mode.

The following modes are valid:

```
typedef enum {
               POWER_SLEEP = 0,
               POWER_DEEP_SLEEP,
               POWER_DEEP_POWER_DOWN
} POWER_MODE_T;
```

8.4.3.2 Param1: peripheral

If sleep mode is selected with the mode parameter, the peripheral parameter is ignored.

The peripheral parameter defines which analog peripherals can wake up the chip from deep-sleep, or deep power-down mode. The selected peripherals remain running in deep-sleep mode. For example, the watchdog oscillator must be running if the WWDT is to remain active in deep-sleep mode or the RTC must be running if it is to remain active in deep power-down mode.

The peripheral parameter is a 32-bit value that corresponds to the PDRUNCFG0 register (see Table 152).

8.4.4 Chip_POWER_SetLowPowerVoltage

This routine sets up internal voltage levels for low power regulation mode. The selected operating frequency must be either 12 MHz or 48 MHz, the two optimal frequencies for power dissipation for the device running from the FRO.

Table 183. Chip_POWER_SetLowPowerVoltage API routine

Routine	Chip_POWER_SetVoltage
LPCOpen Prototype	`void Chip_POWER_SetLowPowerVoltage (uint32_t freq)`
Input parameter	**Param0:** desired frequency (in Hz)
Result	None.
Description	Set voltage levels for low power regulation mode.

8.4.4.1 Param0: frequency

The frequency is the clock rate the CPU and peripherals will be using, either 12 MHz or 48 MHz.

8.4.5 Chip_POWER_SetFROHFRate (LPCOpen) and CLOCK_SetupFROClocking (SDK, defined in fsl_clock.c)

This routine sets up the FRO high frequency output. The selected operating frequency must be either 48 MHz or 96 MHz, which are the two potential high frequency outputs of the FRO. The requested frequency is set up and the appropriate factory trim value will be used.

Table 184. Chip_POWER_SetFROHFRate and CLOCK_SetupFROClocking API routines

Routine	Chip_POWER_SetFROHFRate and CLOCK_SetupFROClocking
LPCOpen Prototype	`void Chip_POWER_SetFROHFRate (uint32_t freq)`
SDK Prototype	`void CLOCK_SetupFROClocking (uint32_t iFreq);`
Input parameter	**Param0:** desired frequency (in Hz)
Result	None.
Description	Setup the FRO high frequency output for the selected frequency, either 48 MHz or 96 MHz.

8.4.5.1 Param0: frequency

The frequency is the desired high frequency output clock for the FRO, 48 MHz or 96 MHz.

8.5 Functional description

8.5.1 Example low power mode control using LPCOpen

8.5.1.1 Enter sleep mode

```
/* peripheral parameter is don’t care*/
Chip_POWER_EnterPowerMode (POWER_SLEEP, 0x0 );
/* going to sleep mode. */
```

8.5.1.2 Enter deep-sleep mode and set up WWDT and BOD for wake-up

```
/* configure wwdt and bod event to wake up the chip from deep-sleep*/
LPC_SYSCON->STARTER0 = 0x3;
/* WDT_OSC and BOD are turned on */
Chip_POWER_EnterPowerMode (       POWER_DEEP_SLEEP, PDRUNCFG_PD_WDT_OSC |
                                  PDRUNCFG_PD_BOD_RESET | PDRUNCFG_PD_BOD_INTR);
/* going to deep-sleep mode. */
```

9.1 How to read this chapter

The IOCON block is included on all LPC51U68 parts. Registers for pins that are not available on a specific package are reserved.

Table 185. Available pins and configuration registers

Package	Total GPIOs	GPIO Port 0	GPIO Port 1
64-pins	48	PIO0_0 to PIO0_26, PIO0_29 to PIO0_31	PIO1_0 to PIO1_17
48-pins	37	PIO0_0 to PIO0_1, PIO0_4 to PIO0_26, PIO0_29 to PIO0_31	PIO1_0 to PIO1_8

Remark: Some functions, such as SCTimer/PWM inputs, Frequency Measure, and ADC triggers are not selected through IOCON. The connections for these function are described in either the Input Multiplexing chapter (Chapter 10) or the chapter for the specific function.

9.2 Features

The following electrical properties are configurable for standard port pins:

- Pull-up/pull-down resistor
- Open-drain mode
- Inverted function

Pins PIO0_23 through PIO0_26 are true open-drain pins that can be configured for different I^{2}C-bus speeds. Configuration options are somewhat different for these pins, as described in this chapter. Refer to a specific device data sheets for electrical details of these and other pins.

9.3 Basic configuration

Enable the clock to the IOCON in the AHBCLKCTRL0 register (Table 115). Once the pins are configured, the IOCON clock can be disabled in order to conserve power.

9.4 General description

9.4.1 Pin configuration

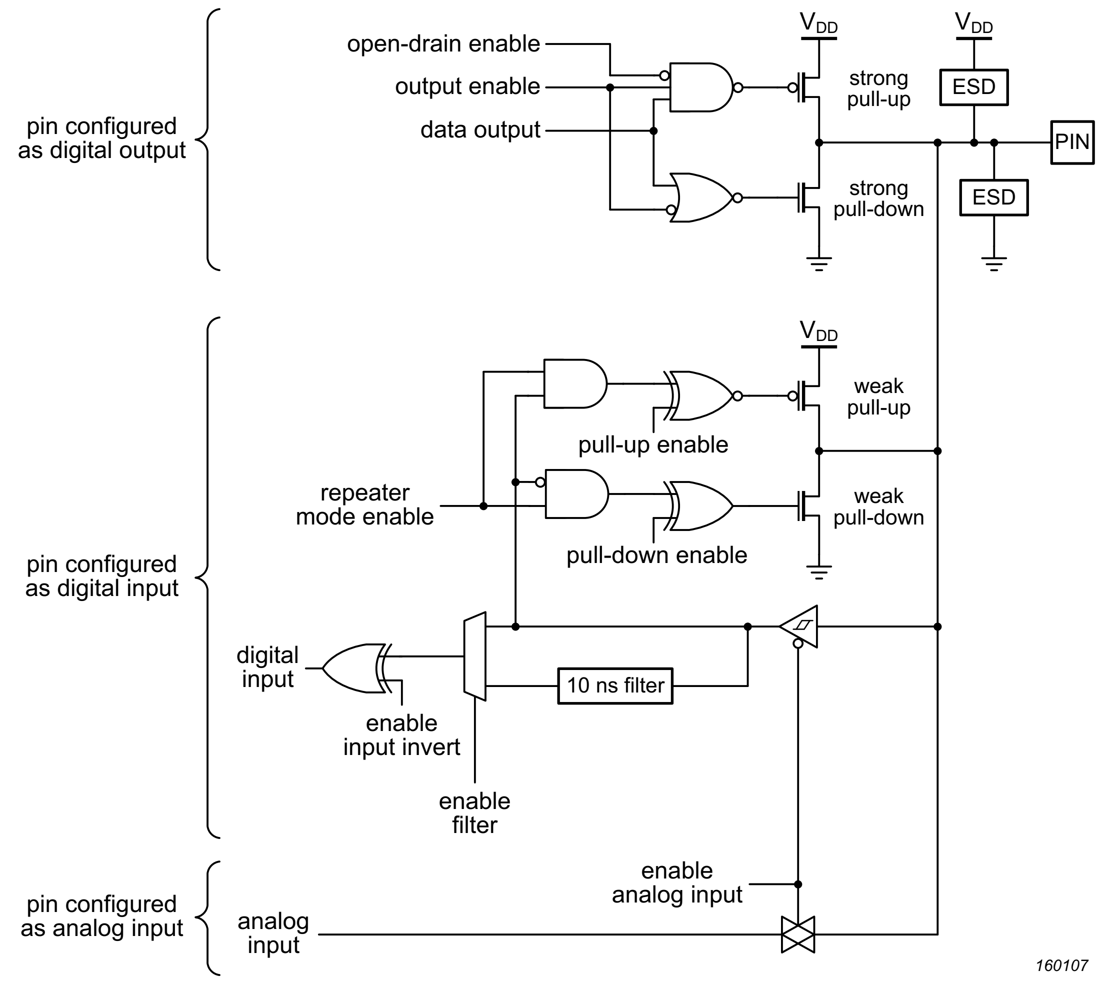


Fig 14. Pin configuration

9.4.2 IOCON registers

The IOCON registers control the functions of device pins. Each GPIO pin has a dedicated control register to select its function and characteristics. Each pin has a unique set of functional capabilities. Not all pin characteristics are selectable on all pins. For instance, pins that have an I^{2}C function can be configured for different I^{2}C-bus modes, while pins that have an analog alternate function have an analog mode can be selected.Details of the IOCON registers are in Section 9.4.2. The following sections describe specific characteristics of pins.

Multiple connections

Since a particular peripheral function may be allowed on more than one pin, it is possible to configure more than one pin to perform the same function. If a peripheral output function is configured to appear on more than one pin, it will in fact be routed to those pins. If a peripheral input function is defined as coming from more than one source, the values will be logically combined, possibly resulting in incorrect peripheral operation. Therefore care should be taken to avoid this situation.

9.4.2.1 Pin function

The FUNC bits in the IOCON registers can be set to GPIO (typically value 0) or to a special function. For pins set to GPIO, the DIR registers determine whether the pin is configured as an input or output (see Section 11.5.3). For any special function, the pin direction is controlled automatically depending on the function. The DIR registers have no effect for special functions.

9.4.2.2 Pin mode

The MODE bits in the IOCON register allow the selection of on-chip pull-up or pull-down resistors for each pin or select the repeater mode.

The possible on-chip resistor configurations are pull-up enabled, pull-down enabled, or no pull-up/pull-down. The default value is pull-up enabled.

The repeater mode enables the pull-up resistor if the pin is high and enables the pull-down resistor if the pin is low. This causes the pin to retain its last known state if it is configured as an input and is not driven externally. Such state retention is not applicable to the deep power-down mode. Repeater mode may typically be used to prevent a pin from floating (and potentially using significant power if it floats to an indeterminate state) if it is temporarily not driven.

9.4.2.3 Hysteresis

The input buffer for digital functions has built-in hysteresis. See the appropriate specific device data sheet for quantitative details.

9.4.2.4 Invert pin

This option is included to avoid having to include an external inverter on an input that is meant to be the opposite polarity of the external signal.

9.4.2.5 Analog/digital mode

When not in digital mode (DIGIMODE = 0) a pin is in analog mode, some digital pin functions are disabled and any analog pin functions are enabled. In digital mode (DIGIMODE = 1), any analog pin functions are disabled and digital pin functions are enabled. This protects the analog input from voltages outside the range of the analog power supply and reference that may sometimes be present on digital pins, since they are typically 5V tolerant. All pin types include this control, even if they do not support any analog functions.

In order to use a pin that has an ADC input option for that purpose, select GPIO (FUNC field = 0) and disable the digital pin function (DIGIMODE = 0). The MODE field should also be set to 0.

In analog mode, the MODE field should be “Inactive” (00); the INVERT, FILTEROFF, and OD settings have no effect. For an unconnected pin that has an analog function, keep the DIGIMODE bit set to 1 (digital mode), and pull-up or pull-down mode selected in the MODE field.

9.4.2.6 Input filter

Some pins include a filter that can be selectively disabled by setting the FILTEROFF bit. The filter suppresses input pulses smaller than about 10 ns.

9.4.2.7 Output slew rate

The SLEW bits of digital outputs that do not need to switch state very quickly should be set to “standard”. This setting allows multiple outputs to switch simultaneously without noticeably degrading the power/ground distribution of the device, and has only a small effect on signal transition time. This is particularly important if analog accuracy is significant to the application. See the relevant specific device data sheet for more details.

9.4.2.8 I^{2}C modes

Pins that support I^{2}C with specialized pin electronics (PIO0_23 through PIO0_28) have additional configuration bits. These have multiple configurations to support I^{2}C variants. These are not hard-wired so that the pins can be more easily used for non-I^{2}C functions. See Table 191 for recommended mode settings.

For non-I^{2}C operation, these pins remain open-drain and can only drive low, regardless of how I2CSLEW and I2CDRIVE are set. They would typically be used with an external pull-up resistor if they are used as outputs unless they are used only to sink current. Leave I2CSLEW = 1, I2CDRIVE = 0, and I2CFILTER = 0 to maximize compatibility with other GPIO pins.

9.4.2.9 Open-Drain Mode

When output is selected, either by selecting a special function in the FUNC field, or by selecting the GPIO function for a pin having a 1 in the related bit of that port’s DIR register, a 1 in the OD bit selects open-drain operation, that is, a 1 disables the high-drive transistor. This option has no effect on the primary I^{2}C pins. Note that the properties of a pin in this simulated open-drain mode are somewhat different than those of a true open drain output.

9.5 Register description

Each port pin PIOm_n has one IOCON register assigned to control the characteristics of the pin.

Remark: See to the Pinning information section of the appropriate device data sheet for details on which pins listed in Table 186 exist on each package configuration.

Table 186. Register overview: I/O configuration (base address 0x4000 1000)

Name	Pin type	Access	Offset	Description	Reset value [1]	Section
PIO0_0	D	R/W	0x000	Pin control for port 0 pin 0	0x0190	9.5.1
PIO0_1	D	R/W	0x004	Pin control for port 0 pin 1.	0x0190	9.5.1
PIO0_2	D	R/W	0x008	Pin control for port 0 pin 2.	0x0190	9.5.1
PIO0_3	D	R/W	0x00C	Pin control for port 0 pin 3.	0x0190	9.5.1
PIO0_4	D	R/W	0x010	Pin control for port 0 pin 4.	0x0190	9.5.1
PIO0_5	D	R/W	0x014	Pin control for port 0 pin 5.	0x0190	9.5.1
PIO0_6	D	R/W	0x018	Pin control for port 0 pin 6.	0x0190	9.5.1
PIO0_7	D	R/W	0x01C	Pin control for port 0 pin 7.	0x0190	9.5.1
PIO0_8	D	R/W	0x020	Pin control for port 0 pin 8.	0x0190	9.5.1
PIO0_9	D	R/W	0x024	Pin control for port 0 pin 9.	0x0190	9.5.1
PIO0_10	D	R/W	0x028	Pin control for port 0 pin 10.	0x0190	9.5.1
PIO0_11	D	R/W	0x02C	Pin control for port 0 pin 11.	0x0190	9.5.1
PIO0_12	D	R/W	0x030	Pin control for port 0 pin 12.	0x0190	9.5.1
PIO0_13	D	R/W	0x034	Pin control for port 0 pin 13.	0x0190	9.5.1
PIO0_14	D	R/W	0x038	Pin control for port 0 pin 14.	0x0190	9.5.1
PIO0_15	D	R/W	0x03C	Pin control for port 0 pin 15.	0x0190	9.5.1
PIO0_16	D	R/W	0x040	Pin control for port 0 pin 16.	0x0195	9.5.1
PIO0_17	D	R/W	0x044	Pin control for port 0 pin 17.	0x0195	9.5.1
PIO0_18	D	R/W	0x048	Pin control for port 0 pin 18.	0x0190	9.5.1
PIO0_19	D	R/W	0x04C	Pin control for port 0 pin 19.	0x0190	9.5.1
PIO0_20	D	R/W	0x050	Pin control for port 0 pin 20.	0x0190	9.5.1
PIO0_21	D	R/W	0x054	Pin control for port 0 pin 21.	0x0190	9.5.1
PIO0_22	D	R/W	0x058	Pin control for port 0 pin 22.	0x0190	9.5.1
PIO0_23	I	R/W	0x05C	Pin control for port 0 pin 23 with I^{2}C support. [2]	0x01A0	9.5.2
PIO0_24	I	R/W	0x060	Pin control for port 0 pin 24 with I^{2}C support. [2]	0x01A0	9.5.2
PIO0_25	I	R/W	0x064	Pin control for port 0 pin 25 with I^{2}C support. [2]	0x01A0	9.5.2
PIO0_26	I	R/W	0x068	Pin control for port 0 pin 26 with I^{2}C support. [2]	0x01A0	9.5.2
PIO0_29	A	R/W	0x074	Pin control for port 0 pin 29. Includes an ADC input.	0x0190	9.5.3
PIO0_30	A	R/W	0x078	Pin control for port 0 pin 30. Includes an ADC input.	0x0190	9.5.3
PIO0_31	A	R/W	0x07C	Pin control for port 0 pin 31. Includes an ADC input.	0x0190	9.5.3
PIO1_0	A	R/W	0x080	Pin control for port 0 pin 0. Includes an ADC input.	0x0190	9.5.3
PIO1_1	A	R/W	0x084	Pin control for port 1 pin 1. Includes an ADC input.	0x0190	9.5.3
PIO1_2	A	R/W	0x088	Pin control for port 1 pin 2. Includes an ADC input.	0x0190	9.5.3
PIO1_3	A	R/W	0x08C	Pin control for port 1 pin 3. Includes an ADC input.	0x0190	9.5.3

Table 186. Register overview: I/O configuration (base address 0x4000 1000)

Name	Pin type	Access	Offset	Description	Reset value [1]	Section
PIO1_4	A	R/W	0x090	Pin control for port 1 pin 4. Includes an ADC input.	0x0190	9.5.3
PIO1_5	A	R/W	0x094	Pin control for port 1 pin 5. Includes an ADC input.	0x0190	9.5.3
PIO1_6	A	R/W	0x098	Pin control for port 1 pin 6. Includes an ADC input.	0x0190	9.5.3
PIO1_7	A	R/W	0x09C	Pin control for port 1 pin 7. Includes an ADC input.	0x0190	9.5.3
PIO1_8	A	R/W	0x0A0	Pin control for port 1 pin 8. Includes an ADC input.	0x0190	9.5.3
PIO1_9	D	R/W	0x0A4	Pin control for port 1 pin 9.	0x0190	9.5.1
PIO1_10	D	R/W	0x0A8	Pin control for port 1 pin 10.	0x0190	9.5.1
PIO1_11	D	R/W	0x0AC	Pin control for port 1 pin 11.	0x0190	9.5.1
PIO1_12	D	R/W	0x0B0	Pin control for port 1 pin 12.	0x0190	9.5.1
PIO1_13	D	R/W	0x0B4	Pin control for port 1 pin 13.	0x0190	9.5.1
PIO1_14	D	R/W	0x0B8	Pin control for port 1 pin 14.	0x0190	9.5.1
PIO1_15	D	R/W	0x0BC	Pin control for port 1 pin 15.	0x0190	9.5.1
PIO1_16	D	R/W	0x0C0	Pin control for port 1 pin 16.	0x0190	9.5.1
PIO1_17	D	R/W	0x0C4	Pin control for port 1 pin 17.	0x0190	9.5.1

[1] Reset Value reflects the data stored in defined bits only. Reserved bits assumed to be 0.

[2] These pins are open drain and require an external pull-up to provide output functionality. These pins have specific support for I2C drive and filtering for modes up to Fast-mode Plus.

9.5.1 Type D IOCON registers (PIO0)

This IOCON register description applies to most port pins. Other pins include ADC or I2C functions that alter the contents of the related IOCON registers.

Functions available on Type D can be found a follows:

- Port 0: see Table 188
- Port 1: see Table 189

Remark: The FUNC field for PIO0_16 and PIO0_17 resets to 0b101 (0x5), selecting the Serial Wire Debug function by default.

Table 187. Type D IOCON registers bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC	-	Selects pin function.	0x0 [1]
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	0x2
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
5	-	-	Reserved. Read value is undefined, only zero should be written.	-
6	INVERT		Input polarity.	0x0
		0	Disabled. Input function is not inverted.	
		1	Enabled. Input is function inverted.	
7	DIGIMODE		Select Analog/Digital mode.	0x1
		0	Analog mode.	
		1	Digital mode.	
8	FILTEROFF		Controls input glitch filter.	0x1
		0	Filter enabled. Noise pulses below approximately 10 ns are filtered out.	
		1	Filter disabled. No input filtering is done.	
9	SLEW		Driver slew rate.	0x0
		0	Standard mode, output slew rate control is enabled. More outputs can be switched simultaneously.	
		1	Fast mode, slew rate control is disabled. Refer to the appropriate specific device data sheet for details.	
10	OD		Controls open-drain mode.	0x0
		0	Normal. Normal push-pull output	
		1	Open-drain. Simulated open-drain output (high drive disabled).	
31:11	-	-	Reserved. Read value is undefined, only zero should be written.	-

[1] For PIO0_16 and PIO0_17, the reset value is 0x5.

Table 188. Type D I/O Control registers: FUNC values and pin functions for port 0

Regname	FUNC = 0	FUNC = 1	FUNC = 2	FUNC = 3	FUNC = 4	FUNC = 5	FUNC = 6	FUNC = 7
PIO0_0	PIO0_0	FC0_RXD_SDA_MOSI	FC3_CTS_SDA_SSEL0	CTIMER0_CAP0	-	SCT0_OUT3	-	-
PIO0_1	PIO0_1	FC0_TXD_SCL_MISO	FC3_RTS_SCL_SSEL1	CTIMER0_CAP1	-	SCT0_OUT1	-	-
PIO0_2	PIO0_2	FC0_CTS_SDA_SSEL0	FC2_SSEL3	-	-	-	-	-
PIO0_3	PIO0_3	FC0_RTS_SCL_SSEL1	FC2_SSEL2	CTIMER1_MAT3	-	-	-	-
PIO0_4	PIO0_4	FC0_SCK	FC3_SSEL2	CTIMER0_CAP2	-	-	-	-
PIO0_5	PIO0_5	FC6_RXD_SDA_MOSI_DATA	SCT0_OUT6	CTIMER0_MAT0	-	-	-	-
PIO0_6	PIO0_6	FC6_TXD_SCL_MISO_WS	-	CTIMER0_MAT1	-	UTICK_CAP0	-	-
PIO0_7	PIO0_7	FC6_SCK	SCT0_OUT0	CTIMER0_MAT2	-	CTIMER0_CAP2	-	-
PIO0_8	PIO0_8	FC2_RXD_SDA_MOSI	SCT0_OUT1	CTIMER0_MAT3	-	-	-	-
PIO0_9	PIO0_9	FC2_TXD_SCL_MISO	SCT0_OUT2	CTIMER3_CAP0	-	FC3_CTS_SDA_SSEL0	-	-
PIO0_10	PIO0_10	FC2_SCK	SCT0_OUT3	CTIMER3_MAT0	-	-	-	-
PIO0_11	PIO0_11	FC3_SCK	FC6_RXD_SDA_MOSI_DATA	-	-	-	-	-
PIO0_12	PIO0_12	FC3_RXD_SDA_MOSI	FC6_TXD_SCL_MISO_WS	-	-	-	-	-
PIO0_13	PIO0_13	FC3_TXD_SCL_MISO	SCT0_OUT4	-	-	-	-	-
PIO0_14	PIO0_14	FC3_CTS_SDA_SSEL0	SCT0_OUT5	-	-	FC1_SCK	-	-
PIO0_15	PIO0_15	FC3_RTS_SCL_SSEL1	-	-	-	FC4_SCK	-	-
PIO0_16	PIO0_16	FC3_SSEL2	FC6_CTS_SDA_SSEL0	CTIMER3_MAT1	-	SWCLK	-	-
PIO0_17	PIO0_17	FC3_SSEL3	FC6_RTS_SCL_SSEL1	CTIMER3_MAT2	-	SWDIO	-	-
PIO0_18	PIO0_18	FC5_TXD_SCL_MISO	SCT0_OUT0	CTIMER0_MAT0	-	-	-	-
PIO0_19	PIO0_19	FC5_SCK	SCT0_OUT1	CTIMER0_MAT1	-	-	-	-
PIO0_20	PIO0_20	FC5_RXD_SDA_MOSI	FC0_SCK	CTIMER3_CAP0	-	-	-	-
PIO0_21	PIO0_21	CLKOUT	FC0_TXD_SCL_MISO	CTIMER3_MAT0	-	-	-	-
PIO0_22	PIO0_22	CLKIN	FC0_RXD_SDA_MOSI	CTIMER3_MAT3	-	-	-	-

Table 189. Type D I/O Control registers: FUNC values and pin functions for port 1

Regname	FUNC = 0	FUNC = 1	FUNC = 2	FUNC = 3	FUNC = 4	FUNC = 5	FUNC = 6	FUNC = 7
PIO1_9	PIO1_9	-	FC3_RXD_SDA_MOSI	CTIMER0_CAP2	-	-	USB0_UP_LED	-
PIO1_10	PIO1_10	-	FC6_TXD_SCL_MISO_WS	SCT0_OUT4	FC1_SCK	-	-	USB0_FRAME
PIO1_11	PIO1_11	-	FC6_RTS_SCL_SSEL1	CTIMER1_CAP0	FC4_SCK	-	-	USB0_VBUS
PIO1_12	PIO1_12	-	FC5_RXD_SDA_MOSI	CTIMER1_MAT0	FC7_SCK	UTICK_CAP2	-	-
PIO1_13	PIO1_13	-	FC5_TXD_SCL_MISO	CTIMER1_MAT1	FC7_RXD_SDA_MOSI_DATA	-	-	-
PIO1_14	PIO1_14	-	FC2_RXD_SDA_MOSI	SCT0_OUT7	FC7_TXD_SCL_MISO_WS	-	-	-
PIO1_15	PIO1_15	-	SCT0_OUT5	CTIMER1_CAP3	FC7_CTS_SDA_SSEL0	-	-	-
PIO1_16	PIO1_16	-	CTIMER0_MAT0	CTIMER0_CAP0	FC7_RTS_SCL_SSEL1	-	-	-
PIO1_17	PIO1_17	-	-	-	MCLK	UTICK_CAP3	-	-

9.5.2 Type I IOCON registers (PIO0)

This IOCON table applies to pins PIO0_23 to PIO0_26. See Table 191 for recommended setting for I2C operation. See Table 192 for function available on type I pins.

Table 190. Type I IOCON registers bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC	-	Selects pin function.	0x0
4:3	-	-	Reserved. Read value is undefined, only zero should be written.	-
5	I2CSLEW		Controls slew rate of I^{2}C pin.	0x1
		0	I^{2}C mode.	
		1	GPIO mode.	
6	INVERT		Input polarity.	0x0
		0	Disabled. Input function is not inverted.	
		1	Enabled. Input is function inverted.	
7	DIGIMODE		Select Analog/Digital mode.	0x1
		0	Analog mode.	
		1	Digital mode.	
8	FILTEROFF		Controls input glitch filter.	0x1
		0	Filter enabled. Noise pulses below approximately 10 ns are filtered out.	
		1	Filter disabled. No input filtering is done.	
9	I2CDRIVE		Controls the current sink capability of the pin.	0x0
		0	Low drive. Output drive sink is 4 mA. This is sufficient for Standard and Fast-mode I^{2}C.	
		1	High drive. Output drive sink is 20 mA. This is needed for Fast Mode Plus I^{2}C. Refer to the appropriate specific device data sheet for details.	
10	I2CFILTER		Configures I^{2}C features for Standard mode, Fast-mode, and Fast-mode Plus operation.	0x0
		0	I^{2}C 50 ns glitch filter enabled.	
		1	I^{2}C 50 ns glitch filter not enabled.	
31:11	-	-	Reserved. Read value is undefined, only zero should be written.	-

Table 191. Suggested IOCON settings for I2C functions

Mode	IOCON register bit					
	10: I2CFILTER	9: I2CDRIVE	8: FILTEROFF	7: DIGIMODE	6: INVERT	5: I2CSLEW
GPIO 4 mA drive	1	0	0 [1]	1 [2]	0	1
GPIO 20 mA drive	1	1	0 [1]	1 [2]	0	1
Standard mode I^{2}C	1	0	0	1	0	0
Fast mode I^{2}C	0	0	0	1	0	0
Fast-mode Plus I^{2}C	0	1	0	1	0	0
High Speed slave I^{2}C	1	1	0	1	0	0

[1] The input filter may be turned by setting FILTEROFF to 1 if the input filter is not needed.

[2] The input may be turned off by clearing DIGIMODE if it is not needed.

Table 192. Type I I/O Control registers: FUNC values and pin functions

Regname/FUNC = 0	FUNC = 1	FUNC = 2	FUNC = 3	FUNC = 4	FUNC = 5	FUNC = 6	FUNC = 7
PIO0_23	FC1_RTS_SCL_SSEL1	-	CTIMER0_CAP0	-	UTICK_CAP1	-	-
PIO0_24	FC1_CTS_SDA_SSEL0	-	CTIMER0_CAP1	-	CTIMER0_MAT0	-	-
PIO0_25	FC4_RTS_SCL_SSEL1	FC6_CTS_SDA_SSEL0	CTIMER0_CAP2	-	CTIMER1_CAP1	-	-
PIO0_26	FC4_CTS_SDA_SSEL0	-	CTIMER0_CAP3	-	-	-	-

9.5.3 Type A IOCON registers (PIO0, PIO1)

This IOCON table applies to pins:

- PIO0_29 to PIO0_31
- PIO1_0 to PIO1_8

See Table 194 for function available on type I pins.

Table 193. Type A IOCON registers bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC	-	Selects pin function.	0x0
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	0x2
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
5	-	-	Reserved. Read value is undefined, only zero should be written.	-
6	INVERT		Input polarity.	0x0
		0	Disabled. Input function is not inverted.	
		1	Enabled. Input is function inverted.	
7	DIGIMODE		Select Analog/Digital mode.	0x1
		0	Analog mode.	
		1	Digital mode.	
8	FILTEROFF		Controls input glitch filter.	0x1
		0	Filter enabled. Noise pulses below approximately 10 ns are filtered out.	
		1	Filter disabled. No input filtering is done.	
9	-	-	Reserved. Read value is undefined, only zero should be written.	-
10	OD		Controls open-drain mode.	0x0
		0	Normal. Normal push-pull output	
		1	Open-drain. Simulated open-drain output (high drive disabled).	
31:11	-	-	Reserved. Read value is undefined, only zero should be written.	-

Table 194. Type A I/O Control registers: FUNC values and pin functions

Reg name	FUNC = 0	FUNC = 1	FUNC = 2	FUNC = 3	FUNC = 4	FUNC = 5	FUNC = 6	FUNC = 7
PIO0_29	PIO0_29/ADC0_0	FC1_RXD_SDA_MOSI	SCT0_OUT2	CTIMER0_MAT3	-	CTIMER0_CAP1	-	CTIMER0_MAT1
PIO0_30	PIO0_30/ADC0_1	FC1_TXD_SCL_MISO	SCT0_OUT3	CTIMER0_MAT2	-	CTIMER0_CAP2	-	-
PIO0_31	PIO0_31/ADC0_2	-	FC2_CTS_SDA_SSEL0	-	-	CTIMER0_CAP3	CTIMER0_MAT3	-
PIO1_0	PIO1_0/ADC0_3	-	FC2_RTS_SCL_SSEL1	CTIMER3_MAT1	-	CTIMER0_CAP0	-	-
PIO1_1	PIO1_1/ADC0_4	-	-	SCT0_OUT4	FC5_SSEL2	FC4_TXD_SCL_MISO	-	-
PIO1_2	PIO1_2/ADC0_5	MCLK	FC7_SSEL3	SCT0_OUT5	FC5_SSEL3	FC4_RXD_SDA_MOSI	-	-
PIO1_3	PIO1_3/ADC0_6	-	FC7_SSEL2	SCT0_OUT6	-	FC3_SCK	CTIMER0_CAP1	USB0_UP_LED
PIO1_4	PIO1_4/ADC0_7	-	FC7_RTS_SCL_SSEL1	SCT0_OUT7	-	FC3_TXD_SCL_MISO	CTIMER0_MAT1	-
PIO1_5	PIO1_5/ADC0_8	-	FC7_CTS_SDA_SSEL0	CTIMER1_CAP0	-	CTIMER1_MAT3	-	USB0_FRAME
PIO1_6	PIO1_6/ADC0_9	-	FC7_SCK	CTIMER1_CAP2	-	CTIMER1_MAT2	-	USB0_VBUS
PIO1_7	PIO1_7/ADC0_10	-	FC7_RXD_SDA_MOSI_DATA	CTIMER1_MAT2	-	CTIMER1_CAP2	-	-
PIO1_8	PIO1_8/ADC0_11	-	FC7_TXD_SCL_MISO_WS	CTIMER1_MAT3	-	CTIMER1_CAP3	-	-

[1] To enable an ADC input, select the GPIO function and disable the digital functions of the pin by clearing the DIGIMODE bit in the related IOCON register.

UM11071

Chapter 10: LPC51U68 Input multiplexing (INPUT MUX)

Rev. 1.1 — 17 May 2018 **User manual**

10.1 How to read this chapter

Input multiplexing is present on all LPC51U68 devices. Depending on the package, not all inputs from external pins may be available.

10.2 Features

- Configures the inputs to the pin interrupt block and pattern match engine.
- Configures the inputs to the DMA triggers.
- Configures the inputs to the frequency measure function. This function is controlled by the FREQMECTRL register in the SYSCON block.

10.3 Basic configuration

Once set up, no clocks are needed for the input multiplexer to function. The system clock is needed only to write to or read from the INPUT MUX registers. Once the input multiplexer is configured, disable the clock to the INPUT MUX block in the AHBCLKCTRL register.

10.4 Pin description

The input multiplexer has no dedicated pins. However, all digital pins of ports 0 and 1 can be selected as inputs to the pin interrupts. Multiplexer inputs from external pins work independently of any other function assigned to the pin as long as no analog function is enabled.

Table 195. INPUT MUX pin description

Pins	Peripheral	Section
Any existing pin on port 0 or 1	Pin interrupts 0 to 3	10.6.1
PIO0_4, PIO0_20, PIO0_24, PIO1_4	Frequency measure block	10.6.4

10.5 General description

The inputs to the DMA triggers, to the eight pin interrupts, and to the frequency measure block are multiplexed to multiple input sources. The sources can be external pins, interrupts, or output signals of other peripherals.

The input multiplexing makes it possible to design event-driven processes without CPU intervention by connecting peripherals like the ADC.

The DMA can use trigger input multiplexing to sequence DMA transactions without the use of interrupt service routines.

10.5.1 Pin interrupt input multiplexing

The input mux for the pin interrupts and pattern match engine multiplexes all existing pins from ports 0 and 1.

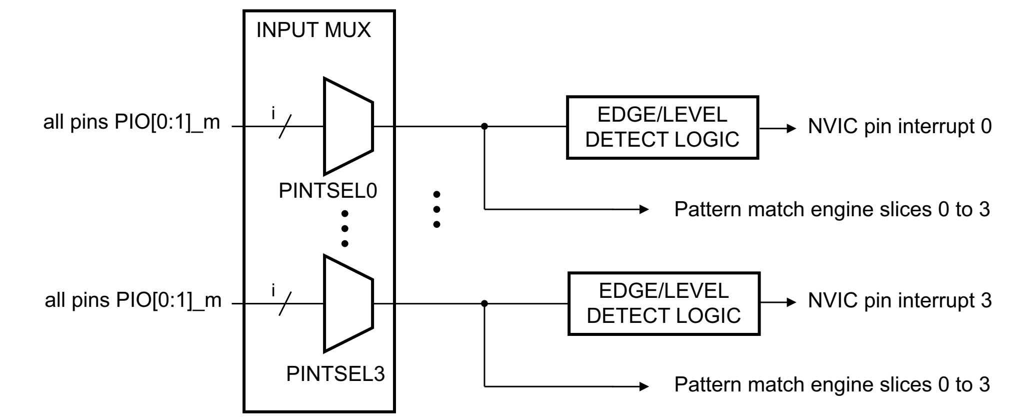


Fig 15. Pin interrupt multiplexing

10.5.2 DMA trigger input multiplexing

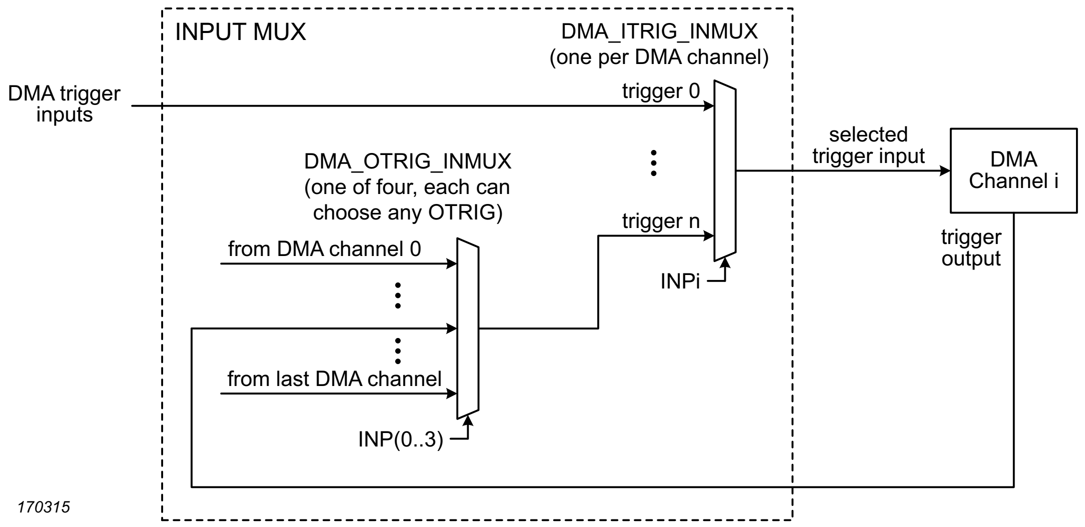


Fig 16. DMA trigger multiplexing

10.6 Register description

All input mux registers reside on word address boundaries. Details of the registers appear in the description of each function.

All address offsets not shown in Table 196 are reserved and should not be written to.

Table 196. Register overview: Input multiplexing (base address 0x4000 5000)

Name	Access	Offset	Description	Reset value	Section
PINTSEL0	R/W	0x0C0	Pin interrupt select register 0	0x0	10.6.1
PINTSEL1	R/W	0x0C4	Pin interrupt select register 1	0x0	10.6.1
PINTSEL2	R/W	0x0C8	Pin interrupt select register 2	0x0	10.6.1
PINTSEL3	R/W	0x0CC	Pin interrupt select register 3	0x0	10.6.1
DMA_ITRIG_INMUX0	R/W	0x0E0	Trigger select register for DMA channel 0	0x1F	10.6.2
DMA_ITRIG_INMUX1	R/W	0x0E4	Trigger select register for DMA channel 1	0x1F	10.6.2
DMA_ITRIG_INMUX2	R/W	0x0E8	Trigger select register for DMA channel 2	0x1F	10.6.2
DMA_ITRIG_INMUX3	R/W	0x0EC	Trigger select register for DMA channel 3	0x1F	10.6.2
DMA_ITRIG_INMUX4	R/W	0x0F0	Trigger select register for DMA channel 4	0x1F	10.6.2
DMA_ITRIG_INMUX5	R/W	0x0F4	Trigger select register for DMA channel 5	0x1F	10.6.2
-	-	0x0F8	Reserved	-	-
-	-	0x0FC	Reserved	-	-
DMA_ITRIG_INMUX8	R/W	0x100	Trigger select register for DMA channel 8	0x1F	10.6.2
DMA_ITRIG_INMUX9	R/W	0x104	Trigger select register for DMA channel 9	0x1F	10.6.2
-	-	0x108	Reserved	-	-
-	-	0x10C	Reserved	-	-
DMA_ITRIG_INMUX12	R/W	0x110	Trigger select register for DMA channel 12	0x1F	10.6.2
DMA_ITRIG_INMUX13	R/W	0x114	Trigger select register for DMA channel 13	0x1F	10.6.2
DMA_ITRIG_ INMUX14	R/W	0x118	Trigger select register for DMA channel 14	0x1F	10.6.2
DMA_ITRIG_INMUX15	R/W	0x11C	Trigger select register for DMA channel 15	0x1F	10.6.2
DMA_ITRIG_INMUX16	R/W	0x120	Trigger select register for DMA channel 16	0x1F	10.6.2
DMA_ITRIG_INMUX17	R/W	0x124	Trigger select register for DMA channel 17	0x1F	10.6.2
DMA_ITRIG_INMUX18	R/W	0x128	Trigger select register for DMA channel 18	0x1F	10.6.2
DMA_ITRIG_INMUX19	R/W	0x12C	Trigger select register for DMA channel 19	0x1F	10.6.2
-	-	0x130	Reserved	-	-
-	-	0x134	Reserved	-	-
DMA_OTRIG_INMUX0	R/W	0x160	DMA output trigger selection to become DMA trigger 16	0x1F	10.6.3
DMA_OTRIG_INMUX1	R/W	0x164	DMA output trigger selection to become DMA trigger 17	0x1F	10.6.3
DMA_OTRIG_INMUX2	R/W	0x168	DMA output trigger selection to become DMA trigger 18	0x1F	10.6.3
DMA_OTRIG_INMUX3	R/W	0x16C	DMA output trigger selection to become DMA trigger 19	0x1F	10.6.3
FREQMEAS_REF	R/W	0x180	Selection for frequency measurement reference clock	0x1F	10.6.4
FREQMEAS_TARGET	R/W	0x184	Selection for frequency measurement target clock	0x1F	10.6.5

10.6.1 Pin interrupt select registers

Each of these four registers selects one pin from among ports 0 and 1 as the source of a pin interrupt or as the input to the pattern match engine. To select a pin for any of the four pin interrupts or pattern match engine inputs, write the GPIO port pin number as 0 to 26 for pins PIO0_0 to PIO0_26 to the INTPIN bit, and 29 to 31 for pins PIO0_29 to PIO0_31 to the INTPIN bits. Port 1 pins correspond to pin numbers 32 to 63. For example, setting INTPIN to 0x5 in PINTSEL0 selects pin PIO0_5 for pin interrupt 0. To determine the GPIO port pin number for a given device package, see the pin description table in the data sheet.

Each of the pin interrupts must be enabled in the NVIC (see Table 79) before it becomes active.

To use the selected pins for pin interrupts or the pattern match engine, see Section 12.5.2 "Pattern match engine".

Table 197. Pin interrupt select registers (PINTSEL[0:3], offsets [0x0C0:0x0CC]) bit description

Bit	Symbol	Description	Reset value
7:0	INTPIN	Pin number select for pin interrupt or pattern match engine input. (PIO0_0 to PIO1_17 correspond to numbers 0 to 63).	0x0
31:8	-	Reserved	-

10.6.2 DMA trigger input mux registers 0 to 17

With the DMA trigger input mux registers, one trigger input can be selected for each of the DMA channels from the potential internal sources. By default, none of the triggers are selected.

Table 198. DMA trigger Input mux registers (DMA_ITRIG_INMUX[0:21], offsets [0x0E0:0x134]) bit description

Bit	Symbol	Description	Reset value
4:0	INP	Trigger input number (decimal value) for DMA channel n (n = 0 to 19). 0 = ADC0 Sequence A interrupt 1 = ADC0 Sequence B interrupt 2 = SCT0 DMA request 0 3 = SCT0 DMA request 1 4 = Timer CTIMER0 Match 0 5 = Timer CTIMER0 Match 1 6 = Timer CTIMER1 Match 0 7 = RESERVED 8 = RESERVED 9 = Timer CTIMER3 Match 0 10 = RESERVED 11 = RESERVED 12 = Pin interrupt 0 13 = Pin interrupt 1 14 = Pin interrupt 2 15 = Pin interrupt 3 16 = DMA output trigger mux 0 17 = DMA output trigger mux 1 18 = DMA output trigger mux 2 19 = DMA output trigger mux 3	0x1F
31:5	-	Reserved.	-

10.6.3 DMA output trigger feedback mux registers 0 to 3

This register provides a multiplexer for inputs 16 to 19 of each DMA trigger input mux register DMA_ITRIG_INMUX. These inputs can be selected from among the trigger outputs generated by the each DMA channel. By default, none of the triggers are selected.

Table 199. DMA output trigger feedback mux registers (DMA_OTRIG_INMUX[0:3], offset [0x160:0x16C]) bit description

Bit	Symbol	Description	Reset value
4:0	INP	DMA trigger output number (decimal value) for DMA channel n (n = 0 to 19).	0x1F
31:5	-	Reserved.	-

10.6.4 Frequency measure function reference clock select register

This register selects a clock for the reference clock of the frequency measure function. By default, no clock is selected.

Also see:

- Section 6.2.3 “Measure the frequency of a clock signal”
- Section 6.6.6 “Frequency measure function”
- Frequency measure control register (FREQMECTRL) - Section 6.5.41
- Frequency target clock select register (FREQMEAS_TARGET) - Section 10.6.5

Table 200. Frequency measure function frequency clock select register (FREQMEAS_REF, offset 0x180) bit description

Bit	Symbol	Description	Reset value
4:0	CLKIN	Clock source number (decimal value) for frequency measure function target clock: 0 = CLK_IN 1 = FRO 12 MHz (fro_12m) 2 = Watchdog oscillator 3 = 32 kHz RTC oscillator 4 = Main clock (see Section 6.5.23) 5 = PIO0_4 6 = PIO0_20 7 = PIO0_24 8 = PIO1_4 0x1F = none Other settings are undefined.	0x1F
31:5	-	Reserved.	-

10.6.5 Frequency measure function target clock select register

This register selects a clock for the target clock of the frequency measure function. By default, no clock is selected.

Also see:

- Section 6.2.3 “Measure the frequency of a clock signal”
- Section 6.6.6 “Frequency measure function”
- Frequency measure control register (FREQMECTRL) - Section 6.5.41

- Frequency target clock select register (FREQMEAS_REF) - Section 10.6.4

Table 201. Frequency measure function target clock select register (FREQMEAS_TARGET, offset 0x184) bit description

Bit	Symbol	Description	Reset value
4:0	CLKIN	Clock source number (decimal value) for frequency measure function target clock: 0 = CLK_IN 1 = FRO 12 MHz (fro_12m) 2 = Watchdog oscillator 3 = 32 kHz RTC oscillator 4 = Main clock (see Section 6.5.23) 5 = PIO0_4 6 = PIO0_20 7 = PIO0_24 8 = PIO1_4 0x1F = none Other settings are undefined.	0x1F
31:5	-	Reserved.	-

UM11071

Chapter 11: LPC51U68 General Purpose I/O (GPIO)

Rev. 1.1 — 17 May 2018 **User manual**

11.1 How to read this chapter

GPIO registers support up to 32 pins on each port. Depending on the device and package type, a subset of those pins may be available, and the unused bits in GPIO registers are reserved (see Table 185).

11.2 Basic configuration

For the GPIO port registers, enable the clock to each GPIO port in the AHBCLKCTRL0 register (Table 115).

11.3 Features

- GPIO pins can be configured as input or output by software.
- GPIO pins other than PIO0_16 and PIO0_17 default to inputs with interrupt disabled at reset. PIO0_16 and PIO0_17 default to serial wire debug functions at reset.
- Pin registers allow pins to be sensed and set individually.
- Direction (input/output) can be set and cleared individually.

11.4 General description

The GPIO pins can be used in several ways to set pins as inputs or outputs and use the inputs as combinations of level and edge sensitive interrupts.

The GPIOs can be used as external interrupts together with the pin interrupt and group interrupt blocks, see Chapter 12 and Chapter 13.

The GPIO port registers configure each GPIO pin as input or output and read the state of each pin if the pin is configured as input or set the state of each pin if the pin is configured as output.

11.5 Register description

Note: In all GPIO registers, bits that are not shown are **reserved**.

GPIO port addresses can be read and written as bytes, halfwords, or words.

Remark: A reset value noted as “ext” in this table and subsequent tables indicates that the data read after reset depends on the state of the pin, which in turn may depend on an external source.

Table 202. Register overview: GPIO port (base address 0x4008 C000)

Name	Access	Offset	Description	Reset value	Width (bits)	Section
B[0:49]	R/W	[0x0000:0x0031]	Byte pin registers for all port 0 and 1 GPIO pins	ext	8	11.5.1
W[0:49]	R/W	[0x1000:0x10C4]	Word pin registers for all port 0 and 1 GPIO pins	ext	32	11.5.2
DIR0	R/W	0x2000	Direction registers port 0	0x0	32	11.5.3
DIR1	R/W	0x2004	Direction registers port 1	0x0	32	11.5.3
MASK0	R/W	0x2080	Mask register port 0	0x0	32	11.5.4
MASK1	R/W	0x2084	Mask register port 1	0x0	32	11.5.4
PIN0	R/W	0x2100	Port pin register port 0	ext	32	11.5.5
PIN1	R/W	0x2104	Port pin register port 1	ext	32	11.5.5
MPIN0	R/W	0x2180	Masked port register port 0	ext	32	11.5.6
MPIN1	R/W	0x2184	Masked port register port 1	ext	32	11.5.6
SET0	R/W	0x2200	Write: Set register for port 0 Read: output bits for port 0	0x0	32	11.5.7
SET1	R/W	0x2204	Write: Set register for port 1 Read: output bits for port 1	0x0	32	11.5.7
CLR0	WO	0x2280	Clear port 0	-	32	11.5.8
CLR1	WO	0x2284	Clear port 1	-	32	11.5.8
NOT0	WO	0x2300	Toggle port 0	-	32	11.5.9
NOT1	WO	0x2304	Toggle port 1	-	32	11.5.9
DIRSET0	WO	0x2380	Set pin direction bits for port 0	0x0	32	11.5.10
DIRSET1	WO	0x2384	Set pin direction bits for port 1	0x0	32	11.5.10
DIRCLR0	WO	0x2400	Clear pin direction bits for port 0	-	32	11.5.11
DIRCLR1	WO	0x2404	Clear pin direction bits for port 1	-	32	11.5.11
DIRNOT0	WO	0x2480	Toggle pin direction bits for port 0	-	32	11.5.12
DIRNOT1	WO	0x2484	Toggle pin direction bits for port 1	-	32	11.5.12

11.5.1 GPIO port byte pin registers

Each GPIO pin has a byte register in this address range. Software typically reads and writes bytes to access individual pins, but can read or write halfwords to sense or set the state of two pins, and read or write words to sense or set the state of four pins.

Table 203. GPIO port byte pin registers (B[0:49], offset [0x0000:0x0031]) bit description

Bit	Symbol	Description	Reset value	Access
0	PBYTE	**Read:** 0 when pin PIOm_n is LOW; 0xFF when pin PIOm_n is HIGH. The read reflects the state of the pin PIOm_n, regardless of direction, masking, or alternate function, except that pins configured as analog I/O always read as 0. **Write:** 0 clears the output bit; any value from 0x01 to 0xFF sets the output bit. **Remark:** One register for each port pin. Supported pins depends on the specific device and package.	ext	R/W
7:1	-	Reserved (0 on read, ignored on write)	0x0	-

11.5.2 GPIO port word pin registers

Each GPIO pin has a word register in this address range. Any byte, halfword, or word read in this range will be all zeros if the pin is low or all ones if the pin is high, regardless of direction, masking, or alternate function, except that pins configured as analog I/O always read as zeros. Any write will clear the pin’s output bit if the value written is all zeros, else it will set the pin’s output bit.

Table 204. GPIO port word pin registers (W[0:49], offsets [0x1000:0x10C4]) bit description

Bit	Symbol	Description	Reset value	Access
31:0	PWORD	**Read:** 0 when pin PIOm_n is LOW; 0xFFFF FFFF when pin PIOm_n is HIGH. The read reflects the state of the pin PIOm_n, regardless of direction, masking, or alternate function, except that pins configured as analog I/O always read as 0. **Write:** 0 clears the output bit; any value from 0x0000 0001 to 0xFFFF FFFF sets the output bit. **Remark:** One register for each port pin. Supported pins depends on the specific device and package.	ext	R/W

11.5.3 GPIO port direction registers

Each GPIO port has one direction register for configuring the port pins as inputs or outputs.

Table 205. GPIO direction port register (DIR[0:1], offset [0x2000:0x2004]) bit description

Bit	Symbol	Description	Reset value	Access
31:0	DIRP	Selects pin direction for pin PIOm_n (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Supported pins depends on the specific device and package. 0 = input. 1 = output.	0x0	R/W

11.5.4 GPIO port mask registers

These registers affect writing and reading the MPORT registers. Zeroes in these registers enable reading and writing; ones disable writing and result in zeros in corresponding positions when reading.

Table 206. GPIO mask port register (MASK[0:1], offset [0x2080:0x2084]) bit description

Bit	Symbol	Description	Reset value	Access
31:0	MASKP	Controls which bits corresponding to PIOm_n are active in the MPORT register (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Supported pins depends on the specific device and package. 0 = Read MPORT: pin state; write MPORT: load output bit. 1 = Read MPORT: 0; write MPORT: output bit not affected.	0x0	R/W

11.5.5 GPIO port pin registers

Reading these registers returns the current state of the pins read, regardless of direction, masking, or alternate functions, except that pins configured as analog I/O always read as 0s. Writing these registers loads the output bits of the pins written to, regardless of the Mask register.

Table 207. GPIO port pin register (PIN[0:1], offset [0x2100:0x2104]) bit description

Bit	Symbol	Description	Reset value	Access
31:0	PORT	Reads pin states or loads output bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Supported pins depends on the specific device and package. 0 = Read: pin is low; write: clear output bit. 1 = Read: pin is high; write: set output bit.	ext	R/W

11.5.6 GPIO masked port pin registers

These registers are similar to the PORT registers, except that the value read is masked by ANDing with the inverted contents of the corresponding MASK register, and writing to one of these registers only affects output register bits that are enabled by zeros in the corresponding MASK register

Table 208. GPIO masked port pin register (MPIN[0:1], offset [0x2180:0x2184]) bit description

Bit	Symbol	Description	Reset value	Access
31:0	MPORTP	Masked port register (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Supported pins depends on the specific device and package. 0 = Read: pin is LOW and/or the corresponding bit in the MASK register is 1; write: clear output bit if the corresponding bit in the MASK register is 0. 1 = Read: pin is HIGH and the corresponding bit in the MASK register is 0; write: set output bit if the corresponding bit in the MASK register is 0.	ext	R/W

11.5.7 GPIO port set registers

Output bits can be set by writing ones to these registers, regardless of MASK registers. Reading from these register returns the port's output bits, regardless of pin directions.

Table 209. GPIO set port register (SET[0:1], offset [0x2200:0x2204]) bit description

Bit	Symbol	Description	Reset value	Access
31:0	SETP	Read or set output bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Supported pins depends on the specific device and package. 0 = Read: output bit: write: no operation. 1 = Read: output bit; write: set output bit.	0x0	R/W

11.5.8 GPIO port clear registers

Output bits can be cleared by writing ones to these write-only registers, regardless of MASK registers.

Table 210. GPIO clear port register (CLR[0:1], offset [0x2280:0x2284]) bit description

Bit	Symbol	Description	Reset value	Access
31:0	CLRP	Clear output bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Supported pins depends on the specific device and package. 0 = No operation. 1 = Clear output bit.	-	WO

11.5.9 GPIO port toggle registers

Output bits can be toggled/inverted/complemented by writing ones to these write-only registers, regardless of MASK registers.

Table 211. GPIO toggle port register (NOT[0:1], offset [0x2300:0x2304]) bit description

Bit	Symbol	Description	Reset value	Access
31:0	NOTP	Toggle output bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Supported pins depends on the specific device and package. 0 = no operation. 1 = Toggle output bit.	-	WO

11.5.10 GPIO port direction set registers

Direction bits can be set by writing ones to these registers.

Table 212. GPIO port direction set register (DIRSET[0:1], offset 0x2380:0x2384) bit description

Bit	Symbol	Description	Reset value	Access
31:0	DIRSETP	Set direction bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Supported pins depends on the specific device and package. 0 = No operation. 1 = Set direction bit.	0x0	WO

11.5.11 GPIO port direction clear registers

Direction bits can be cleared by writing ones to these write-only registers.

Table 213. GPIO port direction clear register (DIRCLR[0:1], offset 0x2400:0x2404) bit description

Bit	Symbol	Description	Reset value	Access
31:0	DIRCLRP	Clear direction bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Supported pins depends on the specific device and package. 0 = No operation. 1 = Clear direction bit.	-	WO

11.5.12 GPIO port direction toggle registers

Direction bits can be set by writing ones to these write-only registers.

Table 214. GPIO port direction toggle register (DIRNOT[0:1], offset 0x2480:0x2484) bit description

Bit	Symbol	Description	Reset value	Access
31:0	DIRNOTP	Toggle direction bits (bit 0 = PIOn_0, bit 1 = PIOn_1, etc.). Supported pins depends on the specific device and package. 0 = no operation. 1 = Toggle direction bit.	-	WO

11.6 Functional description

11.6.1 Reading pin state

Software can read the state of all GPIO pins except those selected for analog input or output in the “I/O Configuration” logic. A pin does not have to be selected for GPIO in “I/O Configuration” in order to read its state. There are four ways to read pin state:

- The state of a single pin can be read with 7 high-order zeros from a Byte Pin register.
- The state of a single pin can be read in all bits of a byte, halfword, or word from a Word Pin register.
- The state of multiple pins in a port can be read as a byte, halfword, or word from a PORT register.
- The state of a selected subset of the pins in a port can be read from a Masked Port (MPORT) register. Pins having a 1 in the port’s Mask register will read as 0 from its MPORT register.

11.6.2 GPIO output

Each GPIO pin has an output bit in the GPIO block. These output bits are the targets of write operations to the pins. Two conditions must be met in order for a pin’s output bit to be driven onto the pin:

1. The pin must be selected for GPIO operation via IOCON (this is the default), and
2. the pin must be selected for output by a 1 in its port’s DIR register.

If either or both of these conditions is (are) not met, writing to the pin has no effect.

There are seven ways to change GPIO output bits:

- Writing to a Byte Pin register loads the output bit from the least significant bit.
- Writing to a Word Pin register loads the output bit with the OR of all of the bits written. (This feature follows the definition of truth of a multi-bit value in programming languages.)
- Writing to a port’s PORT register loads the output bits of all the pins written to.
- Writing to a port’s MPORT register loads the output bits of pins identified by zeros in corresponding positions of the port’s MASK register.
- Writing ones to a port’s SET register sets output bits.
- Writing ones to a port’s CLR register clears output bits.

- Writing ones to a port's NOT register toggles/complements/inverts output bits.

The state of a port's output bits can be read from its SET register. Reading any of the registers described in 11.6.1 returns the state of pins, regardless of their direction or alternate functions.

11.6.3 Masked I/O

A port's MASK register defines which of its pins should be accessible in its MPORT register. Zeroes in MASK enable the corresponding pins to be read from and written to MPORT. Ones in MASK force a pin to read as 0 and its output bit to be unaffected by writes to MPORT. When a port's MASK register contains all zeros, its PORT and MPORT registers operate identically for reading and writing.

Applications in which interrupts can result in Masked GPIO operation, or in task switching among tasks that do Masked GPIO operation, must treat code that uses the Mask register as a protected/restricted region. This can be done by interrupt disabling or by using a semaphore.

The simpler way to protect a block of code that uses a MASK register is to disable interrupts before setting the MASK register, and re-enable them after the last operation that uses the MPORT or MASK register.

More efficiently, software can dedicate a semaphore to the MASK registers, and set/capture the semaphore controlling exclusive use of the MASK registers before setting the MASK registers, and release the semaphore after the last operation that uses the MPORT or MASK registers.

11.6.4 GPIO direction

Each pin in a GPIO port can be configured as input or output using the DIR registers. The direction of individual pins can be set, cleared, or toggled using the DIRSET, DIRCLR, and DIRNOT registers.

11.6.5 Recommended practices

The following lists some recommended uses for using the GPIO port registers:

- For initial setup after Reset or re-initialization, write the PORT registers.
- To change the state of one pin, write a Byte Pin or Word Pin register.
- To change the state of multiple pins at a time, write the SET and/or CLR registers.
- To change the state of multiple pins in a tightly controlled environment like a software state machine, consider using the NOT register. This can require less write operations than SET and CLR.
- To read the state of one pin, read a Byte Pin or Word Pin register.
- To make a decision based on multiple pins, read and mask a PORT register.

UM11071

Chapter 12: LPC51U68 Pin interrupt and pattern match (PINT)

Rev. 1.1 — 17 May 2018 **User manual**

12.1 How to read this chapter

The pin interrupt generator and the pattern match engine are available on all LPC51U68 parts.

12.2 Features

- Pin interrupts
  - Up to eight pins can be selected from all GPIO pins on ports 0 and 1 as edge- or level-sensitive interrupt requests. Each request creates a separate interrupt in the NVIC.
  - Edge-sensitive interrupt pins can interrupt on rising or falling edges or both.
  - Level-sensitive interrupt pins can be HIGH- or LOW-active.
- Pattern match engine
  - Up to 8 pins can be selected from all digital pins on ports 0 and 1 to contribute to a boolean expression. The boolean expression consists of specified levels and/or transitions on various combinations of these pins.
  - Each bit slice minterm (product term) comprising the specified boolean expression can generate its own, dedicated interrupt request.
  - Any occurrence of a pattern match can be programmed to also generate an RXEV notification to the CPU.
  - Pattern match can be used, in conjunction with software, to create complex state machines based on pin inputs.

12.3 Basic configuration

- Pin interrupts:
  - Select up to eight external interrupt pins from all digital port pins on ports 0 and 1 in the Input Mux block (Table 197). The pin selection process is the same for pin interrupts and the pattern match engine. The two features are mutually exclusive.

Enable the clock to the pin interrupt register block in the AHBCLKCTRL0 register (Table 115).

  - In order to use the pin interrupts to wake up the part from deep-sleep mode, enable the pin interrupt wake-up feature in the STARTER0 register for pin interrupt 0 through 3 (Table 155).
  - Each selected pin interrupt is assigned to one interrupt in the NVIC (pin interrupts 0 to 3 in one group, see Table 79).
- Pattern match engine:
  - Select up to eight external pins from all digital port pins on ports 0 and 1 in the Input mux block (Table 197). The pin selection process is the same for pin interrupts and the pattern match engine. The two features are mutually exclusive.

Enable the clock to the pin interrupt register block in the AHBCLKCTRL0 register (Table 115).

– Each bit slice of the pattern match engine is assigned to one interrupt in the NVIC (pin interrupts 0 to 3 in one group, pin interrupts 4 through 7 in another group, see Table 79). Note that only the first 4 are available to the Cortex M0+, which is present on selected devices.

12.3.1 Configure pins as pin interrupts or as inputs to the pattern match engine

Follow these steps to configure pins as pin interrupts:

1. Determine the pins that serve as pin interrupts on the LPC51U68 package. See the data sheet for determining the GPIO port pin number associated with the package pin.
2. For each pin interrupt, program the GPIO port pin number from ports 0 and 1 into one of the eight PINTSEL registers in the Input mux block.

   Remark: The port pin number serves to identify the pin to the PINTSEL register. Any function, including GPIO, can be assigned to this pin via IOCON.
3. Enable each pin interrupt in the NVIC.

Once the pin interrupts or pattern match inputs are configured, the pin interrupt detection levels or the pattern match boolean expression can set up.

See Section 10.6.1 “Pin interrupt select registers” in the Input mux block for the PINTSEL registers.

Remark: The inputs to the Pin interrupt select registers bypass the IOCON function selection. They do not have to be selected as GPIO in IOCON. Make sure that no analog function is selected on pins that are input to the pin interrupts.

12.4 Pin description

The inputs to the pin interrupt and pattern match engine are determined by the pin interrupt select registers in the Input mux. See Section 10.6.1 “Pin interrupt select registers”.

12.5 General description

Pins with configurable functions can serve as external interrupts or inputs to the pattern match engine. Up to eight pins can be configured using the PINTSEL registers in the Input mux block for these features.

12.5.1 Pin interrupts

From all available GPIO pins, up to eight pins can be selected in the system control block to serve as external interrupt pins (see Chapter 10 “LPC51U68 Input multiplexing (INPUT MUX)”). The external interrupt pins are connected to eight individual interrupts in the NVIC and are created based on rising or falling edges or on the input level on the pin.

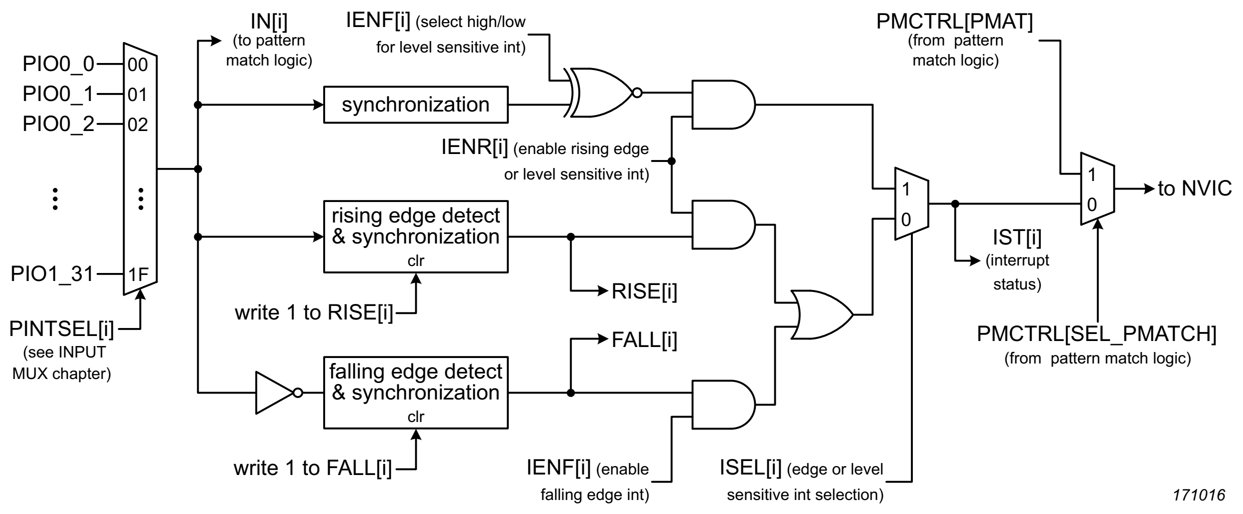

PIO0_0
PIO0_1
PIO0_2
PIO1_31
00
01
02
1F
PINTSEL[i] (see INPUT MUX chapter)
IN[i] (to pattern match logic)
IENF[i] (select high/low for level sensitive int)
synchronization
IENR[i] (enable rising edge or level sensitive int)
rising edge detect & synchronization
clr
write 1 to RISE[i]
RISE[i]
FALL[i]
falling edge detect & synchronization
clr
write 1 to FALL[i]
IENF[i] (enable falling edge int)
ISEL[i] (edge or level sensitive int selection)
PMCTRL[PMAT] (from pattern match logic)
1
0
IST[i] (interrupt status)
to NVIC
PMCTRL[SEL_PMATCH] (from pattern match logic)
171016

Fig 17. Pattern match bit slice with detect logic

12.5.2 Pattern match engine

The pattern match feature allows complex boolean expressions to be constructed from the same set of eight GPIO pins that were selected for the GPIO pin interrupts. Each term in the boolean expression is implemented as one slice of the pattern match engine. A slice consists of an input selector and a detect logic that monitors the selected input continuously and creates a HIGH output if the input qualifies as detected, that is as true. Several terms can be combined to a minterm and a pin interrupt is asserted when the minterm evaluates as true.

The detect logic of each slice can detect the following events on the selected input:

- Edge with memory (sticky): A rising edge, a falling edge, or a rising or falling edge that is detected at any time after the edge-detection mechanism has been cleared. The input qualifies as detected (the detect logic output remains HIGH) until the pattern match engine detect logic is cleared again.
- Event (non-sticky): Every time an edge (rising or falling) is detected, the detect logic output for this pin goes HIGH. This bit is cleared after one clock cycle, and the detect logic can detect another edge,
- Level: A HIGH or LOW level on the selected input.

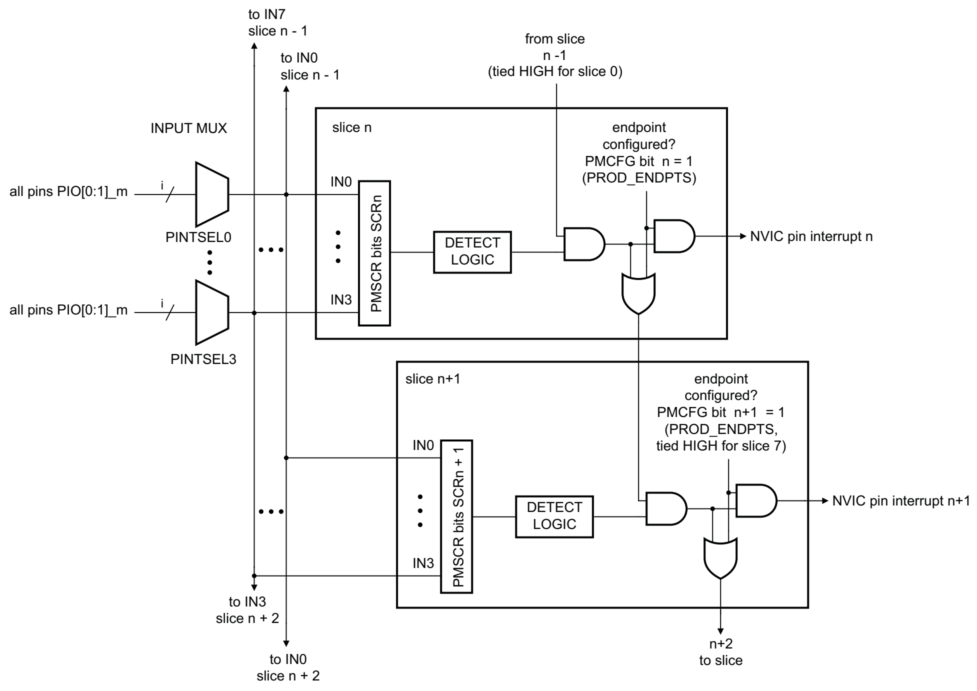
Figure 19 shows the details of the edge detection logic for each slice.

Sticky events can be combined with non-sticky events to create a pin interrupt whenever a rising or falling edge occurs after a qualifying edge event.

A time window can be created during which rising or falling edges can create a pin interrupt by combining a level detect with an event detect. See Section 12.7.3 for details.

The connections between the pins and the pattern match engine are shown in Figure 18. All pins that are inputs to the pattern match engine are selected in the Syscon block and can be GPIO port pins or other pin function depending on the IOCON configuration.

Remark: The pattern match feature requires clocks in order to operate, and can thus not generate an interrupt or wake up the device during reduced power modes below sleep mode.



See Figure 19 for the detect logic block.

Fig 18. Pattern match engine connections

The pattern match logic continuously monitors the eight inputs and generates interrupts when any one or more minterms (product terms) of the specified boolean expression is matched. A separate interrupt request is generated for each individual minterm.

In addition, the pattern match module can be enabled to generate a Receive Event (RXEV) output to the ARM core when the entire boolean expression is true (i.e. when any minterm is matched).

The pattern match function utilizes the same eight interrupt request lines as the pin interrupts so these two features are mutually exclusive as far as interrupt generation is concerned. A control bit is provided to select whether interrupt requests are generated in response to the standard pin interrupts or to pattern matches. Note that, if the pin interrupts are selected, the RXEV request to the CPU can still be enabled for pattern matches.

Remark: Pattern matching cannot be used to wake the part up from deep-sleep mode. Pin interrupts must be selected in order to use the GPIO for wake-up.

The pattern match module is constructed of eight bit-slice elements. Each bit slice is programmed to represent one component of one minterm (product term) within the boolean expression. The interrupt request associated with the last bit slice for a particular minterm will be asserted whenever that minterm is matched.
(See bit slice drawing Figure 19).

The pattern match capability can be used to create complex software state machines. Each minterm (and its corresponding individual interrupt) represents a different transition event to a new state. Software can then establish the new set of conditions (that is a new boolean expression) that will cause a transition out of the current state.

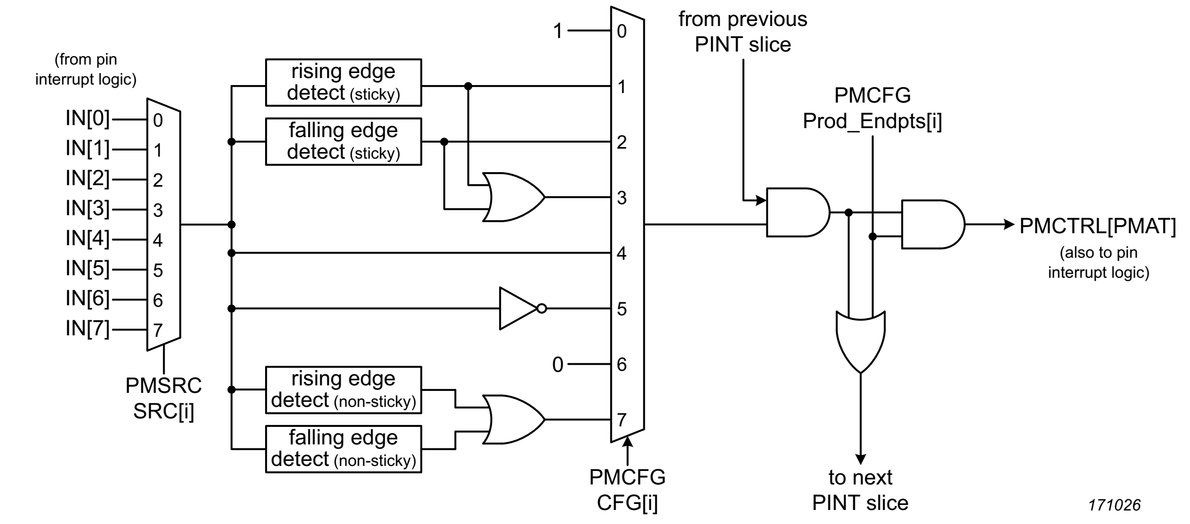

(from pin interrupt logic)
IN[0]
IN[1]
IN[2]
IN[3]
IN[4]
IN[5]
IN[6]
IN[7]
0
1
2
3
4
5
6
7
PMSRC SRC[i]
rising edge detect (sticky)
falling edge detect (sticky)
rising edge detect (non-sticky)
falling edge detect (non-sticky)
1
0
PMCFG CFG[i]
from previous PINT slice
PMCFG Prod_Endpts[i]
PMCTRL[PMAT]
(also to pin interrupt logic)
to next PINT slice
171026

Fig 19. Pattern match bit slice with detect logic

12.5.2.1 Example

Assume the expression: (IN0)~(IN1)(IN3)^ + (IN1)(IN2) + (IN0)~(IN3)~(IN4) is specified through the registers PMSRC (Table 227) and PMCFG (Table 228). Each term in the boolean expression, (IN0), ~(IN1), (IN3)^, etc., represents one bit slice of the pattern match engine.

- In the first minterm (IN0)~(IN1)(IN3)^, bit slice 0 monitors for a high-level on input (IN0), bit slice 1 monitors for a low level on input (IN1) and bit slice 2 monitors for a rising-edge on input (IN3). If this combination is detected, that is if all three terms are true, the interrupt associated with bit slice 2 will be asserted.
- In the second minterm (IN1)(IN2), bit slice 3 monitors input (IN1) for a high level, bit slice 4 monitors input (IN2) for a high level. If this combination is detected, the interrupt associated with bit slice 4 will be asserted.

- In the third minterm (IN0)~(IN3)~(IN4), bit slice 5 monitors input (IN0) for a high level, bit slice 6 monitors input (IN3) for a low level, and bit slice 7 monitors input (IN4) for a low level. If this combination is detected, the interrupt associated with bit slice 7 will be asserted.
- The ORed result of all three minterms asserts the RXEV request to the CPU. That is, if any of the three terms are true, the output is asserted.

Related links:

Section 12.7.2

12.6 Register description

Table 215. Register overview: Pin interrupts/pattern match engine (base address 0x4000 4000)

Name	Access	Offset	Description	Reset value	Section
ISEL	R/W	0x000	Pin Interrupt Mode register	0x0	12.6.1
IENR	R/W	0x004	Pin interrupt level or rising edge interrupt enable register	0x0	12.6.2
SIENR	WO	0x008	Pin interrupt level or rising edge interrupt enable set register	-	12.6.3
CIENR	WO	0x00C	Pin interrupt level or rising edge interrupt enable clear register	-	12.6.4
IENF	R/W	0x010	Pin interrupt active level or falling edge interrupt enable register	0x0	12.6.5
SIENF	WO	0x014	Pin interrupt active level or falling edge interrupt set register	-	12.6.6
CIENF	WO	0x018	Pin interrupt active level or falling edge interrupt clear register	-	12.6.7
RISE	R/W	0x01C	Pin interrupt rising edge register	0x0	12.6.8
FALL	R/W	0x020	Pin interrupt falling edge register	0x0	12.6.9
IST	R/W	0x024	Pin interrupt status register	0x0	12.6.10
PMCTRL	R/W	0x028	Pattern match interrupt control register	0x0	12.6.11
PMSRC	R/W	0x02C	Pattern match interrupt bit-slice source register	0x0	12.6.12
PMCFG	R/W	0x030	Pattern match interrupt bit slice configuration register	0x0	12.6.13

12.6.1 Pin interrupt mode register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Table 197), one bit in the ISEL register determines whether the interrupt is edge or level sensitive.

Table 216. Pin interrupt mode register (ISEL, offset 0x000) bit description

Bit	Symbol	Description	Reset value	Access
7:0	PMODE	Selects the interrupt mode for each pin interrupt. Bit n configures the pin interrupt selected in PINTSELn. 0 = Edge sensitive 1 = Level sensitive	0x0	R/W
31:8	-	Reserved. Read value is undefined, only zero should be written.	-	-

12.6.2 Pin interrupt level or rising edge interrupt enable register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Table 197), one bit in the IENR register enables the interrupt depending on the pin interrupt mode configured in the ISEL register:

- If the pin interrupt mode is edge sensitive (PMODE = 0), the rising edge interrupt is enabled.
- If the pin interrupt mode is level sensitive (PMODE = 1), the level interrupt is enabled. The IENF register configures the active level (HIGH or LOW) for this interrupt.

Table 217. Pin interrupt level or rising edge interrupt enable register (IENR, offset 0x004) bit description

Bit	Symbol	Description	Reset value	Access
7:0	ENRL	Enables the rising edge or level interrupt for each pin interrupt. Bit n configures the pin interrupt selected in PINTSELn. 0 = Rising edge or level interrupt is disabled. 1 = Rising edge or level interrupt is enabled.	0x0	R/W
31:8	-	Reserved. Read value is undefined, only zero should be written.	-	-

12.6.3 Pin interrupt level or rising edge interrupt enable set register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Table 197), one bit in the SIENR register sets the corresponding bit in the IENR register depending on the pin interrupt mode configured in the ISEL register. See IENR description.

Table 218. Pin interrupt level or rising edge interrupt set register (SIENR, offset 0x008) bit description

Bit	Symbol	Description	Reset value	Access
7:0	SETENRL	Ones written to this address set bits in the IENR, thus enabling interrupts. Bit n sets bit n in the IENR register.	-	WO
31:8	-	Reserved.	-	-

12.6.4 Pin interrupt level or rising edge interrupt enable clear register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Table 197), one bit in the CIENR register clears the corresponding bit in the IENR register depending on the pin interrupt mode configured in the ISEL register. See IENR description.

Table 219. Pin interrupt level or rising edge interrupt clear register (CIENR, offset 0x00C) bit description

Bit	Symbol	Description	Reset value	Access
7:0	CENRL	Ones written to this address clear bits in the IENR, thus disabling the interrupts. Bit n clears bit n in the IENR register.	-	WO
31:8	-	Reserved.	-	-

12.6.5 Pin interrupt active level or falling edge interrupt enable register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Table 197), one bit in the IENF register enables the falling edge interrupt or the configures the level sensitivity depending on the pin interrupt mode configured in the ISEL register:

- If the pin interrupt mode is edge sensitive (PMODE = 0), the falling edge interrupt is enabled.
- If the pin interrupt mode is level sensitive (PMODE = 1), the active level of the level interrupt (HIGH or LOW) is configured.

Table 220. Pin interrupt active level or falling edge interrupt enable register (IENF, offset 0x010) bit description

Bit	Symbol	Description	Reset value	Access
7:0	ENAF	Enables the falling edge or configures the active level interrupt for each pin interrupt. Bit n configures the pin interrupt selected in PINTSELn. 0 = Disable falling edge interrupt or set active interrupt level LOW. 1 = Enable falling edge interrupt enabled or set active interrupt level HIGH.	0x0	R/W
31:8	-	Reserved.	-	-

12.6.6 Pin interrupt active level or falling edge interrupt set register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Table 197), one bit in the SIENF register sets the corresponding bit in the IENF register depending on the pin interrupt mode configured in the ISEL register:

- If the pin interrupt mode is edge sensitive (PMODE = 0), the falling edge interrupt is set.
- If the pin interrupt mode is level sensitive (PMODE = 1), the HIGH-active interrupt is selected.

Table 221. Pin interrupt active level or falling edge interrupt set register (SIENF, offset 0x014) bit description

Bit	Symbol	Description	Reset value	Access
7:0	SETENAF	Ones written to this address set bits in the IENF, thus enabling interrupts. Bit n sets bit n in the IENF register. 0 = No operation. 1 = Select HIGH-active interrupt or enable falling edge interrupt.	-	WO
31:8	-	Reserved.	-	-

12.6.7 Pin interrupt active level or falling edge interrupt clear register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Table 197), one bit in the CIENF register sets the corresponding bit in the IENF register depending on the pin interrupt mode configured in the ISEL register:

- If the pin interrupt mode is edge sensitive (PMODE = 0), the falling edge interrupt is cleared.
- If the pin interrupt mode is level sensitive (PMODE = 1), the LOW-active interrupt is selected.

Table 222. Pin interrupt active level or falling edge interrupt clear register (CIENF, offset 0x018) bit description

Bit	Symbol	Description	Reset value	Access
7:0	CENAF	Ones written to this address clears bits in the IENF, thus disabling interrupts. Bit n clears bit n in the IENF register. 0 = No operation. 1 = LOW-active interrupt selected or falling edge interrupt disabled.	-	WO
31:8	-	Reserved.	-	-

12.6.8 Pin interrupt rising edge register

This register contains ones for pin interrupts selected in the PINTSELn registers (see Table 197) on which a rising edge has been detected. Writing ones to this register clears rising edge detection. Ones in this register assert an interrupt request for pins that are enabled for rising-edge interrupts. All edges are detected for all pins selected by the PINTSELn registers, regardless of whether they are interrupt-enabled.

Table 223. Pin interrupt rising edge register (RISE, offset 0x01C) bit description

Bit	Symbol	Description	Reset value	Access
7:0	RDET	Rising edge detect. Bit n detects the rising edge of the pin selected in PINTSELn. Read 0: No rising edge has been detected on this pin since Reset or the last time a one was written to this bit. Write 0: no operation. Read 1: a rising edge has been detected since Reset or the last time a one was written to this bit. Write 1: clear rising edge detection for this pin.	0x0	R/W
31:8	-	Reserved.	-	-

12.6.9 Pin interrupt falling edge register

This register contains ones for pin interrupts selected in the PINTSELn registers (see Table 197) on which a falling edge has been detected. Writing ones to this register clears falling edge detection. Ones in this register assert an interrupt request for pins that are enabled for falling-edge interrupts. All edges are detected for all pins selected by the PINTSELn registers, regardless of whether they are interrupt-enabled.

Table 224. Pin interrupt falling edge register (FALL, offset 0x020) bit description

Bit	Symbol	Description	Reset value	Access
7:0	FDET	Falling edge detect. Bit n detects the falling edge of the pin selected in PINTSELn. Read 0: No falling edge has been detected on this pin since Reset or the last time a one was written to this bit. Write 0: no operation. Read 1: a falling edge has been detected since Reset or the last time a one was written to this bit. Write 1: clear falling edge detection for this pin.	0x0	R/W
31:8	-	Reserved.	-	-

12.6.10 Pin interrupt status register

Reading this register returns ones for pin interrupts that are currently requesting an interrupt. For pins identified as edge-sensitive in the Interrupt Select register, writing ones to this register clears both rising- and falling-edge detection for the pin. For level-sensitive pins, writing ones inverts the corresponding bit in the Active level register, thus switching the active level on the pin.

Table 225. Pin interrupt status register (IST, offset 0x024) bit description

Bit	Symbol	Description	Reset value	Access
7:0	PSTAT	Pin interrupt status. Bit n returns the status, clears the edge interrupt, or inverts the active level of the pin selected in PINTSELn. Read 0: interrupt is not being requested for this interrupt pin. Write 0: no operation. Read 1: interrupt is being requested for this interrupt pin. Write 1, when pin is in edge-sensitive mode: clear rising- and falling-edge detection for this pin. Write 1, when pin is in level-sensitive mode: switch the active level for this pin (in the IENF register).	0x0	R/W
31:8	-	Reserved.	-	-

12.6.11 Pattern Match Interrupt Control Register

The pattern match control register contains one bit to select pattern-match interrupt generation (as opposed to pin interrupts which share the same interrupt request lines), and another to enable the RXEV output to the CPU. This register also allows the current state of any pattern matches to be read.

If the pattern match feature is not used (either for interrupt generation or for RXEV assertion) bits SEL_PMATCH and ENA_RXEV of this register should be left at 0 to conserve power.

Remark: Set up the pattern-match configuration in the PMSRC and PMCFG registers before writing to this register to enable (or re-enable) the pattern-match functionality. This eliminates the possibility of spurious interrupts as the feature is being enabled.

Remark: The pattern match feature requires clocks in order to operate, and can thus not generate an interrupt or wake up the device during reduced power modes below sleep mode.

Table 226. Pattern match interrupt control register (PMCTRL, offset 0x028) bit description

Bit	Symbol	Value	Description	Reset value
0	SEL_PMATCH		Specifies whether the 8 pin interrupts are controlled by the pin interrupt function or by the pattern match function.	0x0
		0	Pin interrupt. Interrupts are driven in response to the standard pin interrupt function.	
		1	Pattern match. Interrupts are driven in response to pattern matches.	
1	ENA_RXEV		Enables the RXEV output to the CPU when the specified boolean expression evaluates to true.	0x0
		0	Disabled. RXEV output to the CPU is disabled.	
		1	Enabled. RXEV output to the CPU is enabled.	
23:2	-	-	Reserved. Do not write 1s to unused bits.	0x0
31:24	PMAT	-	This field displays the current state of pattern matches. A 1 in any bit of this field indicates that the corresponding product term is matched by the current state of the appropriate inputs.	0x0

12.6.12 Pattern Match Interrupt Bit-Slice Source register

The bit-slice source register specifies the input source for each of the eight pattern match bit slices.

Each of the possible eight inputs is selected in the pin interrupt select registers in the SYSCON block. See Section 10.6.1. Input 0 corresponds to the pin selected in the PINTSEL0 register, input 1 corresponds to the pin selected in the PINTSEL1 register, and so forth.

Remark: Writing any value to either the PMCFG register or the PMSRC register, or disabling the pattern-match feature (by clearing both the SEL_PMATCH and ENA_RXEV bits in the PMCTRL register to zeros) will erase all edge-detect history.

Table 227. Pattern match bit-slice source register (PMSRC, offset 0x02C) bit description

Bit	Symbol	Value	Description	Reset value
7:0	Reserved	-	Software should not write 1s to unused bits.	0x0
10:8	SRC0		Selects the input source for bit slice 0	0x0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 0.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 0.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 0.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 0.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 0.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 0.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 0.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 0.	
13:11	SRC1		Selects the input source for bit slice 1	0x0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 1.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 1.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 1.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 1.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 1.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 1.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 1.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 1.	
16:14	SRC2		Selects the input source for bit slice 2	0x0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 2.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 2.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 2.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 2.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 2.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 2.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 2.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 2.	

Table 227. Pattern match bit-slice source register (PMSRC, offset 0x02C) bit description

Bit	Symbol	Value	Description	Reset value
19:17	SRC3		Selects the input source for bit slice 3	0x0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 3.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 3.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 3.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 3.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 3.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 3.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 3.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 3.	
22:20	SRC4		Selects the input source for bit slice 4	0x0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 4.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 4.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 4.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 4.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 4.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 4.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 4.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 4.	
25:23	SRC5		Selects the input source for bit slice 5	0x0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 5.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 5.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 5.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 5.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 5.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 5.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 5.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 5.	
28:26	SRC6		Selects the input source for bit slice 6	0x0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 6.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 6.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 6.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 6.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 6.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 6.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 6.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 6.	

Table 227. Pattern match bit-slice source register (PMSRC, offset 0x02C) bit description

Bit	Symbol	Value	Description	Reset value
31:29	SRC7		Selects the input source for bit slice 7	0x0
		0x0	Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 7.	
		0x1	Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 7.	
		0x2	Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 7.	
		0x3	Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 7.	
		0x4	Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 7.	
		0x5	Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 7.	
		0x6	Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 7.	
		0x7	Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 7.	

12.6.13 Pattern Match Interrupt Bit Slice Configuration register

The bit-slice configuration register configures the detect logic and contains bits to select from among eight alternative conditions for each bit slice that cause that bit slice to contribute to a pattern match. The seven LSBs of this register specify which bit-slices are the end-points of product terms in the boolean expression (i.e. where OR terms are to be inserted in the expression).

Two types of edge detection on each input are possible:

- Sticky: A rising edge, a falling edge, or a rising or falling edge that is detected at any time after the edge-detection mechanism has been cleared. The input qualifies as detected (the detect logic output remains HIGH) until the pattern match engine detect logic is cleared again.
- Non-sticky: Every time an edge (rising or falling) is detected, the detect logic output for this pin goes HIGH. This bit is cleared after one clock cycle, and the edge detect logic can detect another edge,

Remark: To clear the pattern match engine detect logic, write any value to either the PMCFG register or the PMSRC register, or disable the pattern-match feature (by clearing both the SEL_PMATCH and ENA_RXEV bits in the PMCTRL register to zeros). This will erase all edge-detect history.

To select whether a slice marks the final component in a minterm of the boolean expression, write a 1 in the corresponding PROD_ENPTSn bit. Setting a term as the final component has two effects:

1. The interrupt request associated with this bit slice will be asserted whenever a match to that product term is detected.
2. The next bit slice will start a new, independent product term in the boolean expression (i.e. an OR will be inserted in the boolean expression following the element controlled by this bit slice).

Table 228. Pattern match bit slice configuration register (PMCFG, offset 0x030) bit description

Bit	Symbol	Value	Description	Reset value
0	PROD_ENDPTS0		Determines whether slice 0 is an endpoint.	0x0
		0	No effect. Slice 0 is not an endpoint.	
		1	endpoint. Slice 0 is the endpoint of a product term (minterm). Pin interrupt 0 in the NVIC is raised if the minterm evaluates as true.	
1	PROD_ENDPTS1		Determines whether slice 1 is an endpoint.	0x0
		0	No effect. Slice 1 is not an endpoint.	
		1	endpoint. Slice 1 is the endpoint of a product term (minterm). Pin interrupt 1 in the NVIC is raised if the minterm evaluates as true.	
2	PROD_ENDPTS2		Determines whether slice 2 is an endpoint.	0x0
		0	No effect. Slice 2 is not an endpoint.	
		1	endpoint. Slice 2 is the endpoint of a product term (minterm). Pin interrupt 2 in the NVIC is raised if the minterm evaluates as true.	
3	PROD_ENDPTS3		Determines whether slice 3 is an endpoint.	0x0
		0	No effect. Slice 3 is not an endpoint.	
		1	endpoint. Slice 3 is the endpoint of a product term (minterm). Pin interrupt 3 in the NVIC is raised if the minterm evaluates as true.	
4	PROD_ENDPTS4		Determines whether slice 4 is an endpoint.	0x0
		0	No effect. Slice 4 is not an endpoint.	
		1	endpoint. Slice 4 is the endpoint of a product term (minterm). Pin interrupt 4 in the NVIC is raised if the minterm evaluates as true.	
5	PROD_ENDPTS5		Determines whether slice 5 is an endpoint.	0x0
		0	No effect. Slice 5 is not an endpoint.	
		1	endpoint. Slice 5 is the endpoint of a product term (minterm). Pin interrupt 5 in the NVIC is raised if the minterm evaluates as true.	
6	PROD_ENDPTS6		Determines whether slice 6 is an endpoint.	0x0
		0	No effect. Slice 6 is not an endpoint.	
		1	endpoint. Slice 6 is the endpoint of a product term (minterm). Pin interrupt 6 in the NVIC is raised if the minterm evaluates as true.	
7	-	-	Reserved. Bit slice 7 is automatically considered a product end point.	0x0

Table 228. Pattern match bit slice configuration register (PMCFG, offset 0x030) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
10:8	CFG0		Specifies the match contribution condition for bit slice 0.	0x0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	
13:11	CFG1		Specifies the match contribution condition for bit slice 1.	0x0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	

Table 228. Pattern match bit slice configuration register (PMCFG, offset 0x030) bit description *...continued*

Bit	Symbol	Value	Description	Reset value
16:14	CFG2		Specifies the match contribution condition for bit slice 2.	0x0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	
19:17	CFG3		Specifies the match contribution condition for bit slice 3.	0x0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	

Table 228. Pattern match bit slice configuration register (PMCFG, offset 0x030) bit description *...continued*

Bit	Symbol	Value	Description	Reset value
22:20	CFG4		Specifies the match contribution condition for bit slice 4.	0x0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	
25:23	CFG5		Specifies the match contribution condition for bit slice 5.	0x0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	

Table 228. Pattern match bit slice configuration register (PMCFG, offset 0x030) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
28:26	CFG6		Specifies the match contribution condition for bit slice 6.	0x0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	
31:29	CFG7		Specifies the match contribution condition for bit slice 7.	0x0
		0x0	Constant HIGH. This bit slice always contributes to a product term match.	
		0x1	Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x2	Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x3	Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to.	
		0x4	High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register.	
		0x5	Low level. Match occurs when there is a low level on the specified input.	
		0x6	Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices).	
		0x7	Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.	

12.7 Functional description

12.7.1 Pin interrupts

In this interrupt facility, up to 8 pins are identified as interrupt sources by the Pin Interrupt Select registers (PINTSEL0-7). All registers in the pin interrupt block contain 8 bits, corresponding to the pins called out by the PINTSEL0-7 registers. The ISEL register defines whether each interrupt pin is edge- or level-sensitive. The RISE and FALL registers detect edges on each interrupt pin, and can be written to clear (and set) edge detection. The IST register indicates whether each interrupt pin is currently requesting an interrupt, and this register can also be written to clear interrupts.

The other pin interrupt registers play different roles for edge-sensitive and level-sensitive pins, as described in Table 229.

Table 229. Pin interrupt registers for edge- and level-sensitive pins

Name	Edge-sensitive function	Level-sensitive function
IENR	Enables rising-edge interrupts.	Enables level interrupts.
SIENR	Write to enable rising-edge interrupts.	Write to enable level interrupts.
CIENR	Write to disable rising-edge interrupts.	Write to disable level interrupts.
IENF	Enables falling-edge interrupts.	Selects active level.
SIENF	Write to enable falling-edge interrupts.	Write to select high-active.
CIENF	Write to disable falling-edge interrupts.	Write to select low-active.

12.7.2 Pattern Match engine example

Suppose the desired boolean pattern to be matched is:
(IN1) + (IN1 * IN2) + (~IN2 * ~IN3 * IN6fe) + (IN5 * IN7ev)

with:

IN6fe = (sticky) falling-edge on input 6

IN7ev = (non-sticky) event (rising or falling edge) on input 7

Each individual term in the expression shown above is controlled by one bit-slice. To specify this expression, program the pattern match bit slice source and configuration register fields as follows:

- PMSRC register (Table 227):
  - Since bit slice 5 will be used to detect a sticky event on input 6, a 1 can be written to the SRC5 bits to clear any pre-existing edge detects on bit slice 5.
  - SRC0: 001 - select input 1 for bit slice 0
  - SRC1: 001 - select input 1 for bit slice 1
  - SRC2: 010 - select input 2 for bit slice 2
  - SRC3: 010 - select input 2 for bit slice 3
  - SRC4: 011 - select input 3 for bit slice 4
  - SRC5: 110 - select input 6 for bit slice 5
  - SRC6: 101 - select input 5 for bit slice 6

  – SRC7: 111 - select input 7 for bit slice 7
- PMCFG register (Table 228):
  – PROD_ENDPTS0 = 1
  – PROD_ENDPTS02 = 1
  – PROD_ENDPTS5 = 1
  – All other slices are not product term endpoints and their PROD_ENDPTS bits are 0. Slice 7 is always a product term endpoint and does not have a register bit associated with it.
  – = 0100101 - bit slices 0, 2, 5, and 7 are product-term endpoints. (Bit

    slice 7 is an endpoint by default - no associated register bit).
  – CFG0: 000 - high level on the selected input (input 1) for bit slice 0
  – CFG1: 000 - high level on the selected input (input 1) for bit slice 1
  – CFG2: 000 - high level on the selected input (input 2) for bit slice 2
  – CFG3: 101 - low level on the selected input (input 2) for bit slice 3
  – CFG4: 101 - low level on the selected input (input 3) for bit slice 4
  – CFG5: 010 - (sticky) falling edge on the selected input (input 6) for bit slice 5
  – CFG6: 000 - high level on the selected input (input 5) for bit slice 6
  – CFG7: 111 - event (any edge, non-sticky) on the selected input (input 7) for bit slice 7
- PMCTRL register (Table 226):
  – Bit0: Setting this bit will select pattern matches to generate the pin interrupts in place of the normal pin interrupt mechanism.

    For this example, pin interrupt 0 will be asserted when a match is detected on the first product term (which, in this case, is just a high level on input 1).

    Pin interrupt 2 will be asserted in response to a match on the second product term.

    Pin interrupt 5 will be asserted when there is a match on the third product term.

    Pin interrupt 7 will be asserted on a match on the last term.
  – Bit1: Setting this bit will cause the RxEv signal to the CPU to be asserted whenever a match occurs on ANY of the product terms in the expression. Otherwise, the RXEV line will not be used.
  – Bit31:24: At any given time, bits 0, 2, 5 and/or 7 may be high if the corresponding product terms are currently matching.
  – The remaining bits will always be low.

12.7.3 Pattern match engine edge detect examples

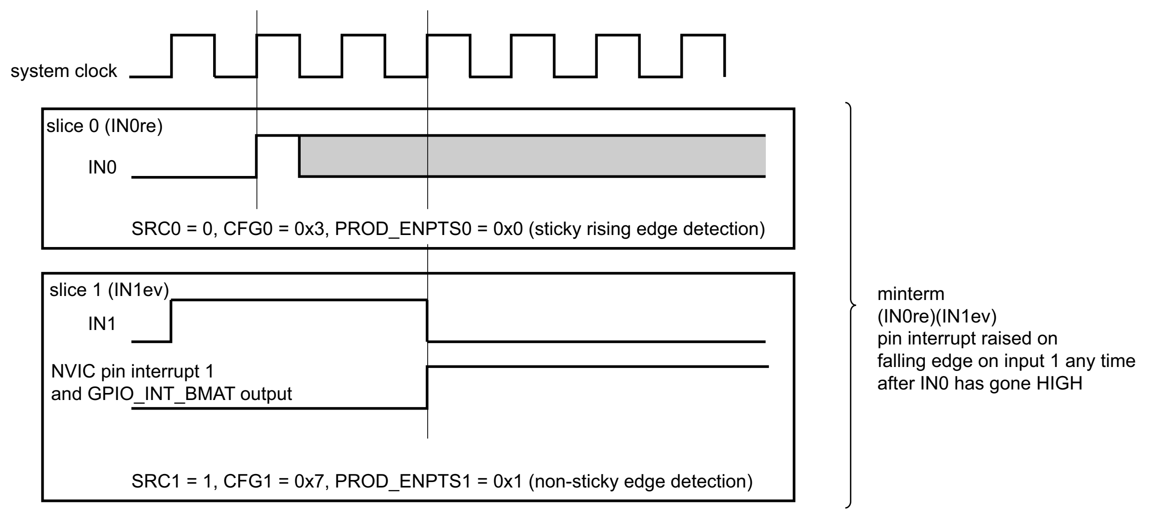

system clock
slice 0 (IN0re)
IN0
SRC0 = 0, CFG0 = 0x3, PROD_ENPTS0 = 0x0 (sticky rising edge detection)
slice 1 (IN1ev)
IN1
NVIC pin interrupt 1
and GPIO_INT_BMAT output
SRC1 = 1, CFG1 = 0x7, PROD_ENPTS1 = 0x1 (non-sticky edge detection)
minterm
(IN0re)(IN1ev)
pin interrupt raised on
falling edge on input 1 any time
after IN0 has gone HIGH

Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 20. Pattern match engine examples: sticky edge detect

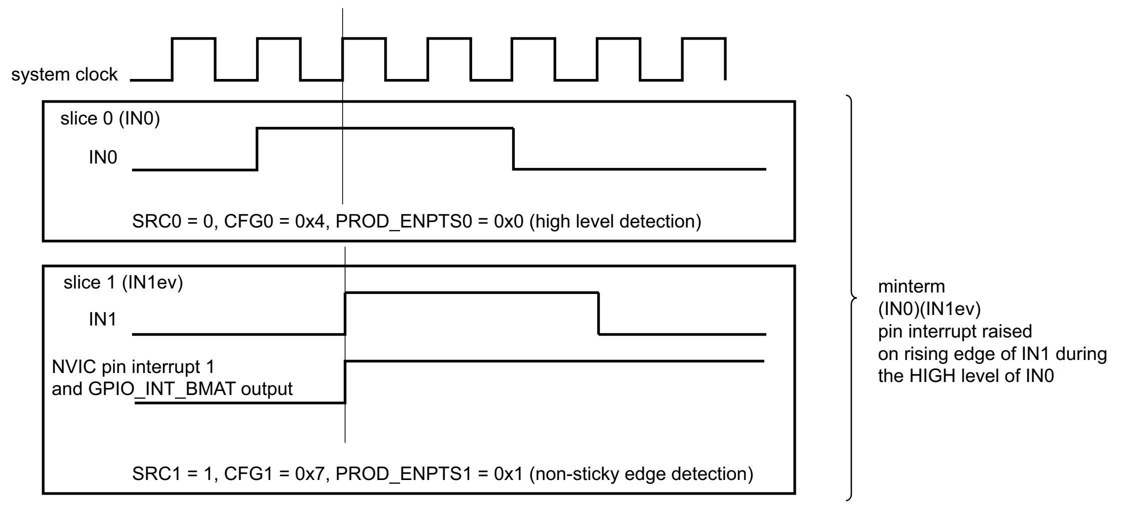

system clock
slice 0 (IN0)
IN0
SRC0 = 0, CFG0 = 0x4, PROD_ENPTS0 = 0x0 (high level detection)
slice 1 (IN1ev)
IN1
NVIC pin interrupt 1
and GPIO_INT_BMAT output
SRC1 = 1, CFG1 = 0x7, PROD_ENPTS1 = 0x1 (non-sticky edge detection)
minterm
(IN0)(IN1ev)
pin interrupt raised
on rising edge of IN1 during
the HIGH level of IN0

Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 21. Pattern match engine examples: Windowed non-sticky edge detect evaluates as true

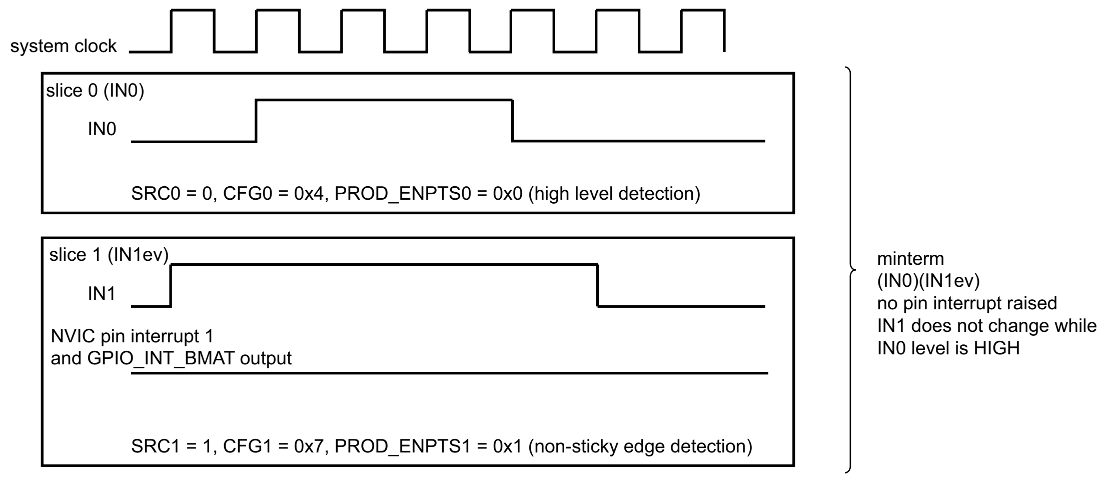

system clock
slice 0 (IN0)
IN0
SRC0 = 0, CFG0 = 0x4, PROD_ENPTS0 = 0x0 (high level detection)
slice 1 (IN1ev)
IN1
NVIC pin interrupt 1
and GPIO_INT_BMAT output
SRC1 = 1, CFG1 = 0x7, PROD_ENPTS1 = 0x1 (non-sticky edge detection)
minterm
(IN0)(IN1ev)
no pin interrupt raised
IN1 does not change while
IN0 level is HIGH

Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 22. Pattern match engine examples: Windowed non-sticky edge detect evaluates as false

Chapter 13: LPC51U68 Group GPIO input interrupt (GINT0/1)

13.1 Features

- The inputs from any number of digital pins can be enabled to contribute to a combined group interrupt.
- The polarity of each input enabled for the group interrupt can be configured HIGH or LOW.
- Enabled interrupts can be logically combined through an OR or AND operation.
- Two group interrupts are supported to reflect two distinct interrupt patterns.
- The grouped interrupts can wake up the part from sleep or deep-sleep modes.

13.2 Basic configuration

For the group interrupt feature, enable the clock to both the GROUP0 and GROUP1 register interfaces in the AHBCLKCTRL0 register ((Table 115). The group interrupt wake-up feature is enabled in the STARTER0 register for GINT0 (Table 155). The interrupt must also be enabled in the NVIC (see Table 79).

The pins can be configured as GPIO pins through IOCON, but they don't have to be. The GINT block reads the input from the pin bypassing IOCON multiplexing. Make sure that no analog function is selected on pins that are input to the group interrupts. Selecting an analog function in IOCON disables the digital portion of the pin and the digital signal is tied to 0.

13.3 General description

The GPIO pins can be used in several ways to set pins as inputs or outputs and use the inputs as combinations of level and edge sensitive interrupts.

For each port/pin connected to one of the two the GPIO Grouped Interrupt blocks (GROUP0 and GROUP1), the GPIO grouped interrupt registers determine which pins are enabled to generate interrupts and what the active polarities of each of those inputs are.

The GPIO grouped interrupt registers also select whether the interrupt output will be level or edge triggered and whether it will be based on the OR or the AND of all of the enabled inputs.

When the designated pattern is detected on the selected input pins, the GPIO grouped interrupt block generates an interrupt. If the part is in a power-savings mode, it I first asynchronously wakes the part up prior to asserting the interrupt request. The interrupt request line can be cleared by writing a one to the interrupt status bit in the control register.

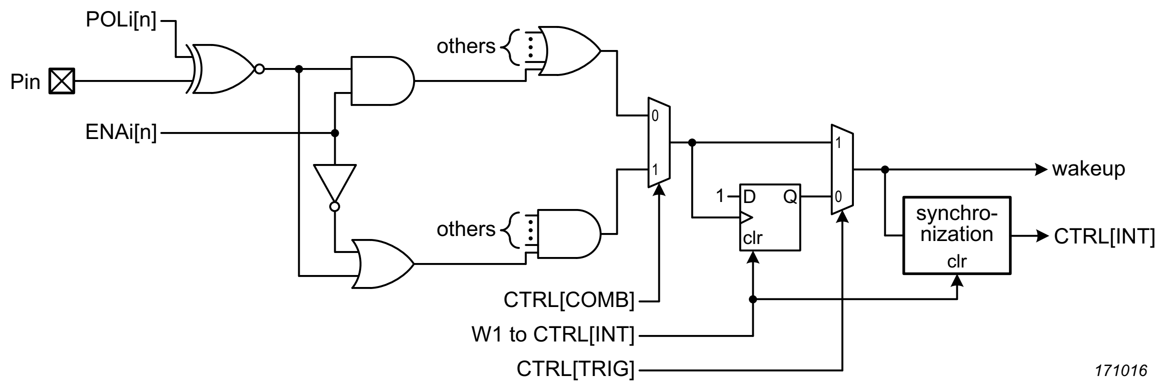


Fig 23. Group GPIO interrupt block diagram

13.4 Register description

Note: In all registers, bits that are not shown are **reserved**.

Table 230. Register overview: GROUP0 interrupt (base address 0x4000 2000 (GINT0) and 0x4000 3000 (GINT1))

Name	Access	Offset	Description	Reset value	Section
CTRL	R/W	0x000	GPIO grouped interrupt control register	0x0	13.4.1
PORT_POL0	R/W	0x020	GPIO grouped interrupt port 0 polarity register	0xFFFF FFFF	13.4.2
PORT_POL1	R/W	0x024	GPIO grouped interrupt port 1 polarity register	0xFFFF FFFF	13.4.2
PORT_ENA0	R/W	0x040	GPIO grouped interrupt port 0 enable register	0x0	13.4.3
PORT_ENA1	R/W	0x044	GPIO grouped interrupt port 1 enable register	0x0	13.4.3

13.4.1 Grouped interrupt control register

Table 231. GPIO grouped interrupt control register (CTRL, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value
0	INT		Group interrupt status. This bit is cleared by writing a one to it. Writing zero has no effect.	0x0
		0	No request. No interrupt request is pending.	
		1	Request active. Interrupt request is active.	
1	COMB		Combine enabled inputs for group interrupt	0x0
		0	Or. OR functionality: A grouped interrupt is generated when any one of the enabled inputs is active (based on its programmed polarity).	
		1	And. AND functionality: An interrupt is generated when all enabled bits are active (based on their programmed polarity).	
2	TRIG		Group interrupt trigger	0x0
		0	Edge-triggered.	
		1	Level-triggered.	
31:3	-	-	Reserved. Read value is undefined, only zero should be written.	0x0

13.4.2 GPIO grouped interrupt port polarity registers

The grouped interrupt port polarity registers determine how the polarity of each enabled pin contributes to the grouped interrupt. Each port is associated with its own port polarity register, and the values of both registers together determine the grouped interrupt.

Each register PORT_POLm controls the polarity of pins in port m.

Table 232. GPIO grouped interrupt port polarity registers (PORT_POL[0:1], offset 0x020 for POL0; 0x024 for POL1) bit description

Bit	Symbol	Description	Reset value
31:0	POL	Configure pin polarity of port m pins for group interrupt. Bit n corresponds to pin PIOm_n of port m. 0 = the pin is active LOW. If the level on this pin is LOW, the pin contributes to the group interrupt. 1 = the pin is active HIGH. If the level on this pin is HIGH, the pin contributes to the group interrupt.	0xFFFF FFFF

13.4.3 GPIO grouped interrupt port enable registers

The grouped interrupt port enable registers enable the pins which contribute to the grouped interrupt. Each port is associated with its own port enable register, and the values of both registers together determine which pins contribute to the grouped interrupt.

Each register PORT_ENm enables pins in port m.

Table 233. GPIO grouped interrupt port enable registers (PORT_ENA[0:1], offset 0x040 for PORT_ENA0; 0x044 PORT_ENA1) bit description

Bit	Symbol	Description	Reset value
31:0	ENA	Enable port 0 pin for group interrupt. Bit n corresponds to pin Pm_n of port m. 0 = the port 0 pin is disabled and does not contribute to the grouped interrupt. 1 = the port 0 pin is enabled and contributes to the grouped interrupt.	0x0

13.5 Functional description

Any subset of the pins in each port can be selected to contribute to a common group interrupt (GINT) and can be enabled to wake the part up from deep-sleep mode.

An interrupt can be requested for each port, based on any selected subset of pins within each port. The pins that contribute to each port interrupt are selected by 1s in the port's Enable register, and an interrupt polarity can be selected for each pin in the port's Polarity register. The level on each pin is exclusive-ORed with its polarity bit, and the result is ANDed with its enable bit. These results are then inclusive-ORed among all the pins in the port to create the port's raw interrupt request.

The raw interrupt request from each of the two group interrupts is sent to the NVIC, which can be programmed to treat it as level- or edge-sensitive, or it can be edge-detected by the wake-up interrupt logic (see Table 155).

Chapter 14: LPC51U68 DMA controller

14.1 How to read this chapter

The DMA controller is available on all LPC51U68 devices.

14.2 Features

- 18 channels, 16 of which are connected to peripheral DMA requests. These come from the Flexcomm Interface (USART, SPI, I^{2}C, and I^{2}S). One spare channel has no DMA request connected and can be used for functions such as memory-to-memory moves. Any otherwise unused channel can also be used for other purposes.
- DMA operations can be triggered by on- or off-chip events. Each DMA channel can select one trigger input from 16 sources. Trigger sources include ADC interrupts, Timer interrupts, pin interrupts, and the SCT DMA request lines.
- Priority is user selectable for each channel (up to eight priority levels).
- Continuous priority arbitration.
- Address cache with four entries (each entry is a pair of transfer addresses).
- Efficient use of data bus.
- Supports single transfers up to 1,024 words. A single transfer is defined as one read from the source address, followed by one write to the destination address.
- Address increment options allow packing and/or unpacking data.

14.3 Basic configuration

Configure the DMA as follows:
Use the AHBCLKCTRL0 register (Table 115) to enable the clock to the DMA registers interface.

- Clear the DMA peripheral reset using the PRESETCTRL0 register (Table 108).
- The DMA controller provides an interrupt to the NVIC, see Table 79.
- Most peripherals that support DMA, the ADC being an exception, have at least one DMA request line associated with them. The related channel(s) must be set up according to the desired operation. the ADC uses a trigger instead of a DMA request. DMA requests and triggers are described in detail in Section 14.5.1
- For peripherals using DMA requests, DMA operation must be triggered before any transfer will occur. This can be done by software, or can optionally be signalled by one of 16 hardware triggers, through the input mux registers DMA_ITRIG_INMUX[0:19]. DMA requests and triggers are described in detail in Section 14.5.1
- Trigger outputs may optionally cause other DMA channels to be triggered for more complex DMA functions. Trigger outputs are connected to DMA_INMUX_INMUX[0:3] as inputs to DMA triggers.

For details on the trigger input and output multiplexing, see Section 10.5.2 “DMA trigger input multiplexing”.

14.4 Pin description

The DMA controller has no direct pin connections. However, some DMA triggers can be associated with pin functions (see Section 14.5.1.2).

14.5 General description

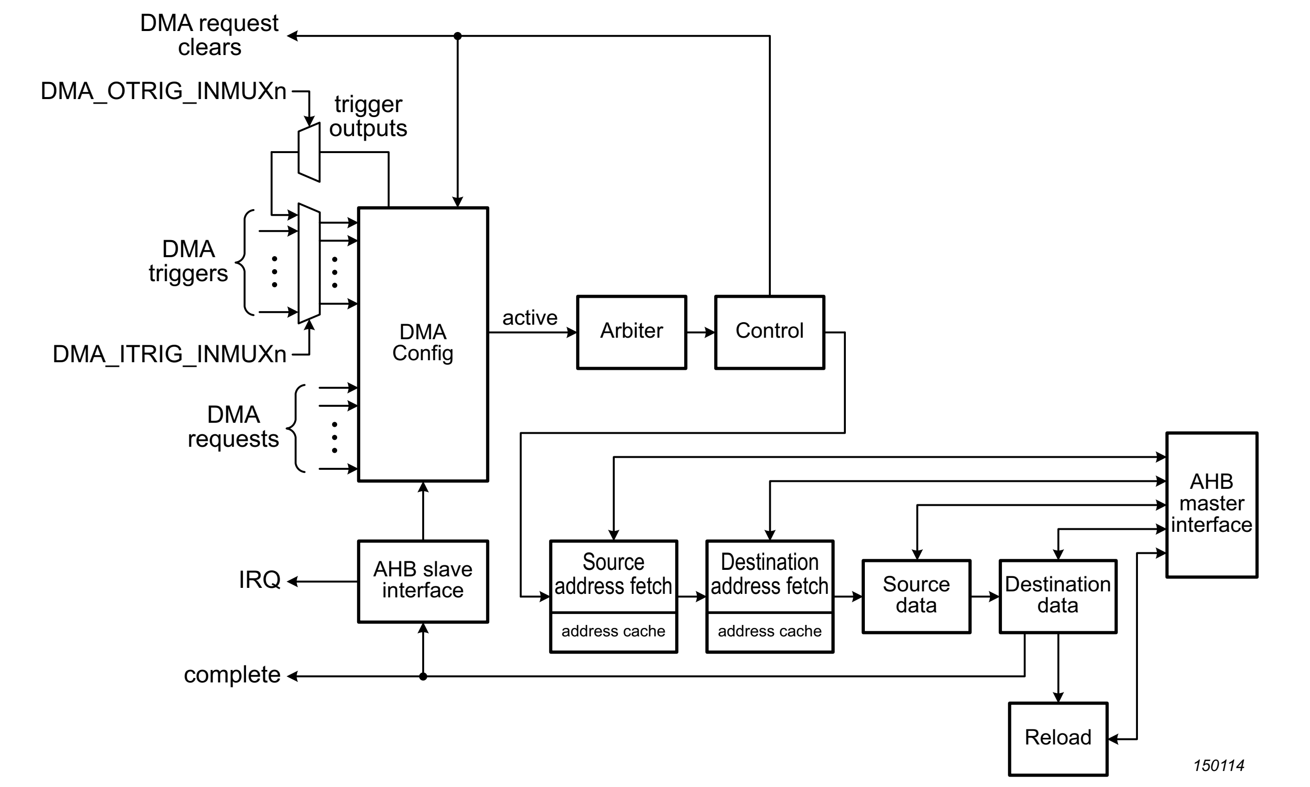


Fig 24. DMA block diagram

14.5.1 DMA requests and triggers

In general, DMA requests are intended to pace transfers to match what the peripheral (including its FIFO if it has one) can do. For example, the USART will issue a transmit DMA request when its transmit FIFO is not full, and a receive DMA request when its receive FIFO is not empty. DMA requests are summarized in Table 234.

Triggers start the transfer. In typical cases, only a software trigger will probably be used. Other possibilities are provided for, such as starting a DMA transfer when certain timer or pin related events occur. Those transfers would usually still be paced by a peripheral DMA request if a peripheral is involved in the transfer. Note that no DMA activity will take place for any particular DMA channel unless that channel has been triggered, either by software or hardware. DMA triggers are summarized in Table 236.

There is one specific exception to the above description, which is the ADC. The ADC doesn't fit the simple pacing signal model very well because of the possibilities represented by the programmable conversion sequences. A sequence complete DMA request is likely to require transferring several non-contiguous result registers at once (see Chapter 28 "LPC51U68 12-bit ADC controller (ADC)"). It might also require other things to be done that can be done by the DMA without software intervention. This model fits better with the trigger facility, so that is how the ADC is connected to the DMA controller.

Once triggered by software or hardware, a DMA operation on a specific channel is initiated by a DMA request if it is enabled for that channel.

A DMA channel using a trigger can respond by moving data from any memory address to any other memory address. This can include fixed peripheral data registers, or incrementing through RAM buffers. The amount of data moved by a single trigger event can range from a single transfer to many transfers. A transfer that is started by a trigger can still be paced using the channel's DMA request. This allows sending a string to a serial peripheral, for instance, without overrunning the peripheral's transmit buffer.

Each DMA channel also has an output that can be used as a trigger input to another channel. The trigger outputs appear in the trigger source list for each channel and can be selected through the DMA_INMUX registers as inputs to other channels.

14.5.1.1 DMA requests

DMA requests are directly connected to the peripherals. Each channel supports one DMA request line and one trigger input which is multiplexed to many possible input sources, as shown in Table 234.

Table 234. DMA requests & trigger muxes

DMA channel #	Request input	DMA trigger mux
0	Flexcomm Interface 0 RX / I2C Slave [1]	DMA_ITRIG_INMUX0
1	Flexcomm Interface 0 TX / I2C Master [1]	DMA_ITRIG_INMUX1
2	Flexcomm Interface 1 RX / I2C Slave [1]	DMA_ITRIG_INMUX2
3	Flexcomm Interface 1 TX / I2C Master [1]	DMA_ITRIG_INMUX3
4	Flexcomm Interface 2 RX / I2C Slave [1]	DMA_ITRIG_INMUX4
5	Flexcomm Interface 2 TX / I2C Master [1]	DMA_ITRIG_INMUX5
6	Flexcomm Interface 3 RX / I2C Slave [1]	DMA_ITRIG_INMUX6
7	Flexcomm Interface 3 TX / I2C Master [1]	DMA_ITRIG_INMUX7
8	Flexcomm Interface 4 RX / I2C Slave [1]	DMA_ITRIG_INMUX8
9	Flexcomm Interface 4 TX / I2C Master [1]	DMA_ITRIG_INMUX9
10	Flexcomm Interface 5 RX / I2C Slave [1]	DMA_ITRIG_INMUX10
11	Flexcomm Interface 5 TX / I2C Master [1]	DMA_ITRIG_INMUX11
12	Flexcomm Interface 6 RX / I2C Slave [1]	DMA_ITRIG_INMUX12
13	Flexcomm Interface 6 TX / I2C Master [1]	DMA_ITRIG_INMUX13
14	Flexcomm Interface 7 RX / I2C Slave [1]	DMA_ITRIG_INMUX14
15	Flexcomm Interface 7 TX / I2C Master [1]	DMA_ITRIG_INMUX15
16	Reserved	-
17	Reserved	-
18	(no DMA request)	DMA_ITRIG_INMUX18
19	(no DMA request)	DMA_ITRIG_INMUX19

[1] See Section 14.5.1.1.1 below for information about DMA for the I^{2}C Monitor function.

14.5.1.1.1 DMA with I2C monitor mode

The I^{2}C monitor function may be used with DMA if one of the channels related to the same Flexcomm Interface is available.

Table 235. DMA with the I^{2}C Monitor function

I^{2}C Master DMA	I^{2}C Slave DMA	I^{2}C Monitor DMA
Not enabled	-	If I^{2}C Monitor DMA is enabled, it will use the DMA channel for the Master function of the same Flexcomm Interface.
Enabled	Not enabled	If I^{2}C Monitor is DMA enabled, it will use the DMA channel for the Slave function of the same Flexcomm Interface.
Enabled	Enabled	The I^{2}C Monitor function cannot use DMA.

14.5.1.2 Hardware triggers

Each DMA channel can use one trigger that is independent of the request input for this channel. The trigger input is selected in the DMA_ITRIG_INMUX registers. There are 20 possible internal trigger sources for each DMA channel. In addition, the DMA trigger output can be routed to the trigger input of another channel through the trigger input multiplexing. See Table 234 and Section 10.5.2 “DMA trigger input multiplexing”.

Note that the ADC is unique in that it uses DMA triggers only, and has no DMA requests.

Table 236. DMA trigger sources

DMA trigger #	Trigger input	Software trigger
0	ADC0 sequence A interrupt	Yes
1	ADC0 sequence B interrupt	Yes
2	SCT0 DMA request 0	Yes
3	SCT0 DMA request 1	Yes
4	Timer CTIMER0 Match 0 DMA request	Yes
5	Timer CTIMER0 Match 1 DMA request	Yes
6	Timer CTIMER1 Match 0 DMA request	Yes
7	RESERVED	-
8	RESERVED	-
9	Timer CTIMER3 Match 0 DMA request	Yes
10	RESERVED	-
11	RESERVED	-
12	Pin interrupt 0	Yes
13	Pin interrupt 1	Yes
14	Pin interrupt 2	Yes
15	Pin interrupt 3	Yes
16	DMA output trigger 0	-
17	DMA output trigger 1	-
18	DMA output trigger 2	-
19	DMA output trigger 3	-

14.5.1.3 Trigger operation detail

A trigger of some kind is always needed to start a transfer on a DMA channel. This can be a hardware or software trigger, and can be used in several ways.

If a channel is configured with the SWTRIG bit equal to 0, the channel can be later triggered either by hardware or software. Software triggering is accomplished by writing a 1 to the appropriate bit in the SETTRIG register. Hardware triggering requires setup of the HWTRIGEN, TRIGPOL, TRIGTYPE, and TRIGBURST fields in the CFG register for the related channel. When a channel is initially set up, the SWTRIG bit in the XFERCFG register can be set, causing the transfer to begin immediately.

Once triggered, transfer on a channel will be paced by DMA requests if the PERIPHREQEN bit in the related CFG register is set. Otherwise, the transfer will proceed at full speed.

The TRIG bit in the CTLSTAT register can be cleared at the end of a transfer, determined by the value CLRTRIG (bit 0) in the XFERCFG register. When a 1 is found in CLRTRIG, the trigger is cleared when the descriptor is exhausted.

14.5.1.4 Trigger output detail

Each channel of the DMA controller provides a trigger output. This allows the possibility of using the trigger outputs as a trigger source to a different channel in order to support complex transfers on selected peripherals. This kind of transfer can, for example, use more than one peripheral DMA request. An example use would be to input data to a holding buffer from one peripheral, and then output the data to another peripheral, with both transfers being paced by the appropriate peripheral DMA request. This kind of operation is called “chained operation” or “channel chaining”.

14.5.2 DMA Modes

The DMA controller doesn’t really have separate operating modes, but there are ways of using the DMA controller that have commonly used terminology in the industry.

Using any specific DMA channel requires initializing the device registers associated with that channel, and setting up the Channel Descriptor. The Channel Descriptor is located somewhere in memory, typically in on-chip SRAM (see Section 14.6.3). The Channel Descriptor is shown in Table 237.

Note that the Channel Descriptor memory locations are actively used by the DMA controller when it operates on the related channel, and so must be re-initialized if the channel is used again. This is not true for additional reload descriptors that are linked to from the Channel Descriptor.

Table 237: Channel Descriptor

Offset	Description
+ 0x0	Reserved
+ 0x4	Source data end address
+ 0x8	Destination end address
+ 0xC	Link to next descriptor

When a DMA transfer involves a fixed peripheral data register, such as, when moving data from memory to a peripheral or moving data from a peripheral to memory, the address used for SRCINC or DSTINC (whichever corresponds to the fixed peripheral data address) is the address of the peripheral data register. The memory address for such a transfer is based on the end (upper) address of the memory buffer. The value can be calculated from the starting address of the buffer and the length of the buffer, where the transfer increment is the value specified by SRCINC or DSTINC (whichever corresponds to the memory buffer):

Buffer ending address = buffer starting address + (XFERCOUNT * the transfer increment)

See Section 14.6.18 for the description of SRCINC and DSTINC. Note that XFERCOUNT is defined as the actual count minus 1 and that is why it is not necessary to subtract 1 from the count in the equation above.

The DMA transfer is initiated by writing the channels CFG and XFERCFG registers and setting the channels ENABLESET bit. When everything is set up for a DMA transfer, it can be started by the occurrence of a hardware trigger of with t a software trigger.

After the channel has had a sufficient number of DMA requests and/or triggers, depending on its configuration, the initial descriptor will be exhausted. At that point, if the transfer configuration directs it, the Channel Descriptor will be reloaded with data from memory pointed to by the “Link to next descriptor” entry of the initial Channel Descriptor. Descriptors loaded in this manner look slightly different than the Channel Descriptor, as shown in Table 237. The difference is that a new transfer configuration is specified in the reload descriptor instead of being written to the XFERCFG register for that channel.

This process repeats as each descriptor is exhausted as long as reload is selected in the transfer configuration for each new descriptor.

Table 238: Reload descriptors

Offset	Description
+ 0x0	Transfer configuration.
+ 0x4	Source end address. This points to the address of the last entry of the source address range if the address is incremented. The address to be used in the transfer is calculated from the end address, data width, and transfer size.
+ 0x8	Destination end address. This points to the address of the last entry of the destination address range if the address is incremented. The address to be used in the transfer is calculated from the end address, data width, and transfer size.
+ 0xC	Link to next descriptor. If used, this address must be aligned to a multiple of 16 bytes (i.e., the size of a descriptor).

14.5.3 Single buffer

This generally applies to memory to memory moves, and peripheral DMA that occurs only occasionally and is set up for each transfer. For this kind of operation, only the initial channel descriptor shown in Table 239 is needed.

Table 239: Channel descriptor for a single transfer

Offset	Description
+ 0x0	Reserved

Table 239: Channel descriptor for a single transfer

Offset	Description
+ 0x4	Source data end address
+ 0x8	Destination data end address
+ 0xC	(not used)

This case is identified by the Reload bit in the XFERCFG register = 0. When the DMA channel receives a DMA request or trigger (depending on how it is configured), it performs one or more transfers as configured, then stops. Once the channel descriptor is exhausted, additional DMA requests or triggers will have no effect until the channel configuration is updated by software.

14.5.4 Ping-Pong

Ping-pong is a special case of a linked transfer. It is described separately because it is typically used more frequently than more complicated versions of linked transfers.

A ping-pong transfer uses two buffers alternately. At any one time, one buffer is being loaded or unloaded by DMA operations. The other buffer has the opposite operation being handled by software, readying the buffer for use when the buffer currently being used by the DMA controller is full or empty. Table 240 shows an example of descriptors for ping-pong from a peripheral to two buffers in memory.

Two reload descriptors are used in additional the Channel Descriptor due to the fact that the Channel Descriptor memory locations are actively used during DMA operation of the related DMA channel (as previously noted). In this example, Descriptor A is typically just a copy of the original Channel Descriptor.

Table 240: Example descriptors for ping-pong operation: peripheral to buffer

Channel Descriptor		Descriptor B		Descriptor A	
+ 0x0	(not used)	+ 0x0	Buffer B transfer configuration	+ 0x0	Buffer A transfer configuration
+ 0x4	Peripheral data end address	+ 0x4	Peripheral data end address	+ 0x4	Peripheral data end address
+ 0x8	Buffer A memory end address	+ 0x8	Buffer B memory end address	+ 0x8	Buffer A memory end address
+ 0xC	Address of descriptor B	+ 0xC	Address of descriptor A	+ 0xC	Address of descriptor B

In this example, the channel descriptor is used first, with a first buffer in memory called buffer A. The configuration of the DMA channel must have been set to indicate a reload. Similarly, both descriptor A and descriptor B must also specify reload. When the channel descriptor is exhausted, descriptor B is loaded using the link to descriptor B, and a transfer interrupt informs the CPU that buffer A is available.

Descriptor B is then used until it is also exhausted, when descriptor A is loaded using the link to descriptor A contained in descriptor B. Then a transfer interrupt informs the CPU that buffer B is available for processing. The process repeats when descriptor A is exhausted, alternately using each of the 2 memory buffers.

14.5.5 Interleaved transfers

One use for the SRCINC and DSTINC configurations (located in the channel transfer configuration registers, XFERCFGn) is to handle data in a buffer such that it is interleaved with other data.

For example, if 4 data samples from several peripherals need to be interleaved into a single data structure, this may be done while the data is being read in by the DMA. Setting SRCINC to 4x width for each channel involved will allow room for 4 samples in a row in the buffer memory. The DMA will place data for each successive value at the next location for that peripheral.

The reverse of this process could be done using DSTINC to de-interleave combined data from the buffer and send it to several peripherals or locations.

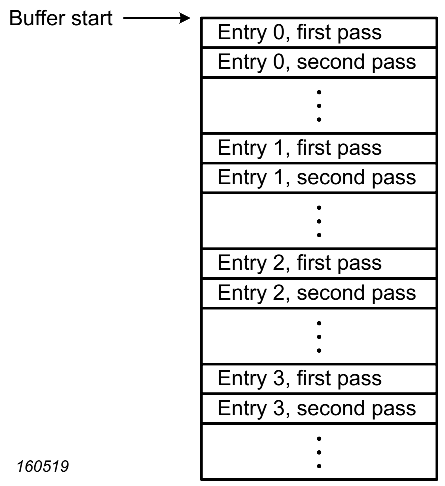


Fig 25. Interleaved transfer in a single buffer

14.5.6 Linked transfers (linked list)

A linked transfer can use any number of descriptors to define a complicated transfer. This can be configured such that a single transfer, a portion of a transfer, one whole descriptor, or an entire structure of links can be initiated by a single DMA request or trigger.

An example of a linked transfer could start out like the example for a ping-pong transfer (Table 240). The difference would be that descriptor B would not link back to descriptor A, but would continue on to another different descriptor. This could continue as long as desired, and can be ended anywhere, or linked back to any point to repeat a sequence of descriptors. Of course, any descriptor not currently in use can be altered by software as well.

14.5.7 Address alignment for data transfers

Transfers of 16 bit width require an address alignment to a multiple of 2 bytes. Transfers of 32 bit width require an address alignment to a multiple of 4 bytes. Transfers of 8 bit width can be at any address.

14.5.8 Channel chaining

Channel chaining is a feature which allows completion of a DMA transfer on channel x to trigger a DMA transfer on channel y. This feature can for example be used to have DMA channel x reading n bytes from UART to memory, and then have DMA channel y transferring the received bytes to the CRC engine, without any action required from the ARM core.

To use channel chaining, first configure DMA channels x and y as if no channel chaining would be used. Then:

- For channel x:
  - If channel x is configured to auto reload the descriptor on exhausting of the descriptor (bit RELOAD in the transfer configuration of the descriptor is set), then enable 'clear trigger on descriptor exhausted' by setting bit CLRTRIG in the channel's transfer configuration in the descriptor.
- For channel y:
  - Configure the input trigger input mux register (DMA_ITRIG_INMUX[0:21]) for channel y to use any of the available DMA trigger muxes (DMA trigger mux 0/1).
  - Configure the chosen DMA trigger mux to select DMA channel x.
  - Enable hardware triggering by setting bit HWTRIGEN in the channel configuration register.
  - Set the trigger type to edge sensitive by clearing bit TRIGTYPE in the channel configuration register.
  - Configure the trigger edge to falling edge by clearing bit TRIGPOL in the channel configuration register.

Note that after completion of channel x the descriptor may be reloaded (if configured so), but remains un-triggered. To configure the chain to auto-trigger itself, setup channels x and y for channel chaining as described above. In addition to that:

- A ping-pong configuration for both channel x and y is recommended, so that data currently moved by channel y is not altered by channel x.
- For channel x:
  - Configure the input trigger input mux register (DMA_ITRIG_INMUX[0:21]) for channel y to use the same DMA trigger mux as chosen for channel y.
  - Enable hardware triggering by setting bit HWTRIGEN in the channel configuration register.
  - Set the trigger type to edge sensitive by clearing bit TRIGTYPE in the channel configuration register.
  - Configure the trigger edge to falling edge by clearing bit TRIGPOL in the channel configuration register.

14.5.9 DMA in reduced power modes

DMA in sleep mode

In sleep mode, the DMA can operate and access all enabled SRAM blocks, without waking up the CPU.

DMA in deep-sleep mode

Some peripherals support DMA service during deep-sleep mode without waking up the CPU or the rest of the device. These peripherals are the Flexcomm Interface functions that include FIFO support (USART, SPI, and I2S).

These wake-ups are based on peripheral FIFO levels, not directly related to peripheral DMA requests and interrupts. See Section 6.5.54 for more information.

14.6 Register description

The DMA registers are grouped into DMA control, interrupt and status registers and DMA channel registers. DMA transfers are controlled by a set of three registers per channel, the CFG[0:19], CTRLSTAT[0:19], and XFERCFG[0:19] registers.

The reset value reflects the data stored in used bits only. It does not include the content of reserved bits.

Table 241. Register overview: DMA controller (base address 0x4008 2000)

Name	Access	Offset	Description	Reset value	Section
Global control and status registers					
CTRL	R/W	0x000	DMA control.	0x0	14.6.1
INTSTAT	RO	0x004	Interrupt status.	0x0	14.6.2
SRAMBASE	R/W	0x008	SRAM address of the channel configuration table.	0x0	14.6.3
Shared registers					
ENABLESET0	R/W	0x020	Channel Enable read and Set for all DMA channels.	0x0	14.6.4
ENABLECLR0	WO	0x028	Channel Enable Clear for all DMA channels.	-	14.6.5
ACTIVE0	RO	0x030	Channel Active status for all DMA channels.	0x0	14.6.6
BUSY0	RO	0x038	Channel Busy status for all DMA channels.	0x0	14.6.7
ERRINT0	R/W	0x040	Error Interrupt status for all DMA channels.	0x0	14.6.8
INTENSET0	R/W	0x048	Interrupt Enable read and Set for all DMA channels.	0x0	14.6.9
INTENCLR0	WO	0x050	Interrupt Enable Clear for all DMA channels.	-	14.6.10
INTA0	R/W	0x058	Interrupt A status for all DMA channels.	0x0	14.6.11
INTB0	R/W	0x060	Interrupt B status for all DMA channels.	0x0	14.6.12
SETVALID0	WO	0x068	Set ValidPending control bits for all DMA channels.	-	14.6.13
SETTRIG0	WO	0x070	Set Trigger control bits for all DMA channels.	-	14.6.14
ABORT0	WO	0x078	Channel Abort control for all DMA channels.	-	14.6.15
Channel 0 registers					
CFG0	R/W	0x400	Configuration register for DMA channel 0.	0x0	14.6.16
CTLSTAT0	RO	0x404	Control and status register for DMA channel 0.	0x0	14.6.17
XFERCFG0	R/W	0x408	Transfer configuration register for DMA channel 0.	0x0	14.6.18
Channel 1 registers					
CFG1	R/W	0x410	Configuration register for DMA channel 1.	0x0	14.6.16
CTLSTAT1	RO	0x414	Control and status register for DMA channel 1.	0x0	14.6.17
XFERCFG1	R/W	0x418	Transfer configuration register for DMA channel 1.	0x0	14.6.18
Channel 2 registers					
CFG2	R/W	0x420	Configuration register for DMA channel 2.	0x0	14.6.16
CTLSTAT2	RO	0x424	Control and status register for DMA channel 2.	0x0	14.6.17
XFERCFG2	R/W	0x428	Transfer configuration register for DMA channel 2.	0x0	14.6.18
Channel 3 registers					
CFG3	R/W	0x430	Configuration register for DMA channel 3.	0x0	14.6.16

Table 241. Register overview: DMA controller (base address 0x4008 2000)

Name	Access	Offset	Description	Reset value	Section
CTLSTAT3	RO	0x434	Control and status register for DMA channel 3.	0x0	14.6.17
XFERCFG3	R/W	0x438	Transfer configuration register for DMA channel 3.	0x0	14.6.18
Channel 4 registers					
CFG4	R/W	0x440	Configuration register for DMA channel 4.	0x0	14.6.16
CTLSTAT4	RO	0x444	Control and status register for DMA channel 4.	0x0	14.6.17
XFERCFG4	R/W	0x448	Transfer configuration register for DMA channel 4.	0x0	14.6.18
Channel 5 registers					
CFG5	R/W	0x450	Configuration register for DMA channel 5.	0x0	14.6.16
CTLSTAT5	RO	0x454	Control and status register for DMA channel 5.	0x0	14.6.17
XFERCFG5	R/W	0x458	Transfer configuration register for DMA channel 5.	0x0	14.6.18
Channel 6 registers					
CFG6	R/W	0x460	Configuration register for DMA channel 6.	0x0	14.6.16
CTLSTAT6	RO	0x464	Control and status register for DMA channel 6.	0x0	14.6.17
XFERCFG6	R/W	0x468	Transfer configuration register for DMA channel 6.	0x0	14.6.18
Channel 7 registers					
CFG7	R/W	0x470	Configuration register for DMA channel 7.	0x0	14.6.16
CTLSTAT7	RO	0x474	Control and status register for DMA channel 7.	0x0	14.6.17
XFERCFG7	R/W	0x478	Transfer configuration register for DMA channel 7.	0x0	14.6.18
Channel 8 registers					
CFG8	R/W	0x480	Configuration register for DMA channel 8.	0x0	14.6.16
CTLSTAT8	RO	0x484	Control and status register for DMA channel 8.	0x0	14.6.17
XFERCFG8	R/W	0x488	Transfer configuration register for DMA channel 8.	0x0	14.6.18
Channel 9 registers					
CFG9	R/W	0x490	Configuration register for DMA channel 9.	0x0	14.6.16
CTLSTAT9	RO	0x494	Control and status register for DMA channel 9.	0x0	14.6.17
XFERCFG9	R/W	0x498	Transfer configuration register for DMA channel 9.	0x0	14.6.18
Channel 10 registers					
CFG10	R/W	0x4A0	Configuration register for DMA channel 10.	0x0	14.6.16
CTLSTAT10	RO	0x4A4	Control and status register for DMA channel 10.	0x0	14.6.17
XFERCFG10	R/W	0x4A8	Transfer configuration register for DMA channel 10.	0x0	14.6.18
Channel 11 registers					
CFG11	R/W	0x4B0	Configuration register for DMA channel 11.	0x0	14.6.16
CTLSTAT11	RO	0x4B4	Control and status register for DMA channel 11.	0x0	14.6.17
XFERCFG11	R/W	0x4B8	Transfer configuration register for DMA channel 11.	0x0	14.6.18
Channel 12 registers					
CFG12	R/W	0x4C0	Configuration register for DMA channel 12.	0x0	14.6.16
CTLSTAT12	RO	0x4C4	Control and status register for DMA channel 12.	0x0	14.6.17
XFERCFG12	R/W	0x4C8	Transfer configuration register for DMA channel 12.	0x0	14.6.18
Channel 13 registers					
CFG13	R/W	0x4D0	Configuration register for DMA channel 13.	0x0	14.6.16
CTLSTAT13	RO	0x4D4	Control and status register for DMA channel 13.	0x0	14.6.17

Table 241. Register overview: DMA controller (base address 0x4008 2000)

Name	Access	Offset	Description	Reset value	Section
XFERCFG13	R/W	0x4D8	Transfer configuration register for DMA channel 13.	0x0	14.6.18
Channel 14 registers					
CFG14	R/W	0x4E0	Configuration register for DMA channel 14.	0x0	14.6.16
CTLSTAT14	RO	0x4E4	Control and status register for DMA channel 14.	0x0	14.6.17
XFERCFG14	R/W	0x4E8	Transfer configuration register for DMA channel 14.	0x0	14.6.18
Channel 15 registers					
CFG15	R/W	0x4F0	Configuration register for DMA channel 15.	0x0	14.6.16
CTLSTAT15	RO	0x4F4	Control and status register for DMA channel 15.	0x0	14.6.17
XFERCFG15	R/W	0x4F8	Transfer configuration register for DMA channel 15.	0x0	14.6.18
Channel 16 registers					
-	-	0x500	Reserved	-	-
-	-	0x504	Reserved	-	-
-	-	0x508	Reserved	-	-
Channel 17 registers					
-	-	0x510	Reserved	-	-
-	-	0x514	Reserved	-	-
-	-	0x518	Reserved	-	-
Channel 18 registers					
CFG18	R/W	0x520	Configuration register for DMA channel 18.	0x0	14.6.16
CTLSTAT18	RO	0x524	Control and status register for DMA channel 18.	0x0	14.6.17
XFERCFG18	R/W	0x528	Transfer configuration register for DMA channel 18.	0x0	14.6.18
Channel 19 registers					
CFG19	R/W	0x530	Configuration register for DMA channel 19.	0x0	14.6.16
CTLSTAT19	RO	0x534	Control and status register for DMA channel 19.	0x0	14.6.17
XFERCFG19	R/W	0x538	Transfer configuration register for DMA channel 19.	0x0	14.6.18

14.6.1 Control register

The CTRL register contains global the control bit for a enabling the DMA controller.

Table 242. Control register (CTRL, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value
0	ENABLE		DMA controller master enable.	0x0
		0	Disabled. The DMA controller is disabled. This clears any triggers that were asserted at the point when disabled, but does not prevent re-triggering when the DMA controller is re-enabled.	
		1	Enabled. The DMA controller is enabled.	
31:1	-	-	Reserved. Read value is undefined, only zero should be written.	-

14.6.2 Interrupt Status register

The Read-Only INTSTAT register provides an overview of DMA status. This allows quick determination of whether any enabled interrupts are pending. Details of which channels are involved are found in the interrupt type specific registers.

Table 243. Interrupt Status register (INTSTAT, offset 0x004) bit description

Bit	Symbol	Value	Description	Reset value
0	-	-	Reserved. Read value is undefined, only zero should be written.	-
1	ACTIVEINT		Summarizes whether any enabled interrupts (other than error interrupts) are pending.	0x0
		0	Not pending. No enabled interrupts are pending.	
		1	Pending. At least one enabled interrupt is pending.	
2	ACTIVEERRINT		Summarizes whether any error interrupts are pending.	0x0
		0	Not pending. No error interrupts are pending.	
		1	Pending. At least one error interrupt is pending.	
31:3	-	-	Reserved. Read value is undefined, only zero should be written.	-

14.6.3 SRAM Base address register

The SRAMBASE register must be configured with an address (preferably in on-chip SRAM) where DMA descriptors will be stored. Software must set up the descriptors for those DMA channels that will be used in the application.

Table 244. SRAM Base address register (SRAMBASE, offset 0x008) bit description

Bit	Symbol	Description	Reset value
8:0	-	Reserved. Read value is undefined, only zero should be written.	-
31:9	OFFSET	Address bits 31:9 of the beginning of the DMA descriptor table. For 18 channels, the table must begin on a 512 byte boundary.	0x0

Each DMA channel has an entry for the channel descriptor in the SRAM table. The values for each channel start at the address offsets found in Table 245. The contents of each channel descriptor are described in Table 245.

Table 245. Channel descriptor map

Descriptor	Table offset
Channel descriptor for DMA channel 0	0x000
Channel descriptor for DMA channel 1	0x010
Channel descriptor for DMA channel 2	0x020
Channel descriptor for DMA channel 3	0x030
Channel descriptor for DMA channel 4	0x040
Channel descriptor for DMA channel 5	0x050
Channel descriptor for DMA channel 6	0x060
Channel descriptor for DMA channel 7	0x070
Channel descriptor for DMA channel 8	0x080
Channel descriptor for DMA channel 9	0x090
Channel descriptor for DMA channel 10	0x0A0
Channel descriptor for DMA channel 11	0x0B0
Channel descriptor for DMA channel 12	0x0C0
Channel descriptor for DMA channel 13	0x0D0
Channel descriptor for DMA channel 14	0x0E0

Table 245. Channel descriptor map

Descriptor	Table offset
Channel descriptor for DMA channel 15	0x0F0
Channel descriptor for DMA channel 18	0x120
Channel descriptor for DMA channel 19	0x130

14.6.4 Enable read and Set registers

The ENABLESET0 register determines whether each DMA channel is enabled or disabled. Disabling a DMA channel does not reset the channel in any way. A channel can be paused and restarted by clearing, then setting the Enable bit for that channel.

Reading ENABLESET0 provides the current state of all of the DMA channels represented by that register. Writing a 1 to a bit position in ENABLESET0 that corresponds to an implemented DMA channel sets the bit, enabling the related DMA channel. Writing a 0 to any bit has no effect. Enables are cleared by writing to ENABLECLR0.

Table 246. Enable read and Set register 0 (ENABLESET0, offset 0x020) bit description

Bit	Symbol	Description	Reset value
31:0	ENA	Enable for DMA channels. Bit n enables or disables DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = disabled. 1 = enabled.	0x0

14.6.5 Enable Clear register

The ENABLECLR0 register is used to clear the enable of one or more DMA channels. This register is write-only.

Table 247. Enable Clear register 0 (ENABLECLR0, offset 0x028) bit description

Bit	Symbol	Description	Reset value
31:0	CLR	Writing ones to this register clears the corresponding bits in ENABLESET0. Bit n clears the channel enable bit n. The number of bits = number of DMA channels in this device. Other bits are reserved.	-

14.6.6 Active status register

The ACTIVE0 register indicates which DMA channels are active at the point when the read occurs. The register is read-only.

A DMA channel is considered active when a DMA operation has been started but not yet fully completed. The Active status will persist from a DMA operation being started, until the pipeline is empty after end of the last descriptor (when there is no reload). An active channel may be aborted by software by setting the appropriate bit in one of the Abort register (see Section 14.6.15).

Table 248. Active status register 0 (ACTIVE0, offset 0x030) bit description

Bit	Symbol	Description	Reset value
31:0	ACT	Active flag for DMA channel n. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = not active. 1 = active.	0x0

14.6.7 Busy status register

The BUSY0 register indicates which DMA channels is busy at the point when the read occurs. This registers is read-only.

A DMA channel is considered busy when there is any operation related to that channel in the DMA controller’s internal pipeline. This information can be used after a DMA channel is disabled by software (but still active), allowing confirmation that there are no remaining operations in progress for that channel.

Table 249. Busy status register 0 (BUSY0, offset 0x038) bit description

Bit	Symbol	Description	Reset value
31:0	BSY	Busy flag for DMA channel n. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = not busy. 1 = busy.	0x0

14.6.8 Error Interrupt register

The ERRINT0 register contains flags for each DMA channel’s Error Interrupt. Any pending interrupt flag in the register will be reflected on the DMA interrupt output.

Reading the registers provides the current state of all DMA channel error interrupts. Writing a 1 to a bit position in ERRINT0 that corresponds to an implemented DMA channel clears the bit, removing the interrupt for the related DMA channel. Writing a 0 to any bit has no effect.

Table 250. Error Interrupt register 0 (ERRINT0, offset 0x040) bit description

Bit	Symbol	Description	Reset value
31:0	ERR	Error Interrupt flag for DMA channel n. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = error interrupt is not active. 1 = error interrupt is active.	0x0

14.6.9 Interrupt Enable read and Set register

The INTENSET0 register controls whether the individual Interrupts for DMA channels contribute to the DMA interrupt output.

Reading the registers provides the current state of all DMA channel interrupt enables. Writing a 1 to a bit position in INTENSET0 that corresponds to an implemented DMA channel sets the bit, enabling the interrupt for the related DMA channel. Writing a 0 to any bit has no effect. Interrupt enables are cleared by writing to INTENCLR0.

Table 251. Interrupt Enable read and Set register 0 (INTENSET0, offset 0x048) bit description

Bit	Symbol	Description	Reset value
31: 0	INTEN	Interrupt Enable read and set for DMA channel n. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = interrupt for DMA channel is disabled. 1 = interrupt for DMA channel is enabled.	0x0

14.6.10 Interrupt Enable Clear register

The INTENCLR0 register is used to clear interrupt enable bits in INTENSET0. The register is write-only.

Table 252. Interrupt Enable Clear register 0 (INTENCLR0, offset 0x050) bit description

Bit	Symbol	Description	Reset value
31:0	CLR	Writing ones to this register clears corresponding bits in the INTENSET0. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved.	-

14.6.11 Interrupt A register

The IntA0 register contains the interrupt A status for each DMA channel. The status will be set when the SETINTA bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in this register clears the related INTA flag. Writing 0 has no effect. Any interrupt pending status in the registers will be reflected on the DMA interrupt output if it is enabled in the related INTENSET register.

Table 253. Interrupt A register 0 (INTA0, offset 0x058) bit description

Bit	Symbol	Description	Reset value
31:0	IA	Interrupt A status for DMA channel n. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = the DMA channel interrupt A is not active. 1 = the DMA channel interrupt A is active.	0x0

14.6.12 Interrupt B register

The INTB0 register contains the interrupt B status for each DMA channel. The status will be set when the SETINTB bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in the register clears the related INTB flag. Writing 0 has no effect. Any interrupt pending status in this register will be reflected on the DMA interrupt output if it is enabled in the INTENSET register.

Table 254. Interrupt B register 0 (INTB0, offset 0x060) bit description

Bit	Symbol	Description	Reset value
31:0	IB	Interrupt B status for DMA channel n. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = the DMA channel interrupt B is not active. 1 = the DMA channel interrupt B is active.	0x0

14.6.13 Set Valid register

The SETVALID0 register allows setting the Valid bit in the CTRLSTAT register for one or more DMA channels. See Section 14.6.17 for a description of the VALID bit. This register is write-only.

The CFGVALID and SV (set valid) bits allow more direct DMA block timing control by software. Each Channel Descriptor, in a sequence of descriptors, can be validated by either the setting of the CFGVALID bit or by setting the channel's SETVALID flag. Normally, the CFGVALID bit is set. This tells the DMA that the Channel Descriptor is active and can be executed. The DMA will continue sequencing through descriptor blocks whose CFGVALID bit are set without further software intervention. Leaving a CFGVALID bit set to 0 allows the DMA sequence to pause at the Descriptor until software triggers the continuation. If, during DMA transmission, a Channel Descriptor is found with CFGVALID set to 0, the DMA checks for a previously buffered SETVALID0 setting for the channel. If found, the DMA will set the descriptor valid, clear the SV setting, and resume processing the descriptor. Otherwise, the DMA pauses until the channels SETVALID0 bit is set.

Table 255. Set Valid 0 register (SETVALID0, offset 0x068) bit description

Bit	Symbol	Description	Reset value
31:0	SV	SETVALID control for DMA channel n. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = no effect. 1 = sets the VALIDPENDING control bit for DMA channel n	-

14.6.14 Set Trigger register

The SETTRIG0 register allows setting the TRIG bit in the CTRLSTAT register for one or more DMA channel. See Section 14.6.17 for a description of the TRIG bit, and Section 14.5.1 for a general description of triggering. This register is write-only.

Table 256. Set Trigger 0 register (SETTRIG0, offset 0x070) bit description

Bit	Symbol	Description	Reset value
31:0	TRIG	Set Trigger control bit for DMA channel 0. Bit n corresponds to DMA channel n. The number of bits = number of DMA channels in this device. Other bits are reserved. 0 = no effect. 1 = sets the TRIG bit for DMA channel n.	-

14.6.15 Abort register

The Abort0 register allows aborting operation of a DMA channel if needed. To abort a selected channel, the channel should first be disabled by clearing the corresponding Enable bit by writing a 1 to the proper bit ENABLECLR. Then wait until the channel is no longer busy by checking the corresponding bit in BUSY. Finally, write a 1 to the proper bit of ABORT. This prevents the channel from restarting an incomplete operation when it is enabled again. This register is write-only.

Table 257. Abort 0 register (ABORT0, offset 0x078) bit description

Bit	Symbol	Description	Reset value
31:0	ABORTCTRL	Abort control for DMA channel 0. Bit n corresponds to DMA channel n. 0 = no effect. 1 = aborts DMA operations on channel n.	-

14.6.16 Channel configuration registers

The CFGn register contains various configuration options for DMA channel n.

See Table 259 for a summary of trigger options.

Table 258. Channel configuration registers bit description

Bit	Symbol	Value	Description	Reset value
0	PERIPHREQEN		Peripheral request Enable. If a DMA channel is used to perform a memory-to-memory move, any peripheral DMA request associated with that channel can be disabled to prevent any interaction between the peripheral and the DMA controller.	0x0
		0	Disabled. Peripheral DMA requests are disabled.	
		1	Enabled. Peripheral DMA requests are enabled.	
1	HWTRIGEN		Hardware Triggering Enable for this channel.	0x0
		0	Disabled. Hardware triggering is not used.	
		1	Enabled. Use hardware triggering.	
3:2	-	-	Reserved. Read value is undefined, only zero should be written.	-
4	TRIGPOL		Trigger Polarity. Selects the polarity of a hardware trigger for this channel.	0x0
		0	Active low - falling edge. Hardware trigger is active low or falling edge triggered, based on TRIGTYPE.	
		1	Active high - rising edge. Hardware trigger is active high or rising edge triggered, based on TRIGTYPE.	

Table 258. Channel configuration registers bit description

Bit	Symbol	Value	Description	Reset value
5	TRIGTYPE		Trigger Type. Selects hardware trigger as edge triggered or level triggered.	0x0
		0	Edge. Hardware trigger is edge triggered. Transfers will be initiated and completed, as specified for a single trigger.	
		1	Level. Hardware trigger is level triggered. Note that when level triggering without burst (BURSTPOWER = 0) is selected, only hardware triggers should be used on that channel. Transfers continue as long as the trigger level is asserted. Once the trigger is deasserted, the transfer will be paused until the trigger is, again, asserted. However, the transfer will not be paused until any remaining transfers within the current BURSTPOWER length are completed.	
6	TRIGBURST		Trigger Burst. Selects whether hardware triggers cause exhaustion of a descriptor, or a burst of a size defined by BURSTPOWER.	0x0
		0	Single transfer. A hardware trigger causes one or more descriptors to be exhausted, stopping when it reaches the end all linked descriptors, or when a CLRTRIG is encountered in the transfer configuration register (XFERCFG).	
		1	Burst transfer. When the trigger for this channel is set to edge triggered, a hardware trigger causes a burst transfer, as defined by BURSTPOWER. When the trigger for this channel is set to level triggered, a hardware trigger causes transfers to continue as long as the trigger is asserted, unless the transfer is complete.	
7	-	-	Reserved. Read value is undefined, only zero should be written.	-
11:8	BURSTPOWER		Burst Power is used in two ways. It always selects the address wrap size when SRCBURSTWRAP and/or DSTBURSTWRAP modes are selected (see descriptions elsewhere in this register). When the TRIGBURST field elsewhere in this register = 1, Burst Power selects how many transfers are performed for each DMA trigger. This can be used, for example, with peripherals that contain a FIFO that can initiate a DMA operation when the FIFO reaches a certain level. 0000: Burst size = 1 (2^0). This corresponds to a single transfer. 0001: Burst size = 2 (2^1). 0010: Burst size = 4 (2^2). ... 1010: Burst size = 1024 (2^{10}). This corresponds to the maximum supported transfer count. others: not supported. The total transfer length as defined in the XFERCOUNT bits in the XFERCFG register must be an even multiple of the burst size. Note that the total number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field).	0x0
13:12	-	-	Reserved. Read value is undefined, only zero should be written.	-
14	SRCBURSTWRAP		Source Burst Wrap. When enabled, the source data address for the DMA is “wrapped”, meaning that the source address range for each burst will be the same. As an example, this could be used to read several sequential registers from a peripheral for each DMA burst, reading the same registers again for each burst.	0x0
		0	Disabled. Source burst wrapping is not enabled for this DMA channel.	
		1	Enabled. Source burst wrapping is enabled for this DMA channel.	

Table 258. Channel configuration registers bit description

Bit	Symbol	Value	Description	Reset value
15	DSTBURSTWRAP		Destination Burst Wrap. When enabled, the destination data address for the DMA is “wrapped”, meaning that the destination address range for each burst will be the same. As an example, this could be used to write several sequential registers to a peripheral for each DMA burst, writing the same registers again for each burst.	0x0
		0	Disabled. Destination burst wrapping is not enabled for this DMA channel.	
		1	Enabled. Destination burst wrapping is enabled for this DMA channel.	
18:16	CHPRIORITY		Priority of this channel when multiple DMA requests are pending. Eight priority levels are supported: 0x0 = highest priority. 0x7 = lowest priority.	0x0
31:19	-	-	Reserved. Read value is undefined, only zero should be written.	-

Table 259. Trigger setting summary

TrigBurst	TrigType	TrigPol	Description
0	0	0	Hardware DMA trigger is falling edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
0	0	1	Hardware DMA trigger is rising edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
0	1	0	Hardware DMA trigger is low level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
0	1	1	Hardware DMA trigger is high level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
1	0	0	Hardware DMA trigger is falling edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	0	1	Hardware DMA trigger is rising edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	1	0	Hardware DMA trigger is low level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	1	1	Hardware DMA trigger is high level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.

14.6.17 Channel control and status registers

The CTLSTATn register provides status flags specific to DMA channel n. These registers are read-only.

Table 260. Channel control and Status registers bit description

Bit	Symbol	Value	Description	Reset value
0	VALIDPENDING		Valid pending flag for this channel. This bit is set when a 1 is written to the corresponding bit in the related SETVALID register when CFGVALID = 1 for the same channel.	0x0
		0	No effect. No effect on DMA operation.	
		1	Valid pending.	
1	-	-	Reserved. Read value is undefined, only zero should be written.	-
2	TRIG		Trigger flag. Indicates that the trigger for this channel is currently set. This bit is cleared at the end of an entire transfer or upon reload when CLRTRIG = 1.	0x0
		0	Not triggered. The trigger for this DMA channel is not set. DMA operations will not be carried out.	
		1	Triggered. The trigger for this DMA channel is set. DMA operations will be carried out.	
31:3	-	-	Reserved. Read value is undefined, only zero should be written.	-

14.6.18 Channel transfer configuration registers

The XFERCFGn register contains transfer related configuration information for DMA channel n. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

See “Trigger operation detail” for details on trigger operation.

Table 261. Channel transfer configuration registers bit description

Bit	Symbol	Value	Description	Reset value
0	CFGVALID		Configuration Valid flag. This bit indicates whether the current descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled.	0x0
		0	Not valid. The current descriptor is not considered valid until validated by an associated SETVALID0 setting.	
		1	Valid. The current descriptor is considered valid.	
1	RELOAD		Indicates whether the channel’s control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers.	0x0
		0	Disabled. Do not reload the channels’ control structure when the current descriptor is exhausted.	
		1	Enabled. Reload the channels’ control structure when the current descriptor is exhausted.	
2	SWTRIG		Software Trigger.	0x0
		0	Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel.	
		1	Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.	

Table 261. Channel transfer configuration registers bit description

Bit	Symbol	Value	Description	Reset value
3	CLRTRIG		Clear Trigger.	0x0
		0	Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started.	
		1	Cleared. The trigger is cleared when this descriptor is exhausted.	
4	SETINTA		Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed.	0x0
		0	No effect.	
		1	Set. The INTA flag for this channel will be set when the current descriptor is exhausted.	
5	SETINTB		Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed.	0x0
		0	No effect.	
		1	Set. The INTB flag for this channel will be set when the current descriptor is exhausted.	
7:6	-	-	Reserved. Read value is undefined, only zero should be written.	-
9:8	WIDTH		Transfer width used for this DMA channel.	0x0
		0x0	8-bit. 8-bit transfers are performed (8-bit source reads and destination writes).	
		0x1	16-bit. 6-bit transfers are performed (16-bit source reads and destination writes).	
		0x2	32-bit. 32-bit transfers are performed (32-bit source reads and destination writes).	
		0x3	Reserved. Reserved setting, do not use.	
11:10	-	-	Reserved. Read value is undefined, only zero should be written.	-
13:12	SRCINC		Determines whether the source address is incremented for each DMA transfer.	0x0
		0x0	No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device.	
		0x1	1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory.	
		0x2	2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer.	
		0x3	4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.	

Table 261. Channel transfer configuration registers bit description

Bit	Symbol	Value	Description	Reset value
15:14	DSTINC		Determines whether the destination address is incremented for each DMA transfer.	0x0
		0x0	No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device.	
		0x1	1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory.	
		0x2	2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer.	
		0x3	4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.	
25:16	XFERCOUNT		Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). XFERCOUNT is used to count down during DMA transfer. When one DMA transfer is completed, XFERCOUNT decrements by 1. Example: The total number of DMA transfer to complete is N. The initial value for XFERCOUNT is N-1. XFERCOUNT = N - 1 means there are N transfers to complete XFEFCOUNT = N - 2 means there are N-1 transfers to complete … XFERCOUNT = 1 means there are 2 transfers to complete XFERCOUNT = 0 means there is 1 transfer to complete XFERCOUNT = 0x3FF means all transfers are completed Please note when all transfers are completed, XFERCOUNT value changes from 0 to 0x3FF. The last value 0x3FF doesn’t mean there are 1024 transfers left to complete. If the initial value for XFERCOUNT is 0x3FF (i.e. when the XferCFGn register is programmed), then it means there are 1024 transfers to complete. The size of each DMA transfer is determined by the WIDTH field of the same xFERCFGn register.	0x0
31:26	-	-	Reserved. Read value is undefined, only zero should be written.	-

UM11071

Chapter 15: LPC51U68 SCTimer/PWM (SCT)

Rev. 1.1 — 17 May 2018 **User manual**

15.1 How to read this chapter

The SCTimer/PWM is available on all LPC51U68 devices.

Remark: For a detailed description of SCTimer/PWM applications and code examples, see Ref. 2 “AN11538”.

15.2 Features

- The SCTimer/PWM supports:
  - Eight inputs.
  - Eight outputs.
  - Ten match/capture registers.
  - Ten events.
  - Ten states.
- Counter/timer features:
  - Each SCTimer is configurable as two 16-bit counters or one 32-bit counter.
  - Counters clocked by system clock or selected input.
  - Configurable as up counters or up-down counters.
  - Configurable number of match and capture registers. Up to 10 match and capture registers total.
  - Upon match and/or an input or output transition create the following events: interrupt; stop, limit, halt the timer or change counting direction; toggle outputs; change the state.
  - Counter value can be loaded into capture register triggered by a match or input/output toggle.
- PWM features:
  - Counters can be used in conjunction with match registers to toggle outputs and create time-proportioned PWM signals. PWM waveforms can change based on the current State.
  - Up to 8 single-edge or 4 dual-edge PWM outputs with independent duty cycle and common PWM cycle length.
- Event creation features:
  - In bi-directional mode, events can be enabled based on the count direction.
  - The following conditions define an event: a counter match condition, an input (or output) condition such as an rising or falling edge or level, a combination of match and/or input/output condition.
  - Selected events can limit, halt, start, or stop a counter or change its direction.
  - Events trigger state changes, output transitions, timer captures, interrupts, and DMA transactions.

  - Match register 0 can be used as an automatic limit.
  - In bi-directional mode, events can be enabled based on the count direction.
  - Match events can be held until another qualifying event occurs.
- State control features:
  - A state is defined by events that can happen in the state while the counter is running.
  - A state changes into another state as a result of an event.
  - Each event can be assigned to one or more states.
  - State variable allows sequencing across multiple counter cycles.

15.3 Basic configuration

Configure the SCT as follows:

- Enable the clock to the SCTimer/PWM (SCT) in the AHBCLKCTRL1 register (Section 6.5.17) to enable the register interface and the peripheral clock.
- Clear the SCT peripheral reset using the PRESETCTRL register (Section 6.5.10).
- The SCT provides an interrupt to the NVIC, see Table 79.
- Use the IOCON registers to connect the SCT outputs to external pins. See Chapter 9.
- The SCT DMA request lines are connected to the DMA trigger inputs via the DMA_ITRIG_PINMUX registers. See Section 10.6.2 “DMA trigger input mux registers 0 to 17”.

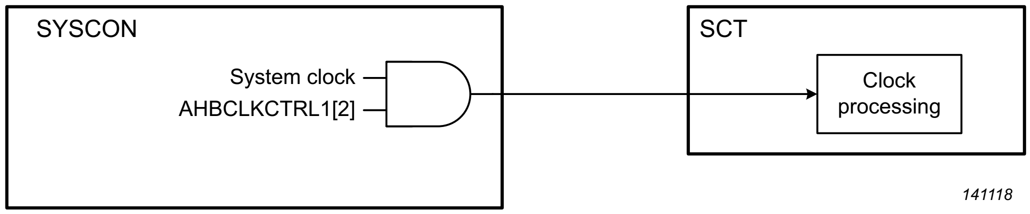


Fig 26. SCT clocking

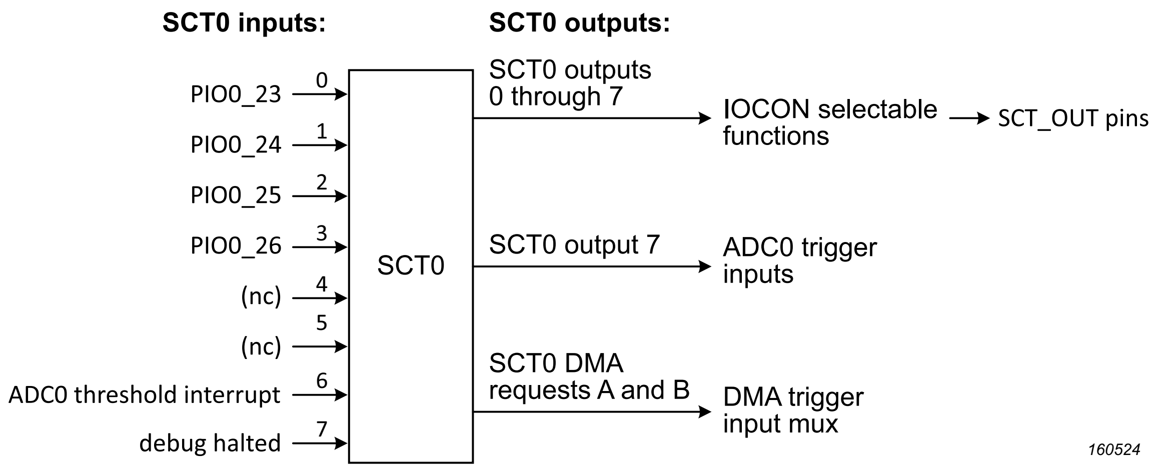


Fig 27. SCT connections

15.4 Pin description

See Chapter 9 to assign the SCT functions to external pins.

SCT input signals are predefined. The signals from external pins and internal signals are connected directly to the SCT inputs and not routed through IOCON.

SCT outputs can be routed to multiple places and can be connected to both a pin and an ADC trigger at the same time.

Table 262. SCT0 pin description (inputs)

Function	Type	Connect to	Reference
SCT0_IN[0:3]	external from pin	4 GPIO pins (PIO0_23 to PIO0_26)	Figure 27
	internal	ADC0_THCMP_IRQ, DEBUG_HALTED	Figure 27

Table 263. SCT0 pin description (outputs)

Type	Function	Connect to	Use register	Reference
external to pin	SCT0_OUT0	PIO0_7, PIO018	IOCON register for the related pin	See Chapter 9
	SCT0_OUT1	PIO0_1, PIO0_8, PIO0_19		
	SCT0_OUT2	PIO0_9, PIO0_29		
	SCT0_OUT3	PIO0_0, PIO0_10, PIO0_30		
	SCT0_OUT4	PIO0_13, PIO1_1, PIO1_10		
	SCT0_OUT5	PIO0_14, PIO1_2, PIO1_15		
	SCT0_OUT6	PIO0_5, PIO1_3		
	SCT0_OUT7	PIO1_4, PIO1_14		
internal	-	ADC0 trigger	SEQ_A, SEQ_B	Table 463, Table 464, Table 477

Recommended IOCON settings are shown in Table 264 and Table 265. See Chapter 9 for definitions of pin types.

Table 264: Suggested SCT input pin settings

IOCON bit(s)	Type D pin
10	I2CFILTER: Set to 1
9	I2CDRIVE: Set to 0.
8	FILTEROFF: Generally set to 1.
7	DIGIMODE: Set to 1.
6	INVERT: Set to 0.
5	I2CSLEW: Set to 1.
4:3	Not used, set to 0.
2:0	FUNC: not used, set to 0. Specific pin inputs are directly connected to the SCT.

Table 265: Suggested SCT output pin settings

IOCON bit(s)	Type D pin	Type A pin	Type I pin
10	OD: Set to 0 unless open-drain output is desired.	Same as type D.	I2CFILTER: Set to 1
9	SLEW: Set to 0.	Not used, set to 0	I2CDRIVE: Set to 0.
8	FILTEROFF: Set to 1.	Same as type D.	Same as type D.
7	DIGIMODE: Set to 1.	Same as type D.	Same as type D.
6	INVERT: Set to 0.	Same as type D.	Same as type D.
5	Not used, set to 0.	Same as type D.	I2CSLEW: Set to 1.
4:3	MODE: set to 0.	Same as type D.	Not used, set to 0.
2:0	FUNC: Must select the correct function for this peripheral.	Same as type D.	Same as type D.
General comment	A good choice for an SCT output.	A reasonable choice for an SCT output.	Not recommended for SCT outputs.

15.5 General description

The SCTimer/PWM is a powerful, flexible timer module capable of creating complex PWM waveforms and performing other advanced timing and control operations with minimal or no CPU intervention.

The SCT can operate as a single 32-bit counter or as two independent, 16-bit counters in uni-directional or bi-directional mode. As with most timers, the SCT supports a selection of match registers against which the count value can be compared, and capture registers where the current count value can be recorded when some pre-defined condition is detected.

An additional feature contributing to the versatility of the SCT is the concept of “events”. The SCT module supports multiple separate events that can be defined by the user based on some combination of parameters including a match on one of the match registers, and/or a transition on one of the SCT inputs or outputs, the direction of count, and other factors.

Every action that the SCT block can perform occurs in direct response to one if these user-defined events without any software overhead. Any event can be enabled to:

- Start, stop, or halt the counter.
- Limit the counter which means to clear the counter in unidirectional mode or change its direction in bi-directional mode.

- Set, clear, or toggle any SCT output.
- Force a capture of the count value into any capture registers.
- Generate an interrupt of DMA request.

The SCT allows the user to group and filter events, thereby selecting some events to be enabled together while others are disabled in a given context. A group of enabled and disabled events can be described as a state, and several states with different sets of enabled and disabled events are allowed. Changing from one state to another is event driven as well and can therefore happen without software intervention. By defining these states, the SCTimer/PWM provides the means to run entire state machines in hardware with any desired level of complexity to accomplish complex waveform and timing tasks.

In a simple system such as a classical timer/counter with capture and match capabilities. all events that could cause the timer to capture the timer value or toggle a match output are enabled and remain enabled at all times while the counter is running. In this case, no events are filtered and the system is described by one state that does not change. This is the default configuration of the SCT.

In a more complex system, two states could be set up that allow some events in one state and not in the other. An event itself, enabled in both states, can then be used, to move from one state to the other and back while filtering out events in either state. In such a two-state system different waveforms at the SCT output can be created depending on the event history. Changing between states is event-driven and happens without any intervention by the CPU.

Formally, the SCTimer/PWM can be programmed as state machine generator. The ability to perform switching between groups of events provides the SCT the unique capability to be utilized as a highly complex State Machine engine. Events identify the occurrence of conditions that warrant state changes and determine the next state to move to. This provides an extremely powerful control tool - particularly when the SCT inputs and outputs are connected to other on-chip resources (such as ADC triggers, other timers etc.) in addition to general-purpose I/O.

In addition to events and states, the SCTimer/PWM provides other enhanced features:

- Four alternative clocking modes including a fully asynchronous mode.
- Selection of any SCT input as a clock source or a clock gate.
- Capability of selecting a “greater-than-or-equal-to” match condition for the purpose of event generation.

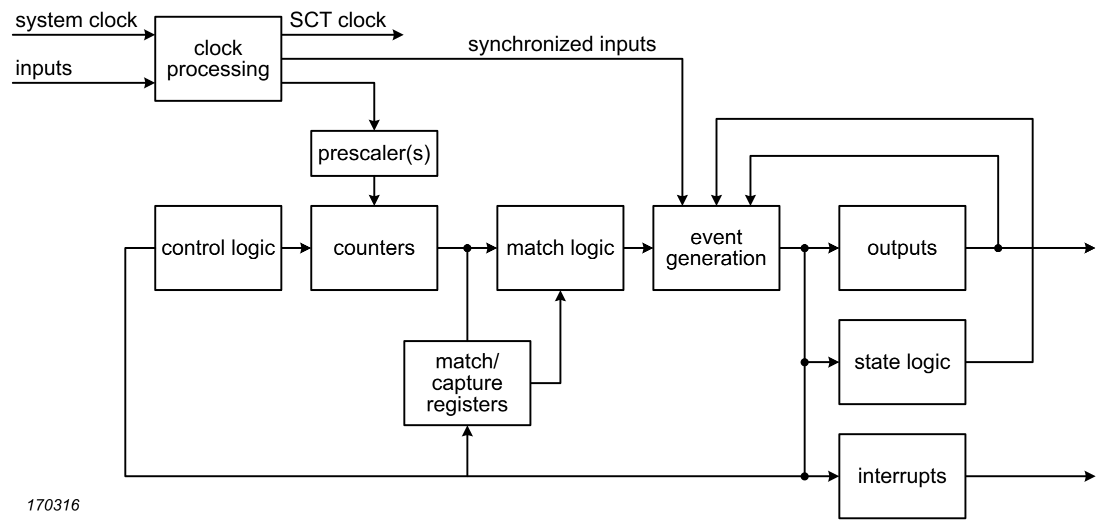


Fig 28. SCTimer/PWM block diagram

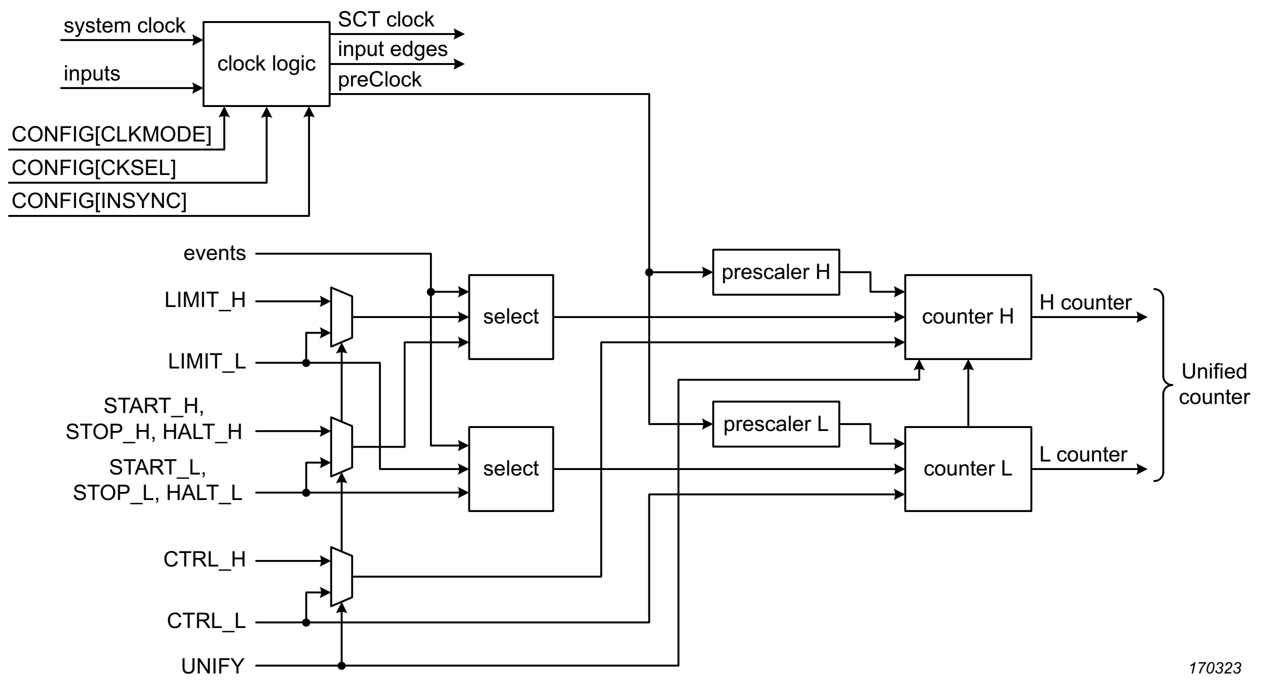


Fig 29. SCTimer/PWM counter and select logic

Remark: In this chapter, the term bus error indicates an SCT response that makes the processor take an exception.

15.6 Register description

The register addresses of the SCTimer/PWM are shown in Table 266. For most of the SCT registers, the register function depends on the setting of certain other register bits:

1. The UNIFY bit in the CONFIG register determines whether the SCT is used as one 32-bit register (for operation as one 32-bit counter/timer) or as two 16-bit counter/timers named L and H. The setting of the UNIFY bit is reflected in the register map:
   - UNIFY = 1: Only one register is used (for operation as one 32-bit counter/timer).
   - UNIFY = 0: Access the L and H registers by a 32-bit read or write operation or can be read or written to individually (for operation as two 16-bit counter/timers).

   Typically, the UNIFY bit is configured by writing to the CONFIG register before any other registers are accessed.
2. The REGMODEn bits in the REGMODE register determine whether each set of Match/Capture registers uses the match or capture functionality:
   - REGMODEn = 0: Registers operate as match and reload registers.
   - REGMODEn = 1: Registers operate as capture and capture control registers.

Table 266. Register overview: SCTimer/PWM (base address 0x4008 5000)

Name	Access	Offset	Description	Reset value	Section
CONFIG	R/W	0x000	SCT configuration register	0x0000 7E00	15.6.2
CTRL	R/W	0x004	SCT control register	0x0004 0004	15.6.3
CTRL_L	R/W	0x004	SCT control register low counter 16-bit	0x0004 0004	15.6.3
CTRL_H	R/W	0x006	SCT control register high counter 16-bit	0x0004 0004	15.6.3
LIMIT	R/W	0x008	SCT limit event select register	0x0	15.6.4
LIMIT_L	R/W	0x008	SCT limit event select register low counter 16-bit	0x0	15.6.4
LIMIT_H	R/W	0x00A	SCT limit event select register high counter 16-bit	0x0	15.6.4
HALT	R/W	0x00C	SCT halt event select register	0x0	15.6.5
HALT_L	R/W	0x00C	SCT halt event select register low counter 16-bit	0x0	15.6.5
HALT_H	R/W	0x00E	SCT halt event select register high counter 16-bit	0x0	15.6.5
STOP	R/W	0x010	SCT stop event select register	0x0	15.6.6
STOP_L	R/W	0x010	SCT stop event select register low counter 16-bit	0x0	15.6.6
STOP_H	R/W	0x012	SCT stop event select register high counter 16-bit	0x0	15.6.6
START	R/W	0x014	SCT start event select register	0x0	15.6.7
START_L	R/W	0x014	SCT start event select register low counter 16-bit	0x0	15.6.7
START_H	R/W	0x016	SCT start event select register high counter 16-bit	0x0	15.6.7
COUNT	R/W	0x040	SCT counter register	0x0	15.6.8
COUNT_L	R/W	0x040	SCT counter register low counter 16-bit	0x0	15.6.8
COUNT_H	R/W	0x042	SCT counter register high counter 16-bit	0x0	15.6.8
STATE	R/W	0x044	SCT state register	0x0	15.6.9
STATE_L	R/W	0x044	SCT state register low counter 16-bit	0x0	15.6.9
STATE_H	R/W	0x046	SCT state register high counter 16-bit	0x0	15.6.9
INPUT	RO	0x048	SCT input register	0x0	15.6.10
REGMODE	R/W	0x04C	SCT match/capture mode register	0x0	15.6.11

Table 266. Register overview: SCTimer/PWM (base address 0x4008 5000) *…continued*

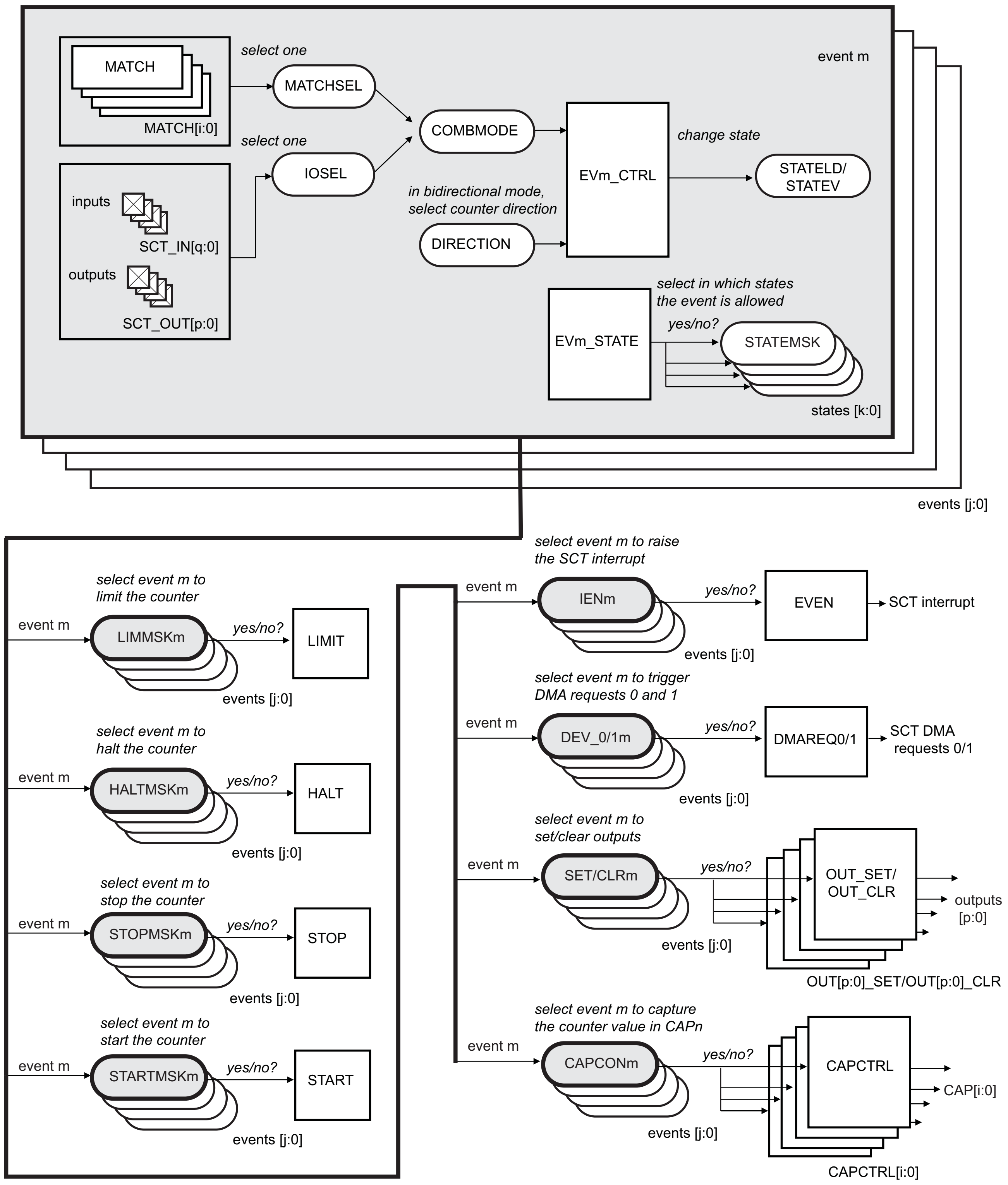
Name	Access	Offset	Description	Reset value	Section
REGMODE_L	R/W	0x04C	SCT match/capture mode register low counter 16-bit	0x0	15.6.11
REGMODE_H	R/W	0x04E	SCT match/capture registers mode register high counter 16-bit	0x0	15.6.11
OUTPUT	R/W	0x050	SCT output register	0x0	15.6.12
OUTPUTDIRCTRL	R/W	0x054	SCT output counter direction control register	0x0	15.6.13
RES	R/W	0x058	SCT conflict resolution register	0x0	15.6.14
DMAREQ0	R/W	0x05C	SCT DMA request 0 register	0x0	15.6.15
DMAREQ1	R/W	0x060	SCT DMA request 1 register	0x0	15.6.15
EVEN	R/W	0x0F0	SCT event interrupt enable register	0x0	15.6.16
EVFLAG	R/W	0x0F4	SCT event flag register	0x0	15.6.17
CONEN	R/W	0x0F8	SCT conflict interrupt enable register	0x0	15.6.18
CONFLAG	R/W	0x0FC	SCT conflict flag register	0x0	15.6.19
MATCH0 to MATCH9	R/W	0x100 to 0x124	SCT match value register of match channels 0 to 9; REGMODE0 to REGMODE9 = 0	0x0	15.6.20
MATCH0_L to MATCH9_L	R/W	0x100 to 0x124	SCT match value register of match channels 0 to 9; low counter 16-bit; REGMODE0_L to REGMODE9_L = 0	0x0	15.6.20
MATCH0_H to MATCH9_H	R/W	0x102 to 0x126	SCT match value register of match channels 0 to 9; high counter 16-bit; REGMODE0_H to REGMODE9_H = 0	0x0	15.6.20
CAP0 to CAP9	R/W	0x100 to 0x124	SCT capture register of capture channel 0 to 9; REGMODE0 to REGMODE9 = 1	0x0	15.6.21
CAP0_L to CAP9_L	R/W	0x100 to 0x124	SCT capture register of capture channel 0 to 9; low counter 16-bit; REGMODE0_L to REGMODE9_L = 1	0x0	15.6.21
CAP0_H to CAP9_H	R/W	0x102 to 0x126	SCT capture register of capture channel 0 to 9; high counter 16-bit; REGMODE0_H to REGMODE9_H = 1	0x0	15.6.21
MATCHREL0 to MATCHREL9	R/W	0x200 to 0x224	SCT match reload value register 0 to 9; REGMODE0 = 0 to REGMODE9 = 0	0x0	15.6.22
MATCHREL0_L to MATCHREL9_L	R/W	0x200 to 0x224	SCT match reload value register 0 to 9; low counter 16-bit; REGMODE0_L = 0 to REGMODE9_L = 0	0x0	15.6.22
MATCHREL0_H to MATCHREL9_H	R/W	0x202 to 0x226	SCT match reload value register 0 to 9; high counter 16-bit; REGMODE0_H = 0 to REGMODE9_H = 0	0x0	15.6.22
CAPCTRL0 to CAPCTRL9	R/W	0x200 to 0x224	SCT capture control register 0 to 9; REGMODE0 = 1 to REGMODE9 = 1	0x0	15.6.23
CAPCTRL0_L to CAPCTRL9_L	R/W	0x200 to 0x224	SCT capture control register 0 to 9; low counter 16-bit; REGMODE0_L = 1 to REGMODE9_L = 1	0x0	15.6.23
CAPCTRL0_H to CAPCTRL9_H	R/W	0x202 to 0x226	SCT capture control register 0 to 9; high counter 16-bit; REGMODE0 = 1 to REGMODE9 = 1	0x0	15.6.23
EV0_STATE	R/W	0x300	SCT event state register 0	0x0	15.6.24
EV0_CTRL	R/W	0x304	SCT event control register 0	0x0	15.6.25
EV1_STATE	R/W	0x308	SCT event state register 1	0x0	15.6.24
EV1_CTRL	R/W	0x30C	SCT event control register 1	0x0	15.6.25
EV2_STATE	R/W	0x310	SCT event state register 2	0x0	15.6.24
EV2_CTRL	R/W	0x314	SCT event control register 2	0x0	15.6.25
EV3_STATE	R/W	0x318	SCT event state register 3	0x0	15.6.24
EV3_CTRL	R/W	0x31C	SCT event control register 3	0x0	15.6.25

Table 266. Register overview: SCTimer/PWM (base address 0x4008 5000) *…continued*

Name	Access	Offset	Description	Reset value	Section
EV4_STATE	R/W	0x320	SCT event state register 4	0x0	15.6.24
EV4_CTRL	R/W	0x324	SCT event control register4	0x0	15.6.25
EV5_STATE	R/W	0x328	SCT event state register 5	0x0	15.6.24
EV5_CTRL	R/W	0x32C	SCT event control register 5	0x0	15.6.25
EV6_STATE	R/W	0x330	SCT event state register 6	0x0	15.6.24
EV6_CTRL	R/W	0x334	SCT event control register 6	0x0	15.6.25
EV7_STATE	R/W	0x338	SCT event state register 7	0x0	15.6.24
EV7_CTRL	R/W	0x33C	SCT event control register 7	0x0	15.6.25
EV8_STATE	R/W	0x340	SCT event state register 8	0x0	15.6.24
EV8_CTRL	R/W	0x344	SCT event control register 8	0x0	15.6.25
EV9_STATE	R/W	0x348	SCT event state register 9	0x0	15.6.24
EV9_CTRL	R/W	0x34C	SCT event control register 9	0x0	15.6.25
OUT0_SET	R/W	0x500	SCT output 0 set register	0x0	15.6.26
OUT0_CLR	R/W	0x504	SCT output 0 clear register	0x0	15.6.27
OUT1_SET	R/W	0x508	SCT output 1 set register	0x0	15.6.26
OUT1_CLR	R/W	0x50C	SCT output 1 clear register	0x0	15.6.27
OUT2_SET	R/W	0x510	SCT output 2 set register	0x0	15.6.26
OUT2_CLR	R/W	0x514	SCT output 2 clear register	0x0	15.6.27
OUT3_SET	R/W	0x518	SCT output 3 set register	0x0	15.6.26
OUT3_CLR	R/W	0x51C	SCT output 3 clear register	0x0	15.6.27
OUT4_SET	R/W	0x520	SCT output 4 set register	0x0	15.6.26
OUT4_CLR	R/W	0x524	SCT output 4 clear register	0x0	15.6.27
OUT5_SET	R/W	0x528	SCT output 5 set register	0x0	15.6.26
OUT5_CLR	R/W	0x52C	SCT output 5 clear register	0x0	15.6.27
OUT6_SET	R/W	0x530	SCT output 6 set register	0x0	15.6.26
OUT6_CLR	R/W	0x534	SCT output 6 clear register	0x0	15.6.27
OUT7_SET	R/W	0x538	SCT output 7 set register	0x0	15.6.26
OUT7_CLR	R/W	0x53C	SCT output 7 clear register	0x0	15.6.27

15.6.1 Register functional grouping

Most SCT registers either configure an event or select an event for a specific action of the counter (or counters) and outputs. Figure 30 shows the registers and register bits that need to be configured for each event.



Note: in this figure, letters are used to represent the maximum quantity of certain SCTimer/PWM features as noted below.

i = match/captures, j = events, k = states, p = outputs, q = inputs

Fig 30. SCT event configuration and selection registers

15.6.1.1 Counter configuration and control registers

The SCT contains two registers for configuring the SCT and monitor and control its operation by software.

- The configuration register (CONFIG) configures the SCT in single, 32-bit counter mode or in dual, 16-bit counter mode, configures the clocking and clock synchronization, and configures automatic limits and the use of reload registers.
- The control register (CTRL) allows to monitor and set the counter direction, and to clear, start, stop, or halt the 32-bit counter or each individual 16-bit counter if in dual-counter mode.

15.6.1.2 Event configuration registers

Each event is associated with two registers:

- One EVn_CTRL register per event to define what triggers the event.
- One EVn_STATE register per event to enable the event.

15.6.1.3 Match and capture registers

The SCT includes a set of registers to store the SCT match or capture values. Each match register is associated with a match reload register which automatically reloads the match register at the beginning of each counter cycle. This register group includes the following registers:

- One REGMODE register per match/capture register to configure each match/capture register for either storing a match value or a capture value.
- A set of match/capture registers with each register, depending on the setting of REGMODE, either storing a match value or a counter value.
- One reload register for each match register.

15.6.1.4 Event select registers for the counter operations

This group contains the registers that select the events which affect the counter. Counter actions are limit, halt, and start or stop and apply to the unified counter or to the two 16-bit counters. Also included is the counter register with the counter value, or values in the dual-counter set-up. This register group includes the following registers:

- LIMIT selects the events that limit the counter.
- START and STOP select events that start or stop the counter.
- HALT selects events that halt the counter: HALT
- COUNT contains the counter value.

The LIMIT, START, STOP, and HALT registers each contain one bit per event that selects for each event whether the event limits, stops, starts, or halts the counter, or counters in dual-counter mode.

In the dual-counter mode, the events can be selected independently for each counter.

15.6.1.5 Event select registers for setting or clearing the outputs

This group contains the registers that select the events which affect the level of each SCT output. Also included are registers to manage conflicts that occur when events try to set or clear the same output. This register group includes the following registers:

- One OUTn_SET register for each output to select the events which set the output.
- One OUTn_CLR register for each output to select the events which clear the output.
- The conflict resolution register which defines an action when more than one event try to control an output at the same time.
- The conflict flag and conflict interrupt enable registers that monitor interrupts arising from output set and clear conflicts.
- The output direction control register that interchanges the set and clear output operation caused by an event in bi-directional mode.

The OUTn_SET and OUTn_CLR registers each contain one bit per event that selects whether the event changes the state a given output n.

In the dual-counter mode, the events can be selected independently for each output.

15.6.1.6 Event select registers for capturing a counter value

This group contains registers that select events which capture the counter value and store it in one of the CAP registers. Each capture register m has one associated CAPCTRLm register which in turn selects the events to capture the counter value.

15.6.1.7 Event select register for initiating DMA transfers

One register is provided for each of the two DMA requests to select the events that can trigger a DMA request.

The DMAREQn register contain one bit for each event that selects whether this event triggers a DMA request. An additional bit enables the DMA trigger when the match registers are reloaded.

15.6.1.8 Interrupt handling registers

The following registers provide flags that are set by events and select the events that when they occur request an interrupt.

- The event flag register provides one flag for each event that is set when the event occurs.
- The event flag interrupt enable register provides one bit for each event to be enabled for the SCT interrupt.

15.6.1.9 Registers for controlling SCT inputs and outputs by software

Two registers are provided that allow software (as opposed to events) to set input and outputs of the SCT:

- The SCT input register to read the state of any of the SCT inputs.
- The SCT output register to set or clear any of the SCT outputs or to read the state of the outputs.

15.6.2 SCT configuration register

This register configures the overall operation of the SCT. Write to this register before any other registers. Only word-writes are permitted to this register. Attempting to write a half-word value results in a bus error.

Table 267. SCT configuration register (CONFIG, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value
0	UNIFY		SCT operation	0x0
		0	The SCT operates as two 16-bit counters named COUNTER_L and COUNTER_H.	
		1	The SCT operates as a unified 32-bit counter.	
2:1	CLKMODE		SCT clock mode	0x0
		0x0	System Clock Mode. The system clock clocks the entire SCT module including the counter(s) and counter prescalers.	
		0x1	Sampled System Clock Mode. The system clock clocks the SCT module, but the counter and prescalers are only enabled to count when the designated edge is detected on the input selected by the CKSEL field. The minimum pulse width on the selected clock-gate input is 1 bus clock period. This mode is the high-performance, sampled-clock mode.	
		0x2	SCT Input Clock Mode. The input/edge selected by the CKSEL field clocks the SCT module, including the counters and prescalers, after first being synchronized to the system clock. The minimum width of the positive and negative phases of the clock input must each be greater than one full period of the bus/system clock.	
		0x3	Asynchronous Mode. The entire SCT module is clocked directly by the input/edge selected by the CKSEL field. In this mode, the SCT outputs are switched synchronously to the SCT input clock - not the system clock. The input clock rate must be at least half the system clock rate and can be the same or faster than the system clock.	
6:3	CKSEL		SCT clock select. The specific functionality of the designated input/edge is dependent on the CLKMODE bit selection in this register.	0x0
		0x0	Rising edges on input 0.	
		0x1	Falling edges on input 0.	
		0x2	Rising edges on input 1.	
		0x3	Falling edges on input 1.	
		0x4	Rising edges on input 2.	
		0x5	Falling edges on input 2.	
		0x6	Rising edges on input 3.	
		0x7	Falling edges on input 3.	
		0x8	Rising edges on input 4.	
		0x9	Falling edges on input 4.	
		0xA	Rising edges on input 5.	
		0xB	Falling edges on input 5.	
		0xC	Rising edges on input 6.	
		0xD	Falling edges on input 6.	
		0xE	Rising edges on input 7.	
		0XF	Falling edges on input 7.	

Table 267. SCT configuration register (CONFIG, offset 0x000) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
7	NORELOAD_L	-	A 1 in this bit prevents the lower match registers from being reloaded from their respective reload registers. Setting this bit eliminates the need to write to the reload registers MATCHREL if the match values are fixed. Software can write to set or clear this bit at any time. This bit applies to both the higher and lower registers when the UNIFY bit is set.	0x0
8	NORELOAD_H	-	A 1 in this bit prevents the higher match registers from being reloaded from their respective reload registers. Setting this bit eliminates the need to write to the reload registers MATCHREL if the match values are fixed. Software can write to set or clear this bit at any time. This bit is not used when the UNIFY bit is set.	0x0
12:9	INSYNC	-	Synchronization for input N (bit 9 = input 0, bit 10 = input 1,..., bit 12 = input 3); all other bits are reserved. A 1 in one of these bits subjects the corresponding input to synchronization to the SCT clock, before it is used to create an event. This synchronization injects a two SCT-clock delay in the input path. Clearing this bit bypasses synchronization on the corresponding input. This bit may be cleared for faster input response time if both of the following conditions are met (for all Clock Modes): The corresponding input is already synchronous to the SCT clock. The SCT clock frequency does not exceed 100 MHz. Note: The SCT clock is the bus/system clock for CKMODE 0-2 or the selected, asynchronous input clock for CKMODE3. Alternatively, for CKMODE2 only, it is also allowable to bypass synchronization if both of the following conditions are met: The corresponding input is synchronous to the designated CKMODE2 input clock. The CKMODE2 input clock frequency is less than one-third the frequency of the bus/system clock.	0xF
16:13	-	-	Reserved.	-
17	AUTOLIMIT_L	-	This bit applies to the lower registers when the UNIFY bit = 0, and both the higher and lower registers when the UNIFY bit is set. Software can write to set or clear this bit at any time. A one in this bit causes a match on match register 0 to be treated as a de-facto LIMIT condition without the need to define an associated event. As with any LIMIT event, this automatic limit causes the counter to be cleared to zero in unidirectional mode or to change the direction of count in bi-directional mode.	0x0
18	AUTOLIMIT_H	-	This bit applies to the upper registers when the UNIFY bit = 0, and is not used when the UNIFY bit is set. Software can write to set or clear this bit at any time. A one in this bit will cause a match on match register 0 to be treated as a de-facto LIMIT condition without the need to define an associated event. As with any LIMIT event, this automatic limit causes the counter to be cleared to zero in unidirectional mode or to change the direction of count in bi-directional mode.	0x0
31:19	-	-	Reserved	-

15.6.3 SCT control register

If bit UNIFY = 1 in the CONFIG register, only the _L bits are used.

If bit UNIFY = 0 in the CONFIG register, this register can be written to as two registers CTRL_L and CTRL_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

All bits in this register can be written to when the counter is stopped or halted. When the counter is running, the only bits that can be written are STOP or HALT. (Other bits can be written in a subsequent write after HALT is set to 1.)

Remark: If CLKMODE = 0x3 is selected, wait at least 12 system clock cycles between a write access to the H, L or unified version of this register and the next write access. This restriction does not apply when writing to the HALT bit or bits and then writing to the CTRL register again to restart the counters - for example because software must update the MATCH register, which is only allowed when the counters are halted.

Remark: If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Table 268. SCT control register (CTRL, offset 0x004) bit description

Bit	Symbol	Value	Description	Reset value
0	DOWN_L	-	This read-only bit is 1 when the L or unified counter is counting down. Hardware sets this bit when the counter is counting up, counter limit occurs, and BIDIR = 1.Hardware clears this bit when the counter is counting down and a limit condition occurs or when the counter reaches 0.	0x0
1	STOP_L	-	When this bit is 1 and HALT is 0, the L or unified counter does not run, but I/O events related to the counter can occur. If a designated start event occurs, this bit is cleared and counting resumes.	0x0
2	HALT_L	-	When this bit is 1, the L or unified counter does not run and no events can occur. A reset sets this bit. When the HALT_L bit is one, the STOP_L bit is cleared. It is possible to remove the halt condition while keeping the SCT in the stop condition (not running) with a single write to this register to simultaneously clear the HALT bit and set the STOP bit. **Remark:** Once set, only software can clear this bit to restore counter operation. This bit is set on reset.	0x1
3	CLRCTR_L	-	When the counter halted (not just stopped), writing a 1 to this bit will clear the L or unified counter. This bit always reads as 0.	0x0
4	BIDIR_L		L or unified counter direction select	0x0
		0	Up. The counter counts up to a limit condition, then is cleared to zero.	
		1	Up-down. The counter counts up to a limit, then counts down to a limit condition or to 0.	
12:5	PRE_L	-	Specifies the factor by which the SCT clock is prescaled to produce the L or unified counter clock. The counter clock is clocked at the rate of the SCT clock divided by PRE_L+1. **Remark:** Clear the counter (by writing a 1 to the CLRCTR bit) whenever changing the PRE value.	0x0
15:13	-	-	Reserved	-

Table 268. SCT control register (CTRL, offset 0x004) bit description

Bit	Symbol	Value	Description	Reset value
16	DOWN_H	-	This read-only bit is 1 when the H counter is counting down. Hardware sets this bit when the counter is counting, a counter limit condition occurs, and BIDIR is 1. Hardware clears this bit when the counter is counting down and a limit condition occurs or when the counter reaches 0.	0x0
17	STOP_H	-	When this bit is 1 and HALT is 0, the H counter does not, run but I/O events related to the counter can occur. If such an event matches the mask in the Start register, this bit is cleared and counting resumes.	0x0
18	HALT_H	-	When this bit is 1, the H counter does not run and no events can occur. A reset sets this bit. When the HALT_H bit is one, the STOP_H bit is cleared. It is possible to remove the halt condition while keeping the SCT in the stop condition (not running) with a single write to this register to simultaneously clear the HALT bit and set the STOP bit. **Remark:** Once set, this bit can only be cleared by software to restore counter operation. This bit is set on reset.	0x1
19	CLRCTR_H	-	When the counter halted (not just stopped), writing a 1 to this bit will clear the H counter. This bit always reads as 0.	0x0
20	BIDIR_H		Direction select	0x0
		0	The H counter counts up to its limit condition, then is cleared to zero.	
		1	The H counter counts up to its limit, then counts down to a limit condition or to 0.	
28:21	PRE_H	-	Specifies the factor by which the SCT clock is prescaled to produce the H counter clock. The counter clock is clocked at the rate of the SCT clock divided by PRELH+1. **Remark:** Clear the counter (by writing a 1 to the CLRCTR bit) whenever changing the PRE value.	0x0
31:29	-	-	Reserved	-

15.6.4 SCT limit event select register

The running counter can be limited by an event. When any of the events selected in this register occur, the counter is cleared to zero from its current value or changes counting direction if in bi-directional mode.

Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit causes its associated event to serve as a LIMIT event. When any limit event occurs, the counter is reset to zero in uni-directional mode or changes its direction of count in bi-directional mode and keeps running.To define the actual limiting event (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Remark: Counting up to all ones or counting down to zero is always equivalent to a limit event occurring.

Note that in addition to using this register to specify events that serve as limits, it is also possible to automatically cause a limit condition whenever a match register 0 match occurs. This eliminates the need to define an event for the sole purpose of creating a limit. The AUTOLIMITL and AUTOLIMITH bits in the configuration register enable/disable this feature (see Table 267).

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers LIMIT_L and LIMIT_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Table 269. SCT limit event select register (LIMIT, offset 0x008) bit description

Bit	Symbol	Description	Reset value
15:0	LIMMSK_L	If bit n is one, event n is used as a counter limit for the L or unified counter (event 0 = bit 0, event 1 = bit 1, ...). The number of bits = number of events supported by this SCT.	0x0
31:16	LIMMSK_H	If bit n is one, event n is used as a counter limit for the H counter (event 0 = bit 16, event 1 = bit 17, ...). The number of bits = number of events supported by this SCT.	0x0

15.6.5 SCT halt event select register

The running counter can be disabled (halted) by an event. When any of the events selected in this register occur, the counter stops running and all further events are disabled.

Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit will cause its associated event to serve as a HALT event. To define the actual events that cause the counter to halt (a match, an I/O pin toggle, etc.), see the EVn_CTRL registers.

Remark: A HALT condition can only be removed when software clears the HALT bit in the CTRL register (Table 268).

If UNIFY = 1 in the CONFIG register, only the L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers HALT_L and HALT_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Table 270. SCT halt event select register (HALT, offset 0x00C) bit description

Bit	Symbol	Description	Reset value
15:0	HALTMSK_L	If bit n is one, event n sets the HALT_L bit in the CTRL register (event 0 = bit 0, event 1 = bit 1, ...). The number of bits = number of events supported by this SCT.	0x0
31:16	HALTMSK_H	If bit n is one, event n sets the HALT_H bit in the CTRL register (event 0 = bit 16, event 1 = bit 17, ...). The number of bits = number of events supported by this SCT.	0x0

15.6.6 SCT stop event select register

The running counter can be stopped by an event. When any of the events selected in this register occur, counting is suspended, that is the counter stops running and remains at its current value. Event generation remains enabled, and any event selected in the START register such as an I/O event or an event generated by the other counter can restart the counter.

This register specifies which events stop the counter. Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit will cause its associated event to serve as a STOP event. To define the actual event that causes the counter to stop (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Remark: Software can stop and restart the counter by writing to the CTRL register.

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers STOPT_L and STOP_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Table 271. SCT stop event select register (STOP, offset 0x010) bit description

Bit	Symbol	Description	Reset value
15:0	STOPMSK_L	If bit n is one, event n sets the STOP_L bit in the CTRL register (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events supported by this SCT.	0x0
31:16	STOPMSK_H	If bit n is one, event n sets the STOP_H bit in the CTRL register (event 0 = bit 16, event 1 = bit 17, …). The number of bits = number of events supported by this SCT.	0x0

15.6.7 SCT start event select register

The stopped counter can be re-started by an event. When any of the events selected in this register occur, counting is restarted from the current counter value.

Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit will cause its associated event to serve as a START event. When any START event occurs, hardware will clear the STOP bit in the Control Register CTRL. Note that a START event has no effect on the HALT bit. Only software can remove a HALT condition. To define the actual event that starts the counter (an I/O pin toggle or an event generated by the other running counter in dual-counter mode), see the EVn_CTRL register.

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers START_L and START_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Table 272. SCT start event select register (START, offset 0x014) bit description

Bit	Symbol	Description	Reset value
15:0	STARTMSK_L	If bit n is one, event n clears the STOP_L bit in the CTRL register (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events supported by this SCT.	0x0
31:16	STARTMSK_H	If bit n is one, event n clears the STOP_H bit in the CTRL register (event 0 = bit 16, event 1 = bit 17, …). The number of bits = number of events supported by this SCT.	0x0

15.6.8 SCT counter register

If UNIFY = 1 in the CONFIG register, the counter is a unified 32-bit register and both the _L and _H bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers COUNT_L and COUNT_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation. In this case, the L and H registers count independently under the control of the other registers.

Writing to the COUNT_L, COUNT_H, or unified register is only allowed when the corresponding counter is halted (HALT bits are set to 1 in the CTRL register). Attempting to write to the counter when it is not halted causes a bus error. Software can read the counter registers at any time.

Table 273. SCT counter register (COUNT, offset 0x040) bit description

Bit	Symbol	Description	Reset value
15:0	CTR_L	When UNIFY = 0, read or write the 16-bit L counter value. When UNIFY = 1, read or write the lower 16 bits of the 32-bit unified counter.	0x0
31:16	CTR_H	When UNIFY = 0, read or write the 16-bit H counter value. When UNIFY = 1, read or write the upper 16 bits of the 32-bit unified counter.	0x0

15.6.9 SCT state register

Each group of enabled and disabled events is assigned a number called the state variable. For example, a state variable with a value of 0 could have events 0, 2, and 3 enabled and all other events disabled. A state variable with the value of 1 could have events 1, 4, and 5 enabled and all others disabled.

Remark: The EVm_STATE registers define which event is enabled in each group.

Software can read the state associated with a counter at any time. Writing to the STATE_L, STATE_H, or unified register is only allowed when the corresponding counter is halted (HALT bits are set to 1 in the CTRL register).

The state variable is the main feature that distinguishes the SCTimer/PWM from other counter/timer/ PWM blocks. Events can be made to occur only in certain states. Events, in turn, can perform the following actions:

- set and clear outputs
- limit, stop, and start the counter
- cause interrupts and DMA requests
- modify the state variable

The value of a state variable is completely under the control of the application. If an application does not use states, the value of the state variable remains zero, which is the default value.

A state variable can be used to track and control multiple cycles of the associated counter in any desired operational sequence. The state variable is logically associated with a state machine diagram which represents the SCT configuration. See Section 15.6.24 and 15.6.25 for more about the relationship between states and events.

The STATELD/STADEV fields in the event control registers of all defined events set all possible values for the state variable. The change of the state variable during multiple counter cycles reflects how the associated state machine moves from one state to the next.

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers STATE_L and STATE_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Table 274. SCT state register (STATE, offset 0x044) bit description

Bit	Symbol	Description	Reset value
4:0	STATE_L	State variable.	0x0
15:5	-	Reserved.	-
20:16	STATE_H	State variable.	0x0
31:21	-	Reserved.	-

15.6.10 SCT input register

Software can read the state of the SCT inputs in this read-only register in slightly different forms.

1. The AIN bit displays the state of the input captured on each rising edge of the SCT clock This corresponds to a nearly direct read-out of the input but can cause spurious fluctuations in case of an asynchronous input signal.
2. The SIN bit displays the form of the input as it is used for event detection. This may include additional stages of synchronization, depending on what is specified for that input in the INSYNC field in the CONFIG register:
   – If the INSYNC bit is set for the input, the input is triple-synchronized to the SCT clock resulting in a stable signal that is delayed by three SCT clock cycles.
   – If the INSYNC bit is not set, the SIN bit value is identical to the AIN bit value.

Table 275. SCT input register (INPUT, offset 0x048) bit description

Bit	Symbol	Description	Reset value
0	AIN0	Input 0 state. Input 0 state on the last SCT clock edge.	-
1	AIN1	Input 1 state. Input 1 state on the last SCT clock edge.	-
2	AIN2	Input 2 state. Input 2 state on the last SCT clock edge.	-
3	AIN3	Input 3 state. Input 3 state on the last SCT clock edge.	-
15:4	AIN…	Input state for the remainder of inputs implemented in this SCT.	-
16	SIN0	Input 0 state. Input 0 state following the synchronization specified by INSYNC0.	-
17	SIN1	Input 1 state. Input 1 state following the synchronization specified by INSYNC0.	-
18	SIN2	Input 2 state. Input 2 state following the synchronization specified by INSYNC0.	-
19	SIN3	Input 3 state. Input 3 state following the synchronization specified by INSYNC0.	-
31:20	SIN…	Input state for the remainder of states implemented in this SCT.	-

15.6.11 SCT match/capture mode register

If UNIFY = 1 in the CONFIG register, only the _L bits of this register are used. In this case, REGMODE_H is not used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers REGMODE_L and REGMODE_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation. The _L bits/registers control the L match/capture registers, and the _H bits/registers control the H match/capture registers.

The SCT contains multiple Match/Capture registers. The Register Mode register selects whether each register acts as a Match register (see Section 15.6.20) or as a Capture register (see Section 15.6.21). Each Match/Capture register has an accompanying register which functions as a Reload register when the primary register is used as a Match register (Section 15.6.22) or as a Capture-Control register when the register is used as a capture register (Section 15.6.23). REGMODE_H is used only when the UNIFY bit is 0.

Table 276. SCT match/capture mode register (REGMODE, offset 0x04C) bit description

Bit	Symbol	Description	Reset value
15:0	REGMOD_L	Each bit controls one match/capture register (register 0 = bit 0, register 1 = bit 1, ...). The number of bits = number of match/captures supported by this SCT. 0 = register operates as match register. 1 = register operates as capture register.	0x0
31:16	REGMOD_H	Each bit controls one match/capture register (register 0 = bit 16, register 1 = bit 17, ...). The number of bits = number of match/captures supported by this SCT. 0 = register operates as match registers. 1 = register operates as capture registers.	0x0

15.6.12 SCT output register

Each SCT output has a corresponding bit in this register to allow software to control the output state directly or read its current state.

While the counter is running, outputs are set, cleared, or toggled only by events. However, using this register, software can write to any of the output registers when both counters are halted to control the outputs directly. Writing to the OUT register is only allowed when all counters (L-counter, H-counter, or unified counter) are halted (HALT bits are set to 1 in the CTRL register).

Software can read this register at any time to sense the state of the outputs.

Table 277. SCT output register (OUTPUT, offset 0x050) bit description

Bit	Symbol	Description	Reset value
15:0	OUT	Writing a 1 to bit n forces the corresponding output HIGH. Writing a 0 forces the corresponding output LOW (output 0 = bit 0, output 1 = bit 1, ...). The number of bits = number of outputs supported by this SCT.	0x0
31:16	-	Reserved	-

15.6.13 SCT bi-directional output control register

For bi-directional mode, this register specifies (for each output) the impact of the counting direction on the meaning of set and clear operations on the output (see Section 15.6.26 and Section 15.6.27). The purpose of this register is to facilitate the creation of center-aligned output waveforms without the need to define additional events.

Table 278. SCT bidirectional output control register (OUTPUTDIRCTRL, offset 0x054) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SETCLR0		Set/clear operation on output 0.	0x0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
		0x3	Value 0x3 is reserved. Do not program this value.	

Table 278. SCT bidirectional output control register (OUTPUTDIRCTRL, offset 0x054) bit description

Bit	Symbol	Value	Description	Reset value
3:2	SETCLR1		Set/clear operation on output 1. Value 0x3 is reserved. Do not program this value.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
5:4	SETCLR2		Set/clear operation on output 2. Value 0x3 is reserved. Do not program this value.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
7:6	SETCLR3		Set/clear operation on output 3. Value 0x3 is reserved. Do not program this value.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
9:8	SETCLR4		Set/clear operation on output 4. Value 0x3 is reserved. Do not program this value.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
31:10	SETCLR...		Set/clear operation controls for the remainder of outputs on this SCT.	0

[1] For as many outputs as are supported by the specific SCTimer/PWM.

15.6.14 SCT conflict resolution register

The output conflict resolution register specifies what action should be taken if multiple events (or even the same event) dictate that a given output should be both set and cleared at the same time.

To enable an event to toggle an output each time the event occurs, set the bits for that event in both the OUTn_SET and OUTn_CLR registers and set the On_RES value to 0x3 in this register.

Table 279. SCT conflict resolution register (RES, offset 0x058) bit description

Bit	Symbol	Value	Description	Reset value
1:0	O0RES		Effect of simultaneous set and clear on output 0.	0x0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR0 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output (or set based on the SETCLR0 field).	
		0x3	Toggle output.	
3:2	O1RES		Effect of simultaneous set and clear on output 1.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR1 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output (or set based on the SETCLR1 field).	
		0x3	Toggle output.	

Table 279. SCT conflict resolution register (RES, offset 0x058) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
5:4	O2RES		Effect of simultaneous set and clear on output 2.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR2 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output n (or set based on the SETCLR2 field).	
		0x3	Toggle output.	
7:6	O3RES		Effect of simultaneous set and clear on output 3.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR3 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output (or set based on the SETCLR3 field).	
		0x3	Toggle output.	
9:8	O4RES		Effect of simultaneous set and clear on output 4.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR4 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output (or set based on the SETCLR4 field).	
		0x3	Toggle output.	
31:10	O…RES		Resolution controls for the remainder of outputs on this SCT.	0

[1] For as many outputs as are supported by the specific SCTimer/PWM.

15.6.15 SCT DMA request 0 and 1 registers

The SCT includes two DMA request outputs. These registers enable the DMA requests to be triggered when a particular event occurs or when counter Match registers are loaded from its Reload registers. The DMA request registers are word-write only. Attempting to write a half-word value to these registers result in a bus error.

Event-triggered DMA requests are particularly useful for launching DMA activity to or from other peripherals under the control of the SCT.

Table 280. SCT DMA 0 request register (DMAREQ0, offset 0x05C) bit description

Bit	Symbol	Description	Reset value
15:0	DEV_0	If bit n is one, event n triggers DMA request 0 (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events supported by this SCT.	0x0
29:16	-	Reserved	-
30	DRL0	A 1 in this bit triggers DMA request 0 when it loads the MATCH_L/Unified registers from the RELOAD_L/Unified registers.	0x0
31	DRQ0	This read-only bit indicates the state of DMA Request 0. Note that if the related DMA channel is enabled and properly set up, it is unlikely that software will see this flag, it will be cleared rapidly by the DMA service. The flag remaining set could point to an issue with DMA setup.	0x0

Table 281. SCT DMA 1 request register (DMAREQ1, offset 0x060) bit description

Bit	Symbol	Description	Reset value
15:0	DEV_1	If bit n is one, event n triggers DMA request 1 (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events supported by this SCT.	0x0
29:16	-	Reserved	-
30	DRL1	A 1 in this bit triggers DMA request 1 when it loads the Match L/Unified registers from the Reload L/Unified registers.	0x0
31	DRQ1	This read-only bit indicates the state of DMA Request 1. Note that if the related DMA channel is enabled and properly set up, it is unlikely that software will see this flag, it will be cleared rapidly by the DMA service. The flag remaining set could point to an issue with DMA setup.	0x0

15.6.16 SCT event interrupt enable register

This register enables flags to request an interrupt if the FLAGn bit in the SCT event flag register (Section 15.6.17) is also set.

Table 282. SCT event interrupt enable register (EVEN, offset 0x0F0) bit description

Bit	Symbol	Description	Reset value
15:0	IEN	The SCT requests an interrupt when bit n of this register and the event flag register are both one (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events supported by this SCT.	0x0
31:16	-	Reserved	-

15.6.17 SCT event flag register

This register records events. Writing ones to this register clears the corresponding flags and negates the SCT interrupt request if all of the enabled flag register bits are zero.

Table 283. SCT event flag register (EVFLAG, offset 0x0F4) bit description

Bit	Symbol	Description	Reset value
15:0	FLAG	Bit n is one if event n has occurred since reset or a 1 was last written to this bit (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of events supported by this SCT.	0x0
31:16	-	Reserved	-

15.6.18 SCT conflict interrupt enable register

This register enables the no-change conflict events specified in the SCT conflict resolution register to generate an interrupt request.

Table 284. SCT conflict interrupt enable register (CONEN, offset 0x0F8) bit description

Bit	Symbol	Description	Reset value
15:0	NCEN	The SCT requests an interrupt when bit n of this register and the SCT conflict flag register are both one (output 0 = bit 0, output 1 = bit 1, …). The number of bits = number of outputs supported by this SCT.	0x0
31:16	-	Reserved	

15.6.19 SCT conflict flag register

This register records a no-change conflict occurrence and also provides details of any bus errors. Writing ones to the NCFLAG bits clears the corresponding read bits and negates the SCT interrupt request if all enabled Flag bits are zero.

Table 285. SCT conflict flag register (CONFLAG, offset 0x0FC) bit description

Bit	Symbol	Description	Reset value
15:0	NCFLAG	Bit n is one if a no-change conflict event occurred on output n since reset or a 1 was last written to this bit (output 0 = bit 0, output 1 = bit 1, ...). The number of bits = number of outputs supported by this SCT.	0x0
29:16	-	Reserved.	-
30	BUSERRL	The most recent bus error from this SCT involved writing CTR L/Unified, STATE L/Unified, MATCH L/Unified, or the Output register when the L/U counter was not halted. A word write to certain L and H registers can be half successful and half unsuccessful.	0x0
31	BUSERRH	The most recent bus error from this SCT involved writing CTR H, STATE H, MATCH H, or the Output register when the H counter was not halted.	0x0

15.6.20 SCT match registers 0 to 9 (REGMODEn bit = 0)

Match registers are compared to the counters to help create events. When the UNIFY bit is 0, the L and H registers are independently compared to the L and H counters. When UNIFY is 1, the combined L and H registers hold a 32-bit value that is compared to the unified counter. A Match can only occur in a clock in which the counter is running (STOP and HALT are both 0).

Match registers can be read at any time. Writing to the MATCH_L, MATCH_H, or unified register is only allowed when the corresponding counter is halted (HALT bits are set to 1 in the CTRL register). Match events occur in the SCT clock in which the counter is (or would be) incremented to the next value. When a Match event limits its counter as described in Section 15.6.4, the value in the Match register is the last value of the counter before it is cleared to zero (or decremented if BIDIR is 1).

There is no “write-through” from Reload registers to Match registers. Before starting a counter, software can write one value to the Match register used in the first cycle of the counter and a different value to the corresponding Match Reload register used in the second cycle.

Table 286. SCT match registers 0 to 9 (MATCH[0:9], offset 0x100 (MATCH0) to 0x124 (MATCH9)) bit description (REGMODEn bit = 0)

Bit	Symbol	Description	Reset value
15:0	MATCHn_L	When UNIFY = 0, read or write the 16-bit value to be compared to the L counter. When UNIFY = 1, read or write the lower 16 bits of the 32-bit value to be compared to the unified counter.	0x0
31:16	MATCHn_H	When UNIFY = 0, read or write the 16-bit value to be compared to the H counter. When UNIFY = 1, read or write the upper 16 bits of the 32-bit value to be compared to the unified counter.	0x0

15.6.21 SCT capture registers 0 to 9 (REGMODEn bit = 1)

These registers allow software to record the counter values upon occurrence of the events selected by the corresponding Capture Control registers occurred.

Table 287. SCT capture registers 0 to 9 (CAP[0:9], offset 0x100 (CAP0) to 0x124 (CAP9)) bit description (REGMODEn bit = 1)

Bit	Symbol	Description	Reset value
15:0	CAPn_L	When UNIFY = 0, read the 16-bit counter value at which this register was last captured. When UNIFY = 1, read the lower 16 bits of the 32-bit value at which this register was last captured.	0x0
31:16	CAPn_H	When UNIFY = 0, read the 16-bit counter value at which this register was last captured. When UNIFY = 1, read the upper 16 bits of the 32-bit value at which this register was last captured.	0x0

15.6.22 SCT match reload registers 0 to 9 (REGMODEn bit = 0)

A Match register (L, H, or unified 32-bit) is loaded from its corresponding Reload register at the start of each new counter cycle, that is

- when BIDIR = 0 and the counter is cleared to zero upon reaching it limit condition.
- when BIDIR = 1 and the counter counts down to 0, unless the appropriate NORELOAD bit is set in the CFG register.

Table 288. SCT match reload registers 0 to 9 (MATCHREL[0:9], offset 0x200 (MATCHREL0) to 0x224 (MATCHREL9)) bit description (REGMODEn bit = 0)

Bit	Symbol	Description	Reset value
15:0	RELOADn_L	When UNIFY = 0, specifies the 16-bit value to be loaded into the MATCHn_L register. When UNIFY = 1, specifies the lower 16 bits of the 32-bit value to be loaded into the MATCHn register.	0x0
31:16	RELOADn_H	When UNIFY = 0, specifies the 16-bit to be loaded into the MATCHn_H register. When UNIFY = 1, specifies the upper 16 bits of the 32-bit value to be loaded into the MATCHn register.	0x0

15.6.23 SCT capture control registers 0 to 9 (REGMODEn bit = 1)

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers CAPCTRLn_L and CAPCTRLn_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Based on a selected event, the capture registers can be loaded with the current counter value when the event occurs.

Each Capture Control register (L, H, or unified 32-bit) controls which events cause the load of corresponding Capture register from the counter.

Table 289. SCT capture control registers 0 to 9 (CAPCTRL[0:9], offset 0x200 (CAPCTRL0) to 0x224 (CAPCTRL9)) bit description (REGMODEn bit = 1)

Bit	Symbol	Description	Reset value
15:0	CAPCONn_L	If bit m is one, event m causes the CAPn_L (UNIFY = 0) or the CAPn (UNIFY = 1) register to be loaded (event 0 = bit 0, event 1 = bit 1, …). The number of bits = number of match/captures supported by this SCT.	0x0
31:16	CAPCONn_H	If bit m is one, event m causes the CAPn_H (UNIFY = 0) register to be loaded (event 0 = bit 16, event 1 = bit 17, …). The number of bits = number of match/captures supported by this SCT.	0x0

15.6.24 SCT event enable registers 0 to 9

Each event can be enabled in some contexts (or states) and disabled in others. Each event defined in the EV_CTRL register has one associated event enable register that can enable or disable the event for each available state.

Each event has one associated SCT event state mask register that allow this event to happen in one or more states of the counter selected by the HEVENT bit in the corresponding EVn_CTRL register.

An event n is disabled when its EVn_STATE register contains all zeros, since it is masked regardless of the current state.

In simple applications that do not use states, write 0x01 to this register to enable each event in exactly one state. Since the state doesn't change (that is, the state variable always remains at its reset value of 0), writing 0x01 permanently enables this event.

Table 290. SCT event state mask registers 0 to 9 (EV[0:9]_STATE, offsets 0x300 (EV0_STATE) to 0x348 (EV9_STATE)) bit description

Bit	Symbol	Description	Reset value
15:0	STATEMSKn	If bit m is one, event n happens in state m of the counter selected by the HEVENT bit (n = event number, m = state number; state 0 = bit 0, state 1= bit 1, …). The number of bits = number of states supported by this SCT.	0
31:16	-	Reserved.	-

15.6.25 SCT event control registers 0 to 9

This register defines the conditions for an event to occur based on the counter values or input and output states.Once the event is configured, it can be selected to trigger multiple actions (for example stop the counter and toggle an output) unless the event is blocked in the current state of the SCT or the counter is halted. To block a particular event from occurring, use the EV_STATE register. To block all events for a given counter, set the HALT bit in the CTRL register or select an event to halt the counter.

An event can be programmed to occur based on a selected input or output edge or level and/or based on its counter value matching a selected match register. In bi-directional mode, events can also be enabled based on the direction of count.

When the UNIFY bit is 0, each event is associated with a particular counter by the HEVENT bit in its event control register. An event is permanently disabled when its event state mask register contains all 0s.

Each event can modify its counter STATE value. If more than one event associated with the same counter occurs in a given clock cycle, only the state change specified for the highest-numbered event among them takes place. Other actions dictated by any simultaneously occurring events all take place.

Table 291. SCT event control register 0 to 9 (EV[0:9]_CTRL, offset 0x304 (EV0_CTRL) to 0x34C (EV9_CTRL)) bit description

Bit	Symbol	Value	Description	Reset value
3:0	MATCHSEL	-	Selects the Match register associated with this event (if any). A match can occur only when the counter selected by the HEVENT bit is running.	0x0
4	HEVENT		Select L/H counter. Do not set this bit if UNIFY = 1.	0x0
		0	Selects the L state and the L match register selected by MATCHSEL.	
		1	Selects the H state and the H match register selected by MATCHSEL.	
5	OUTSEL		Input/output select	0x0
		0	Selects the inputs selected by IOSEL.	
		1	Selects the outputs selected by IOSEL.	
9:6	IOSEL	-	Selects the input or output signal associated with this event (if any). If CKMODE is 1x, the input that is used as the clock may not be selected to trigger events.	0x0
11:10	IOCOND		Selects the I/O condition for event n. (The detection of edges on outputs lag the conditions that switch the outputs by one SCT clock). In order to guarantee proper edge/state detection, an input must have a minimum pulse width of at least one SCT clock period .	0x0
		0x0	LOW	
		0x1	Rise	
		0x2	Fall	
		0x3	HIGH	
13:12	COMBMODE		Selects how the specified match and I/O condition are used and combined.	0x0
		0x0	OR. The event occurs when either the specified match or I/O condition occurs.	
		0x1	MATCH. Uses the specified match only.	
		0x2	IO. Uses the specified I/O condition only.	
		0x3	AND. The event occurs when the specified match and I/O condition occur simultaneously.	
14	STATELD		This bit controls how the STATEV value modifies the state selected by HEVENT when this event is the highest-numbered event occurring for that state.	0x0
		0	STATEV value is added into STATE (the carry-out is ignored).	
		1	STATEV value is loaded into STATE.	
19:15	STATEV		This value is loaded into or added to the state selected by HEVENT, depending on STATELD, when this event is the highest-numbered event occurring for that state. If STATELD and STATEV are both zero, there is no change to the STATE value.	0x0
20	MATCHMEM		If this bit is one and the COMBMODE field specifies a match component to the triggering of this event, then a match is considered to be active whenever the counter value is GREATER THAN OR EQUAL TO the value specified in the match register when counting up, LESS THEN OR EQUAL TO the match value when counting down. If this bit is zero, a match is only be active during the cycle when the counter is equal to the match value.	0x0

Table 291. SCT event control register 0 to 9 (EV[0:9]_CTRL, offset 0x304 (EV0_CTRL) to 0x34C (EV9_CTRL)) bit description

Bit	Symbol	Value	Description	Reset value
22:21	DIRECTION		Direction qualifier for event generation. This field only applies when the counters are operating in BIDIR mode. If BIDIR = 0, the SCT ignores this field. Value 0x3 is reserved.	0x0
		0x0	Direction independent. This event is triggered regardless of the count direction.	
		0x1	Counting up. This event is triggered only during up-counting when BIDIR = 1.	
		0x2	Counting down. This event is triggered only during down-counting when BIDIR = 1.	
31:23	-	-	Reserved	-

15.6.26 SCT output set registers 0 to 7

Based on a selected event, each SCT output can be set.

There is one output set register for each SCT output which selects which events can set that output. Each bit of an output set register is associated with a different event (bit 0 with event 0, etc.). A selected event can set or clear the output depending on the setting of the SETCLRn field in the OUTPUTDIRCTRL register. To define the actual event that sets the output (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Remark: If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Table 292. SCT output set register (OUT[0:7]_SET, offset 0x500 (OUT0_SET) to 0x538 (OUT7_SET) bit description

Bit	Symbol	Description	Reset value
15:0	SET	A 1 in bit m selects event m to set output n (or clear it if SETCLRn = 0x1 or 0x2) output 0 = bit 0, output 1 = bit 1, … The number of bits = number of events supported by this SCT. When the counter is used in bi-directional mode, it is possible to reverse the action specified by the output set and clear registers when counting down, See the OUTPUTCTRL register.	0x0
31:16	-	Reserved	-

15.6.27 SCT output clear registers 0 to 7

Based on a selected event, each SCT output can be cleared.

There is one register for each SCT output which selects which events can clear that output. Each bit of an output clear register is associated with a different event (bit 0 with event 0, etc.). A selected event can clear or set the output depending on the setting of the SETCLRn field in the OUTPUTDIRCTRL register. To define the actual event that clears the output (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Remark: If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Table 293. SCT output clear register (OUT[0:7]_CLR, offset 0x504 (OUT0_CLR) to 0x53C (OUT7_CLR)) bit description

Bit	Symbol	Description	Reset value
15:0	CLR	A 1 in bit m selects event m to clear output n (or set it if SETCLRn = 0x1 or 0x2) event 0 = bit 0, event 1 = bit 1, … The number of bits = number of events supported by this SCT. When the counter is used in bi-directional mode, it is possible to reverse the action specified by the output set and clear registers when counting down, See the OUTPUTCTRL register.	0x0
31:16	-	Reserved	-

15.7 Functional description

15.7.1 Match logic

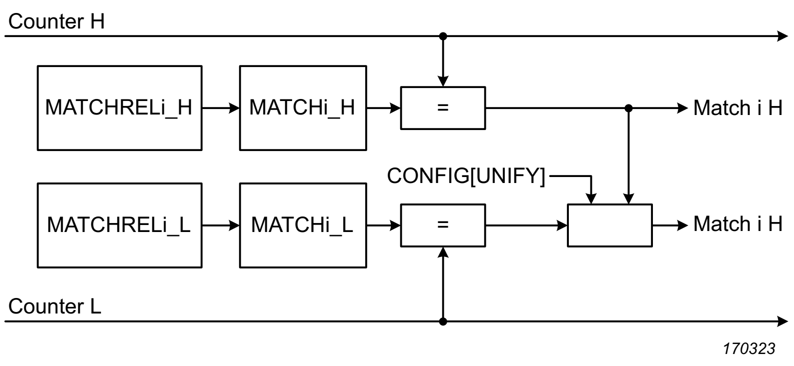


Fig 31. Match logic

15.7.2 Capture logic

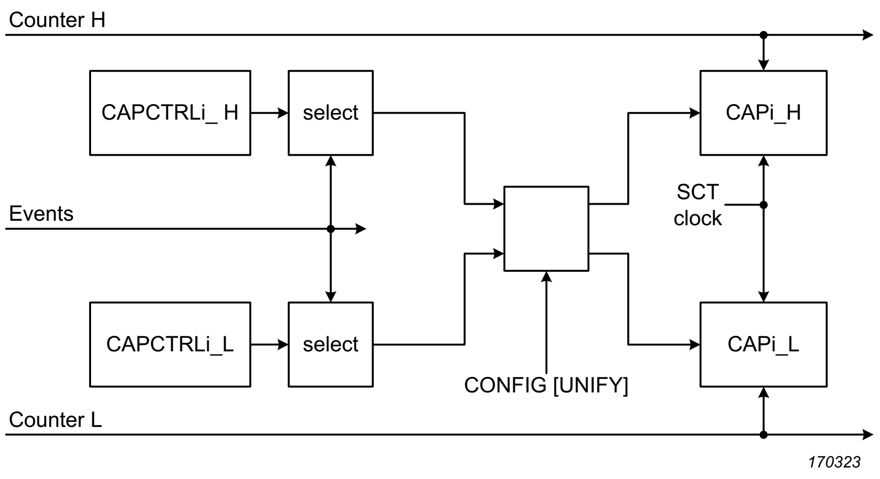


Fig 32. Capture logic

15.7.3 Event selection

State variables allow control of the SCT across more than one cycle of the counter. Counter matches, input/output edges, and state values are combined into a set of general-purpose events that can switch outputs, request interrupts, and change state values.

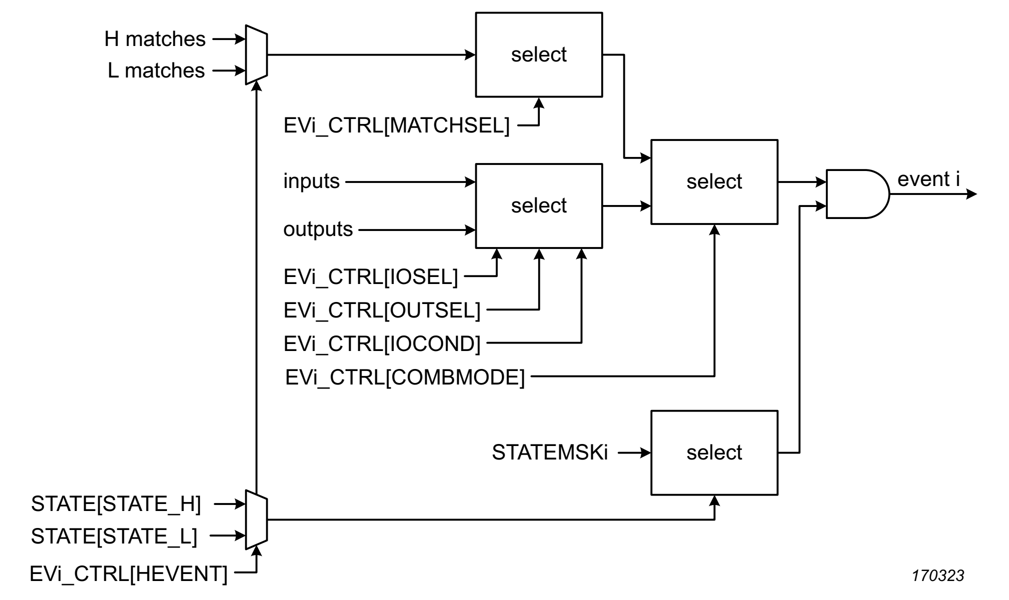


Fig 33. Event selection

15.7.4 Output generation

Figure 34 shows one output slice of the SCT.

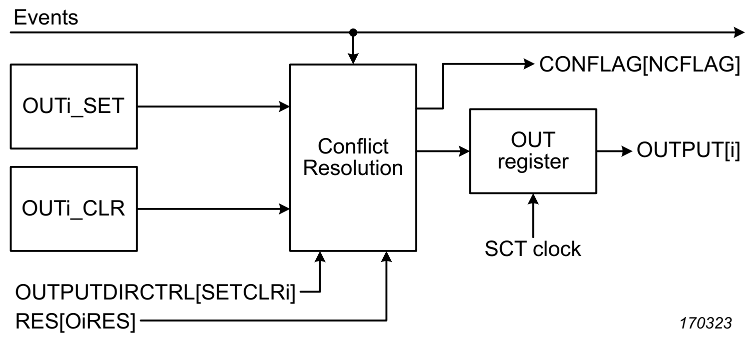


Fig 34. Output slice i

15.7.5 State logic

The SCT can be configured as a timer/counter with multiple programmable states. The states are user-defined through the events that can be captured in each particular state. In a multi-state SCT. the SCT can change from one state to another state when a user-defined event triggers a state change. The state change is triggered through each event’s EV_CTRL register in one of the following ways:

- The event can increment the current state number by a new value.
- The event can write a new state value.

If an event increments the state number beyond the number of available states, the SCT enters a locked state in which all further events are ignored while the counter is still running. Software must interfere to change out of this state.

Software can capture the counter value (and potentially create an interrupt and write to all outputs) when the event moving the SCT into a locked state occurs.Later, while the SCT is in the locked state, software can read the counter again to record the time passed since the locking event and can also read the state variable to obtain the current state number

If the SCT registers an event that forces an abort, putting the SCT in a locked state can be a safe way to record the time that has passed since the abort event while no new events are allowed to occur. Since multiple states (any state number between the maximum implemented state and 31) are locked states, multiple abort or error events can be defined each incrementing the state number by a different value.

15.7.6 Interrupt generation

The SCT generates one interrupt to the NVIC.

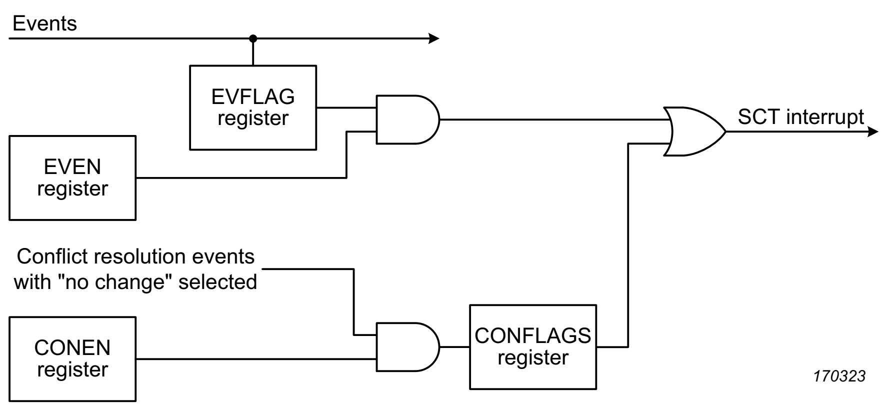


Fig 35. SCT interrupt generation

15.7.7 Clearing the prescaler

When enabled by a non-zero PRE field in the Control register, the prescaler acts as a clock divider for the counter, like a fractional part of the counter value. The prescaler is cleared whenever the counter is cleared or loaded for any of the following reasons:

- Hardware reset
- Software writing to the counter register
- Software writing a 1 to the CLRCTR bit in the control register
- an event selected by a 1 in the counter limit register when BIDIR = 0

When BIDIR is 0, a limit event caused by an I/O signal can clear a non-zero prescaler. However, a limit event caused by a Match only clears a non-zero prescaler in one special case as described Section 15.7.8.

A limit event when BIDIR is 1 does not clear the prescaler. Rather it clears the DOWN bit in the Control register, and decrements the counter on the same clock if the counter is enabled in that clock.

15.7.8 Match vs. I/O events

Counter operation is complicated by the prescaler and by clock mode 01 in which the SCT clock is the bus clock. However, the prescaler and counter are enabled to count only when a selected edge is detected on a clock input.

- The prescaler is enabled when the clock mode is not 01, or when the input edge selected by the CLKSEL field is detected.
- The counter is enabled when the prescaler is enabled, and (PRELIM=0 or the prescaler is equal to the value in PRELIM).

An I/O component of an event can occur in any SCT clock when its counter HALT bit is 0. In general, a Match component of an event can only occur in a UT clock when its counter HALT and STOP bits are both 0 and the counter is enabled.

Table 294 shows when the various kinds of events can occur.

Table 294. Event conditions

COMBMODE	IOMODE	Event can occur on clock:
IO	Any	Event can occur whenever HALT = 0 (type A).
MATCH	Any	Event can occur when HALT = 0 and STOP = 0 and the counter is enabled (type C).
OR	Any	From the IO component: Event can occur whenever HALT = 0 (A). From the match component: Event can occur when HALT = 0 and STOP = 0 and the counter is enabled (C).
AND	LOW or HIGH	Event can occur when HALT = 0 and STOP = 0 and the counter is enabled (C).
AND	RISE or FALL	Event can occur whenever HALT = 0 (A).

15.7.9 SCT operation

In its simplest, single-state configuration, the SCT operates as an event controlled one- or bidirectional counter. Events can be configured to be counter match events, an input or output level, transitions on an input or output pin, or a combination of match and input/output behavior. In response to an event, the SCT output or outputs can transition, or the SCT can perform other actions such as creating an interrupt or starting, stopping, or resetting the counter. Multiple simultaneous actions are allowed for each event. Furthermore, any number of events can trigger one specific action of the SCT.

An action or multiple actions of the SCT uniquely define an event. A state is defined by which events are enabled to trigger an SCT action or actions in any stage of the counter. Events not selected for this state are ignored.

In a multi-state configuration, states change in response to events. A state change is an additional action that the SCT can perform when the event occurs. When an event is configured to change the state, the new state defines a new set of events resulting in different actions of the SCT. Through multiple cycles of the counter, events can change the state multiple times and thus create a large variety of event controlled transitions on the SCT outputs and/or interrupts.

Once configured, the SCT can run continuously without software intervention and can generate multiple output patterns entirely under the control of events.

- To configure the SCT, see Section 15.7.10.
- To start, run, and stop the SCT, see Section 15.7.11.

- To configure the SCT as simple event controlled counter/timer, see Section 15.7.12.

15.7.10 Configure the SCT

To set up the SCT for multiple events and states, perform the following configuration steps:

15.7.10.1 Configure the counter

1. Configure the L and H counters in the CONFIG register by selecting two independent 16-bit counters (L counter and H counter) or one combined 32-bit counter in the UNIFY field.
2. Select the SCT clock source in the CONFIG register (fields CLKMODE and CLKSEL) from any of the inputs or an internal clock.

15.7.10.2 Configure the match and capture registers

1. Select how many match and capture registers the application uses (not more than what is available on this device):
   – In the REGMODE register, select for each of the match/capture register pairs whether the register is used as a match register or capture register.
2. Define match conditions for each match register selected:
   – Each match register MATCH sets one match value, if a 32-bit counter is used, or two match values, if the L and H 16-bit counters are used.
   – Each match reload register MATCHRELOAD sets a reload value that is loaded into the match register when the counter reaches a limit condition or the value 0.

15.7.10.3 Configure events and event responses

1. Define when each event can occur in the following way in the EVn_CTRL registers (up to 6, one register per event):
   – Select whether the event occurs on an input or output changing, on an input or output level, a match condition of the counter, or a combination of match and input/output conditions in field COMBMODE.
   – For a match condition:

     Select the match register that contains the match condition for the event to occur. Enter the number of the selected match register in field MATCHSEL.

     If using L and H counters, define whether the event occurs on matching the L or the H counter in field HEVENT.
   – For an SCT input or output level or transition:

     Select the input number or the output number that is associated with this event in fields IOSEL and OUTSEL.

     Define how the selected input or output triggers the event (edge or level sensitive) in field IOCOND.
2. Define what the effect of each event is on the SCT outputs in the OUTn_SET or OUTn_CLR registers (up to the maximum number of outputs on this device, one register per output):

   – For each SCT output, select which events set or clear this output. More than one event can change the output, and each event can change multiple outputs.

3. Define how each event affects the counter:
   – Set the corresponding event bit in the LIMIT register for the event to set an upper limit for the counter.

     When a limit event occurs in unidirectional mode, the counter is cleared to zero and begins counting up on the next clock edge.

     When a limit event occurs in bidirectional mode, the counter begins to count down from the current value on the next clock edge.
   – Set the corresponding event bit in the HALT register for the event to halt the counter. If the counter is halted, it stops counting and no new events can occur. The counter operation can only be restored by clearing the HALT_L and/or the HALT_H bits in the CTRL register.
   – Set the corresponding event bit in the STOP register for the event to stop the counter. If the counter is stopped, it stops counting. However, an event that is configured as a transition on an input/output can restart the counter.
   – Set the corresponding event bit in the START register for the event to restart the counting. Only events that are defined by an input changing can be used to restart the counter.
4. Define which events contribute to the SCT interrupt:
   – Set the corresponding event bit in the EVEN and the EVFLAG registers to enable the event to contribute to the SCT interrupt.

15.7.10.4 Configure multiple states

1. In the EVn_STATE register for each event (up to the maximum number of events on this device, one register per event), select the state or states (up to 2) in which this event is allowed to occur. Each state can be selected for more than one event.
2. Determine how the event affects the system state:

   In the EVn_CTRL registers (up to the maximum number of events on this device, one register per event), set the new state value in the STATEV field for this event. If the event is the highest numbered in the current state, this value is either added to the existing state value or replaces the existing state value, depending on the field STATELD.

   Remark: If there are higher numbered events in the current state, this event cannot change the state.

   If the STATEV and STATELD values are set to zero, the state does not change.

15.7.10.5 Miscellaneous options

- There are a certain (selectable) number of capture registers. Each capture register can be programmed to capture the counter contents when one or more events occur.
- If the counter is in bidirectional mode, the effect of set and clear of an output can be made to depend on whether the counter is counting up or down by writing to the OUTPUTDIRCTRL register.

15.7.11 Run the SCT

1. Configure the SCT (see Section 15.7.10 “Configure the SCT”).
2. Write to the STATE register to define the initial state. By default the initial state is state 0.
3. To start the SCT, write to the CTRL register:
   - Clear the counters.
   - Clear or set the STOP_L and/or STOP_H bits.

     Remark: The counter starts counting once the STOP bit is cleared as well. If the STOP bit is set, the SCT waits instead for an event to occur that is configured to start the counter.
   - For each counter, select unidirectional or bidirectional counting mode (field BIDIR_L and/or BIDIR_H).
   - Select the prescale factor for the counter clock (CTRL register).
   - Clear the HALT_L and/or HALT_H bit. By default, the counters are halted and no events can occur.
4. To stop the counters by software at any time, stop or halt the counter (write to STOP_L and/or STOP_H bits or HALT_L and/or HALT_H bits in the CTRL register).
   - When the counters are stopped, both an event configured to clear the STOP bit or software writing a zero to the STOP bit can start the counter again.
   - When the counter are halted, only a software write to clear the HALT bit can start the counter again. No events can occur.
   - When the counters are halted, software can set any SCT output HIGH or LOW directly by writing to the OUT register.

The current state can be read at any time by reading the STATE register.

To change the current state by software (that is independently of any event occurring), set the HALT bit and write to the STATE register to change the state value. Writing to the STATE register is only allowed when the counter is halted (the HALT_L and/or HALT_H bits are set) and no events can occur.

15.7.12 Configure the SCT without using states

The SCT can be used as standard counter/timer with external capture inputs and match outputs without using the state logic. To operate the SCT without states, configure the SCT as follows:

- Write zero to the STATE register (zero is the default).
- Write zero to the STATELD and STATEV fields in the EVCTRL registers for each event.
- Write 0x1 to the EVn_STATE register of each event. Writing 0x1 enables the event.

  In effect, the event is allowed to occur in a single state which never changes while the counter is running.

15.7.13 SCT PWM Example

Figure 36 shows a simple application of the SCT using two sets of match events (EV0/1 and EV3/4) to set/clear SCT output 0. The timer is automatically reset whenever it reaches the MAT0 match value.

In the initial state 0, match event EV0 sets output 0 to HIGH and match event EV1 clears output 0. The SCT input 0 is monitored: If input0 is found LOW by the next time the timer is reset(EV2), the state is changed to state 1, and EV3/4 are enabled, which create the same output but triggered by different match values. If input 0 is found HIGH by the next time the timer is reset, the associated event (EV5) causes the state to change back to state 0where the events EV0 and EV1 are enabled.

The example uses the following SCT configuration:

- 1 input
- 1 output
- 5 match registers
- 6 events and match 0 used with autolimit function
- 2 states

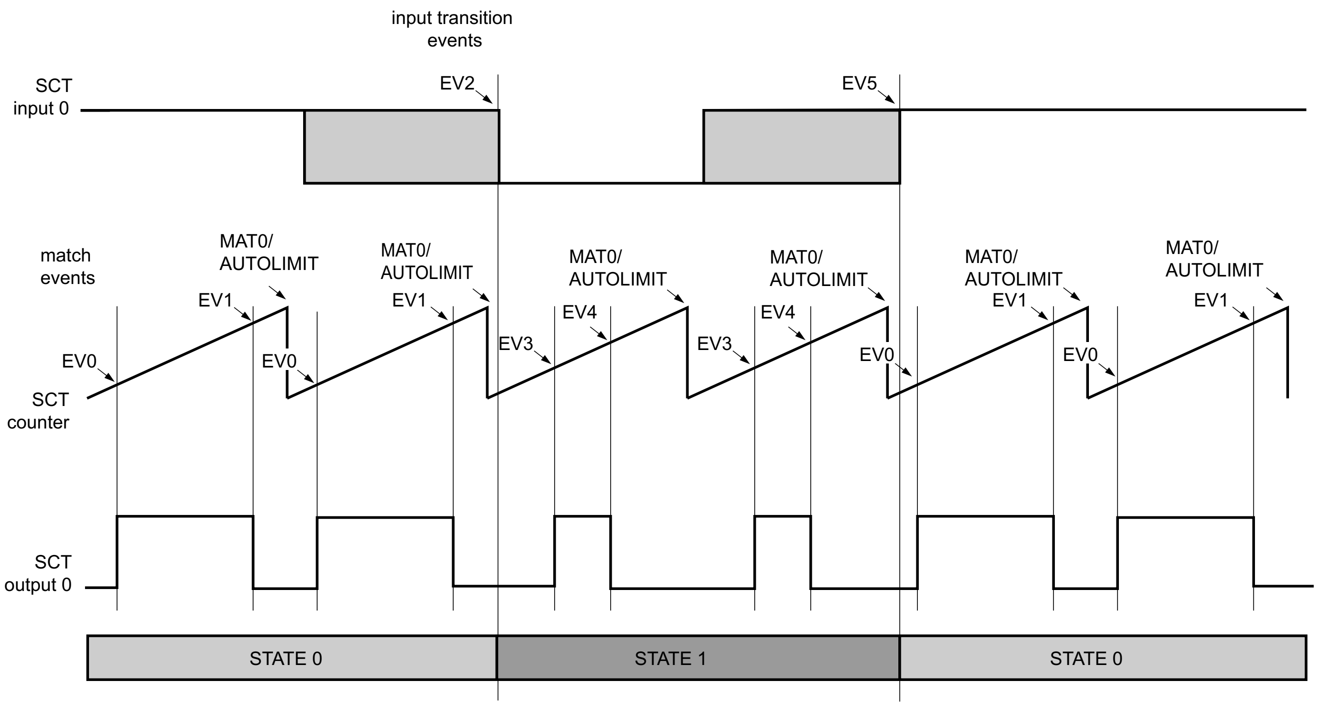


Fig 36. SCT configuration example

This application of the SCT uses the following configuration (all register values not listed in Table 295 are set to their default values):

Table 295. SCT configuration example

Configuration	Registers	Setting
Counter	CONFIG	Uses one counter (UNIFY = 1).
	CONFIG	Enable the autolimit for MAT0. (AUTOLIMIT = 1.)
	CTRL	Uses unidirectional counter (BIDIR_L = 0).
Clock base	CONFIG	Uses default values for clock configuration.
Match/Capture registers	REGMODE	Configure one match register for each match event by setting REGMODE_L bits 0,1, 2, 3, 4 to 0. This is the default.
Define match values	MATCH 0/1/2/3/4	Set a match value MATCH0/1/2/4/5_L in each register. The match 0 register serves as an automatic limit event that resets the counter. without using an event. To enable the automatic limit, set the AUTOLIMIT bit in the CONFIG register.
Define match reload values	MATCHREL 0/1/2/3/4	Set a match reload value RELOAD0/1/2/3/4_L in each register (same as the match value in this example).
Define when event 0 occurs	EV0_CTRL	• Set COMBMODE = 0x1. Event 0 uses match condition only. • Set MATCHSEL = 1. Select match value of match register 1. The match value of MAT1 is associated with event 0.
Define when event 1 occurs	EV1_CTRL	• Set COMBMODE = 0x1. Event 1 uses match condition only. • Set MATCHSEL = 2 Select match value of match register 2. The match value of MAT2 is associated with event 1.
Define when event 2 occurs	EV2_CTRL	• Set COMBMODE = 0x3. Event 2 uses match condition and I/O condition. • Set IOSEL = 0. Select input 0. • Set IOCOND = 0x0. Input 0 is LOW. • Set MATCHSEL = 0. Chooses match register 0 to qualify the event.
Define how event 2 changes the state	EV2_CTRL	Set STATEV bits to 1 and the STATED bit to 1. Event 2 changes the state to state 1.
Define when event 3 occurs	EV3_CTRL	• Set COMBMODE = 0x1. Event 3 uses match condition only. • Set MATCHSEL = 0x3. Select match value of match register 3. The match value of MAT3 is associated with event 3.
Define when event 4 occurs	EV4_CTRL	• Set COMBMODE = 0x1. Event 4 uses match condition only. • Set MATCHSEL = 0x4. Select match value of match register 4.The match value of MAT4 is associated with event 4.
Define when event 5 occurs	EV5_CTRL	• Set COMBMODE = 0x3. Event 5 uses match condition and I/O condition. • Set IOSEL = 0. Select input 0. • Set IOCOND = 0x3. Input 0 is HIGH. • Set MATCHSEL = 0. Chooses match register 0 to qualify the event.
Define how event 5 changes the state	EV5_CTRL	Set STATEV bits to 0 and the STATED bit to 1. Event 5 changes the state to state 0.
Define by which events output 0 is set	OUT0_SET	Set SET0 bits 0 (for event 0) and 3 (for event 3) to one to set the output when these events 0 and 3 occur.
Define by which events output 0 is cleared	OUT0_CLR	Set CLR0 bits 1 (for events 1) and 4 (for event 4) to one to clear the output when events 1 and 4 occur.
Configure states in which event 0 is enabled	EV0_STATE	Set STATEMSK0 bit 0 to 1. Set all other bits to 0. Event 0 is enabled in state 0.
Configure states in which event 1 is enabled	EV1_STATE	Set STATEMSK1 bit 0 to 1. Set all other bits to 0. Event 1 is enabled in state 0.
Configure states in which event 2 is enabled	EV2_STATE	Set STATEMSK2 bit 0 to 1. Set all other bits to 0. Event 2 is enabled in state 0.

Table 295. SCT configuration example

Configuration	Registers	Setting
Configure states in which event 3 is enabled	EV3_STATE	Set STATEMSK3 bit 1 to 1. Set all other bits to 0. Event 3 is enabled in state 1.
Configure states in which event 4 is enabled	EV4_STATE	Set STATEMSK4 bit 1 to 1. Set all other bits to 0. Event 4 is enabled in state 1.
Configure states in which event 5 is enabled	EV5_STATE	Set STATEMSK5 bit 1 to 1. Set all other bits to 0. Event 5 is enabled in state 1.

UM11071

Chapter 16: LPC51U68 Standard counter/timers (CTIMER0-1, 3)

Rev. 1.1 — 17 May 2018 **User manual**

16.1 How to read this chapter

These three standard timers are available on all LPC51U68 devices.

16.2 Features

- Each is a 32-bit counter/timer with a programmable 32-bit prescaler. Four of the timers include external capture and match pin connections.
- Counter or timer operation.
- Up to four 32-bit captures that can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt (the number of capture inputs for each timer that are actually available on device pins may vary by part number).
- The timer and prescaler may be configured to be cleared on a designated capture event. This feature permits easy pulse-width measurement by clearing the timer on the leading edge of an input pulse and capturing the timer value on the trailing edge.
- Four 32-bit match registers that allow:
  - Continuous operation with optional interrupt generation on match.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.
- Up to 4 external outputs corresponding to match registers with the following capabilities (the number of match outputs for each timer that are actually available on device pins may vary by part number):
  - Set LOW on match.
  - Set HIGH on match.
  - Toggle on match.
  - Do nothing on match.
- Up to 4 match registers can be configured for PWM operation, allowing up to 3 single edged controlled PWM outputs. (The number of match outputs for each timer that are actually available on device pins may vary by part number.)
- Up to 2 match registers can be used to generate DMA requests.

16.3 Basic configuration

- Set the appropriate bits to enable clocks to timers that will be used: CTIMER0 and CTIMER1 in the AHBCLKCTRL1 register (Section 6.5.17), and CTIMER3 in the ASYNCAPBCLKCTRL register (Section 6.5.61).

- Clear the timer reset using the ASYNCPRESETCTRL register (Table 109 for CTIMER0 and CTIMER1) and the PRESETCTRL1 register (Table 163 for 3 and 4). Note that bit positions in the reset control registers match the bit positions in the clock control registers.
- Pins: Select timer pins and pin modes as needed through the relevant IOCON registers (Chapter 9 “LPC51U68 I/O pin configuration (IOCON)”).
- Interrupts: See register MCR (Table 304) and CCR (Table 306) for match and capture events. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register. For interrupt connections, see Table 79.
- DMA: Some timer match conditions can be used to generate timed DMA requests, see Table 236.

16.4 Applications

- Interval Timer for counting internal events.
- PWM outputs
- Pulse Width Demodulator via Capture inputs.
- Free running timer.

16.5 General description

The counter/timer block is designed to count cycles of the APB bus clock or an externally supplied clock and can optionally generate interrupts or perform other actions at specified timer values based on four match registers. Each counter/timer also includes capture inputs to trap the timer value when an input signal transitions, optionally generating an interrupt.

In PWM mode, three match registers can be used to provide a single-edge controlled PWM output on the match output pins. One match register is used to control the PWM cycle length.

16.5.1 Capture inputs

The capture signal can be configured to load the Capture Register with the value in the counter/timer and optionally generate an interrupt. The capture signal is generated by one of the pins with a capture function. Each capture signal is connected to one capture channel of the timer.

The Counter/Timer block can select a capture signal as a clock source instead of the APB bus clock. For more details see Section 16.7.11.

16.5.2 Match outputs

When a match register equals the timer counter (TC), the corresponding match output can either toggle, go LOW, go HIGH, or do nothing. The External Match Register (EMR) and the PWM Control Register (PWMCON) control the functionality of this output.

16.5.3 Architecture

The block diagram for the timers is shown in Figure 37.

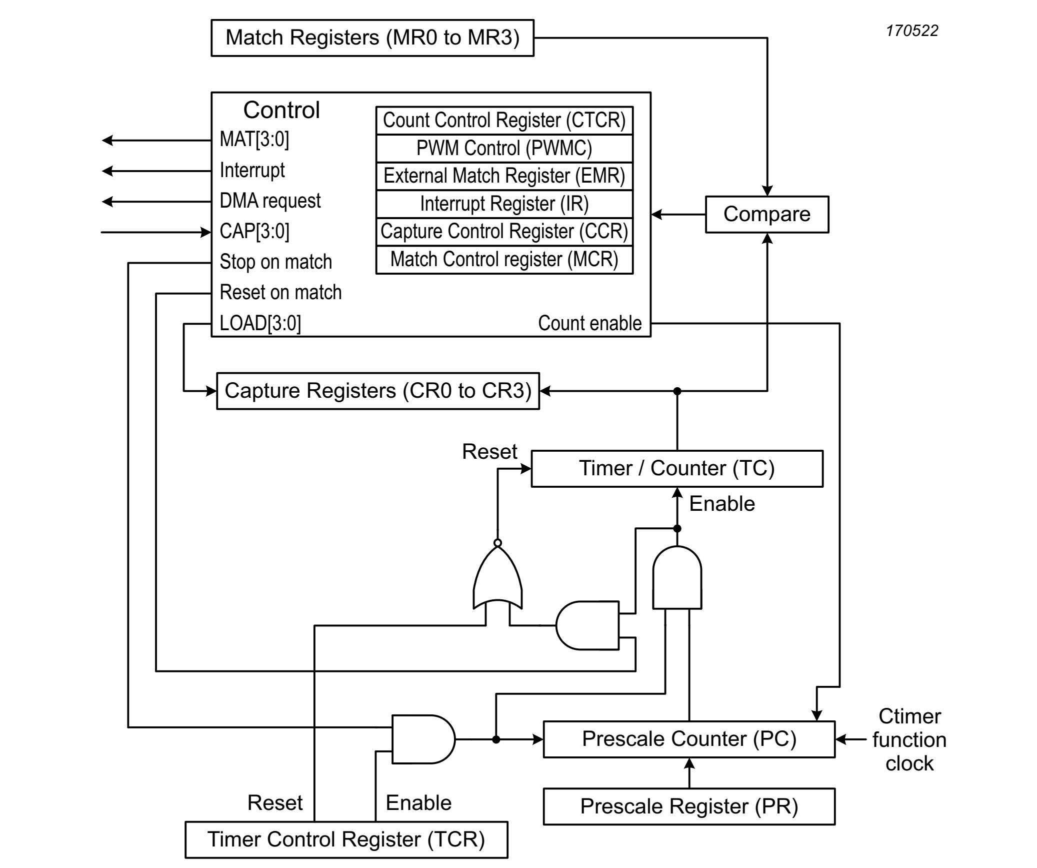


Fig 37. 32-bit counter/timer block diagram

16.6 Pin description

Table 296 gives a summary of each of the Timer/Counter related pins. Recommended IOCON settings are shown in Table 188 and Table 189. Also note that different part number and package variations may provide different CTIMER related pin functions.

Table 296. Timer/Counter pin description

Pin	Type	Description
CTIMER0_CAP3:0 CTIMER1_CAP3:0 CTIMER3_CAP3:0	Input	Capture Signals- A transition on a capture pin can be configured to load one of the Capture Registers with the value in the Timer Counter and optionally generate an interrupt. Capture functionality can be selected from a number of pins. Timer/Counter block can select a capture signal as a clock source instead of the APB bus clock. For more details see Section 16.7.11.
CTIMER0_MAT1:0 CTIMER1_MAT1:0 CTIMER3_MAT1:0	Output	External Match Output - When a match register (MR3:0) equals the timer counter (TC) this output can either toggle, go low, go high, or do nothing. The External Match Register (EMR) controls the functionality of this output. Match Output functionality can be selected on a number of pins in parallel.

Table 297: Suggested CTIMER timer pin settings

IOCON bit(s)	Type D pin	Type A pin	Type I pin
10	OD: Set to 0 unless open-drain output is desired.	Same as type D.	I2CFILTER: Set to 1
9	SLEW: Generally set to 0.	Not used, set to 0	I2CDRIVE: Set to 0.
8	FILTEROFF: Generally set to 1.	Same as type D.	Same as type D.
7	DIGIMODE: Set to 1.	Same as type D.	Same as type D.
6	INVERT: Set to 0.	Same as type D.	Same as type D.
5	Not used, set to 0.	Same as type D.	I2CSLEW: Set to 1.
4:3	MODE: Set to 0 (pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input).	Same as type D.	Not used, set to 0.
2:0	FUNC: Must select the correct function for this peripheral.	Same as type D.	Same as type D.
General comment	A good choice for timer capture input or match output.	A reasonable choice for timer capture input or match output.	Not recommended for timer match outputs.

16.6.1 Multiple CAP and MAT pins

Software can select from multiple pins for the CAP or MAT functions in the IOCON registers, which are described in Chapter 9 “LPC51U68 I/O pin configuration (IOCON)”. Note that match conditions may be used internally without the use of a device pin.

16.7 Register description

Each Timer/Counter contains the registers shown in Table 298.

Table 298. Register overview: CTIMER0/1/3 (register base addresses 0x4000 8000 (CTIMER0), 0x4000 9000 (CTIMER1), 0x4004 8000 (CTIMER3))

Name	Access	Offset	Description	Reset value[1]	Section
IR	R/W	0x00	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.	0x0	16.7.1
TCR	R/W	0x04	Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	0x0	16.7.2
TC	R/W	0x08	Timer Counter. The 32-bit TC is incremented every PR+1 cycles of the APB bus clock. The TC is controlled through the TCR.	0x0	16.7.3
PR	R/W	0x0C	Prescale Register. When the Prescale Counter (PC) is equal to this value, the next clock increments the TC and clears the PC.	0x0	16.7.4
PC	R/W	0x10	Prescale Counter. The 32 bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented and the PC is cleared. The PC is observable and controllable through the bus interface.	0x0	16.7.5
MCR	R/W	0x14	Match Control Register. The MCR is used to control whether an interrupt is generated, and whether the TC is reset when a Match occurs.	0x0	16.7.6
MR0	R/W	0x18	Match Register 0. MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC.	0x0	16.7.7
MR1	R/W	0x1C	Match Register 1. See MR0 description.	0x0	16.7.7
MR2	R/W	0x20	Match Register 2. See MR0 description.	0x0	16.7.7
MR3	R/W	0x24	Match Register 3. See MR0 description.	0x0	16.7.7
CCR	R/W	0x28	Capture Control Register. The CCR controls which edges of the capture inputs are used to load the Capture Registers and whether or not an interrupt is generated when a capture takes place.	0x0	16.7.8
CR0	RO	0x2C	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the CAPn.0 input.	0x0	16.7.9
CR1	RO	0x30	Capture Register 1. See CR0 description.	0x0	16.7.9
CR2	RO	0x34	Capture Register 2. See CR0 description.	0x0	16.7.9
CR3	RO	0x38	Capture Register 3. See CR0 description.	0x0	16.7.9
EMR	R/W	0x3C	External Match Register. The EMR controls the match function and the external match pins.	0x0	16.7.10
CTCR	R/W	0x70	Count Control Register. The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	0x0	16.7.11
PWMC	R/W	0x74	PWM Control Register. The PWMCON enables PWM mode for the external match pins.	0x0	16.7.12

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

16.7.1 Interrupt Register

The Interrupt Register consists of 4 bits for the match interrupts and 4 bits for the capture interrupts. If an interrupt is generated then the corresponding bit in the IR will be high. Otherwise, the bit will be low. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect. The act of clearing an interrupt for a timer match also clears any corresponding DMA request. Writing a zero has no effect.

Table 299. Interrupt Register (IR, offset 0x00) bit description

Bit	Symbol	Description	Reset Value
0	MR0INT	Interrupt flag for match channel 0.	0x0
1	MR1INT	Interrupt flag for match channel 1.	0x0
2	MR2INT	Interrupt flag for match channel 2.	0x0
3	MR3INT	Interrupt flag for match channel 3.	0x0
4	CR0INT	Interrupt flag for capture channel 0 event.	0x0
5	CR1INT	Interrupt flag for capture channel 1 event.	0x0
6	CR2INT	Interrupt flag for capture channel 2 event.	0x0
7	CR3INT	Interrupt flag for capture channel 3 event.	0x0
31:8	-	Reserved. Read value is undefined, only zero should be written.	-

16.7.2 Timer Control Register

The Timer Control Register (TCR) is used to control the operation of the Timer/Counter.

Table 300. Timer Control Register (TCR, offset 0x04) bit description

Bit	Symbol	Value	Description	Reset value
0	CEN		Counter enable.	0x0
		0	Disabled.The counters are disabled.	
		1	Enabled. The Timer Counter and Prescale Counter are enabled.	
1	CRST		Counter reset.	0x0
		0	Disabled. Do nothing.	
		1	Enabled. The Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of the APB bus clock. The counters remain reset until TCR[1] is returned to zero.	
31:2	-	-	Reserved. Read value is undefined, only zero should be written.	-

16.7.3 Timer Counter register

The 32-bit Timer Counter register is incremented when the prescale counter reaches its terminal count. Unless it is reset before reaching its upper limit, the Timer Counter will count up through the value 0xFFFF FFFF and then wrap back to the value 0x0000 0000. This event does not cause an interrupt, but a match register can be used to detect an overflow if needed.

Table 301. Timer counter register (TC, offset 0x08) bit description

Bit	Symbol	Description	Reset value
31:0	TCVAL	Timer counter value.	0x0

16.7.4 Prescale register

The 32-bit Prescale register specifies the maximum value for the Prescale Counter.

Table 302. Timer prescale register (PR, offset 0x0C) bit description

Bit	Symbol	Description	Reset value
31:0	PRVAL	Prescale counter value.	0x0

16.7.5 Prescale Counter register

The 32-bit Prescale Counter controls division of the APB bus clock by some constant value before it is applied to the Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The Prescale Counter is incremented on every APB bus clock. When it reaches the value stored in the Prescale register, the Timer Counter is incremented and the Prescale Counter is reset on the next APB bus clock. This causes the Timer Counter to increment on every APB bus clock when PR = 0, every 2 APB bus clocks when PR = 1, etc.

Table 303. Timer prescale counter register (PC, offset 0x10) bit description

Bit	Symbol	Description	Reset value
31:0	PCVAL	Prescale counter value.	0x0

16.7.6 Match Control Register

The Match Control Register is used to control what operations are performed when one of the Match Registers matches the Timer Counter.

Table 304. Match Control Register (MCR, offset 0x14) bit description

Bit	Symbol	Description	Reset Value
0	MR0I	Interrupt on MR0: an interrupt is generated when MR0 matches the value in the TC. 0 = disabled. 1 = enabled.	0x0
1	MR0R	Reset on MR0: the TC will be reset if MR0 matches it. 0 = disabled. 1 = enabled.	0x0
2	MR0S	Stop on MR0: the TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC. 0 = disabled. 1 = enabled.	0x0
3	MR1I	Interrupt on MR1: an interrupt is generated when MR1 matches the value in the TC. 0 = disabled. 1 = enabled. 0 = disabled. 1 = enabled.	0x0
4	MR1R	Reset on MR1: the TC will be reset if MR1 matches it. 0 = disabled. 1 = enabled.	0x0
5	MR1S	Stop on MR1: the TC and PC will be stopped and TCR[0] will be set to 0 if MR1 matches the TC. 0 = disabled. 1 = enabled.	0x0
6	MR2I	Interrupt on MR2: an interrupt is generated when MR2 matches the value in the TC. 0 = disabled. 1 = enabled.	0x0
7	MR2R	Reset on MR2: the TC will be reset if MR2 matches it. 0 = disabled. 1 = enabled.	0x0
8	MR2S	Stop on MR2: the TC and PC will be stopped and TCR[0] will be set to 0 if MR2 matches the TC. 0 = disabled. 1 = enabled.	0x0
9	MR3I	Interrupt on MR3: an interrupt is generated when MR3 matches the value in the TC. 0 = disabled. 1 = enabled.	0x0

Table 304. Match Control Register (MCR, offset 0x14) bit description

Bit	Symbol	Description	Reset Value
10	MR3R	Reset on MR3: the TC will be reset if MR3 matches it. 0 = disabled. 1 = enabled.	0x0
11	MR3S	Stop on MR3: the TC and PC will be stopped and TCR[0] will be set to 0 if MR3 matches the TC. 0 = disabled. 1 = enabled.	0x0
31:12	-	Reserved. Read value is undefined, only zero should be written.	-

16.7.7 Match Registers

The Match register values are continuously compared to the Timer Counter value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the Timer Counter, or stop the timer. Actions are controlled by the settings in the MCR register.

Table 305. Timer match registers (MR[0:3], offset [0x18:0x24]) bit description

Bit	Symbol	Description	Reset value
31:0	MATCH	Timer counter match value.	0x0

16.7.8 Capture Control Register

The Capture Control Register is used to control whether one of the four Capture Registers is loaded with the value in the Timer Counter when the capture event occurs, and whether an interrupt is generated by the capture event. Setting both the rising and falling bits at the same time is a valid configuration, resulting in a capture event for both edges. In the description below, "n" represents the timer number, 0 or 1.

Note: If Counter mode is selected for a particular CAP input in the CTCR, the 3 bits for that input in this register should be programmed as 000, but capture and/or interrupt can be selected for the other CAP inputs. Note that different part number and package variations may provide different capture input pin functions.

Table 306. Capture Control Register (CCR, offset 0x28) bit description

Bit	Symbol	Description	Reset Value
0	CAP0RE	Rising edge of capture channel 0: a sequence of 0 then 1 causes CR0 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0x0
1	CAP0FE	Falling edge of capture channel 0: a sequence of 1 then 0 causes CR0 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0x0
2	CAP0I	Generate interrupt on channel 0 capture event: a CR0 load generates an interrupt.	0x0
3	CAP1RE	Rising edge of capture channel 1: a sequence of 0 then 1 causes CR1 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0x0
4	CAP1FE	Falling edge of capture channel 1: a sequence of 1 then 0 causes CR1 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0x0
5	CAP1I	Generate interrupt on channel 1 capture event: a CR1 load generates an interrupt.	0x0
6	CAP2RE	Rising edge of capture channel 2: a sequence of 0 then 1 causes CR2 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0x0
7	CAP2FE	Falling edge of capture channel 2: a sequence of 1 then 0 causes CR2 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0x0
8	CAP2I	Generate interrupt on channel 2 capture event: a CR2 load generates an interrupt.	0x0

Table 306. Capture Control Register (CCR, offset 0x28) bit description

Bit	Symbol	Description	Reset Value
9	CAP3RE	Rising edge of capture channel 3: a sequence of 0 then 1 causes CR3 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0x0
10	CAP3FE	Falling edge of capture channel 3: a sequence of 1 then 0 causes CR3 to be loaded with the contents of TC. 0 = disabled. 1 = enabled.	0x0
11	CAP3I	Generate interrupt on channel 3 capture event: a CR3 load generates an interrupt.	0x0
31:12	-	Reserved. Read value is undefined, only zero should be written.	-

16.7.9 Capture Registers

Each Capture register is associated with one capture channel and may be loaded with the counter/timer value when a specified event occurs on the signal defined for that capture channel. The signal could originate from an external pin or from an internal source. The settings in the Capture Control Register register determine whether the capture function is enabled, and whether a capture event happens on the rising edge of the associated signal, the falling edge, or on both edges.

Table 307. Capture registers (CR[0:3], offsets [0x2C:0x38]) bit description

Bit	Symbol	Description	Reset value
31:0	CAP	Timer counter capture value.	0x0

16.7.10 External Match Register

The External Match Register provides both control and status of the external match pins. In the descriptions below, “n” represents the timer number, 0 or 1, and “m” represent a Match number, 0 through 3. Note that different part number and package variations may provide different match output pin functions.

Match events for Match 0 and Match 1 in each timer can cause a DMA request, see Section 16.8.2.

If the match outputs are configured as PWM output, the function of the external match registers is determined by the PWM rules (Section 16.8.1 “Rules for single edge controlled PWM outputs” on page 259).

Table 308. External match register (EMR, offset 0x3C) bit description

Bit	Symbol	Value	Description	Reset value
0	EM0	-	External Match 0. This bit reflects the state of output MAT0, whether or not this output is connected to a pin. When a match occurs between the TC and MR0, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by EMR[5:4]. This bit is driven to the MAT pins if the match function is selected via IOCON. 0 = LOW. 1 = HIGH.	0x0
1	EM1	-	External Match 1. This bit reflects the state of output MAT1, whether or not this output is connected to a pin. When a match occurs between the TC and MR1, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by EMR[7:6]. This bit is driven to the MAT pins if the match function is selected via IOCON. 0 = LOW. 1 = HIGH.	0x0
2	EM2	-	External Match 2. This bit reflects the state of output MAT2, whether or not this output is connected to a pin. When a match occurs between the TC and MR2, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by EMR[9:8]. This bit is driven to the MAT pins if the match function is selected via IOCON. 0 = LOW. 1 = HIGH.	0x0

Table 308. External match register (EMR, offset 0x3C) bit description

Bit	Symbol	Value	Description	Reset value
3	EM3	-	External Match 3. This bit reflects the state of output MAT3, whether or not this output is connected to a pin. When a match occurs between the TC and MR3, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by MR[11:10]. This bit is driven to the MAT pins if the match function is selected via IOCON. 0 = LOW. 1 = HIGH.	0x0
5:4	EMC0		External Match Control 0. Determines the functionality of External Match 0.	0x0
		0x0	Do Nothing.	
		0x1	Clear. Clear the corresponding External Match bit/output to 0 (MAT0 pin is LOW if pinned out).	
		0x2	Set. Set the corresponding External Match bit/output to 1 (MAT0 pin is HIGH if pinned out).	
		0x3	Toggle. Toggle the corresponding External Match bit/output.	
7:6	EMC1		External Match Control 1. Determines the functionality of External Match 1.	0x0
		0x0	Do Nothing.	
		0x1	Clear. Clear the corresponding External Match bit/output to 0 (MAT1 pin is LOW if pinned out).	
		0x2	Set. Set the corresponding External Match bit/output to 1 (MAT1 pin is HIGH if pinned out).	
		0x3	Toggle. Toggle the corresponding External Match bit/output.	
9:8	EMC2		External Match Control 2. Determines the functionality of External Match 2.	0x0
		0x0	Do Nothing.	
		0x1	Clear. Clear the corresponding External Match bit/output to 0 (MAT2 pin is LOW if pinned out).	
		0x2	Set. Set the corresponding External Match bit/output to 1 (MAT2 pin is HIGH if pinned out).	
		0x3	Toggle. Toggle the corresponding External Match bit/output.	
11:10	EMC3		External Match Control 3. Determines the functionality of External Match 3.	0x0
		0x0	Do Nothing.	
		0x1	Clear. Clear the corresponding External Match bit/output to 0 (MAT3 pin is LOW if pinned out).	
		0x2	Set. Set the corresponding External Match bit/output to 1 (MAT3 pin is HIGH if pinned out).	
		0x3	Toggle. Toggle the corresponding External Match bit/output.	
31:12	-	-	Reserved. Read value is undefined, only zero should be written.	-

16.7.11 Count Control Register

The Count Control Register (CTCR) is used to select between Timer and Counter mode, and in Counter mode to select the pin and edge(s) for counting.

When Counter Mode is chosen as a mode of operation, the CAP input (selected by the CTCR bits 3:2) is sampled on every rising edge of the APB bus clock. After comparing two consecutive samples of this CAP input, one of the following four events is recognized: rising edge, falling edge, either of edges or no changes in the level of the selected CAP input. Only if the identified event occurs and the event corresponds to the one selected by bits 1:0 in the CTCR register, will the Timer Counter register be incremented.

Effective processing of the externally supplied clock to the counter has some limitations. Since two successive rising edges of the APB bus clock are used to identify only one edge on the CAP selected input, the frequency of the CAP input cannot exceed one half of the APB bus clock. Consequently, duration of the HIGH/LOWLOW levels on the same CAP input in this case cannot be shorter than 1/APB bus clock.

Bits 7:4 of this register are also used to enable and configure the capture-clears-timer feature. This feature allows for a designated edge on a particular CAP input to reset the timer to all zeros. Using this mechanism to clear the timer on the leading edge of an input pulse and performing a capture on the trailing edge, permits direct pulse-width measurement using a single capture input without the need to perform a subtraction operation in software.

Table 309. Count Control Register (CTCR, offset 0x70) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	CTMODE		Counter/Timer Mode This field selects which rising APB bus clock edges can increment Timer's Prescale Counter (PC), or clear PC and increment Timer Counter (TC). Timer Mode: the TC is incremented when the Prescale Counter matches the Prescale Register.	0x0
		0x0	Timer Mode. Incremented every rising APB bus clock edge.	
		0x1	Counter Mode rising edge. TC is incremented on rising edges on the CAP input selected by bits 3:2.	
		0x2	Counter Mode falling edge. TC is incremented on falling edges on the CAP input selected by bits 3:2.	
		0x3	Counter Mode dual edge. TC is incremented on both edges on the CAP input selected by bits 3:2.	
3:2	CINSEL		Count Input Select When bits 1:0 in this register are not 0, these bits select which CAP pin is sampled for clocking. **Note:** If Counter mode is selected for a particular CAPn input in the CTCR, the 3 bits for that input in the Capture Control Register (CCR) must be programmed as 000. However, capture and/or interrupt can be selected for the other 3 CAPn inputs in the same timer.	0x0
		0x0	Channel 0. CAPn.0 for CTIMERn	
		0x1	Channel 1. CAPn.1 for CTIMERn	
		0x2	Channel 2. CAPn.2 for CTIMERn	
		0x3	Channel 3. CAPn.3 for CTIMERn	
4	ENCC	-	Setting this bit to 1 enables clearing of the timer and the prescaler when the capture-edge event specified in bits 7:5 occurs.	0x0

Table 309. Count Control Register (CTCR, offset 0x70) bit description

Bit	Symbol	Value	Description	Reset Value
7:5	SELCC		Edge select. When bit 4 is 1, these bits select which capture input edge will cause the timer and prescaler to be cleared. These bits have no effect when bit 4 is low. Note that different part number and package variations may provide different capture input pin functions.	0x0
		0x0	Channel 0 Rising Edge. Rising edge of the signal on capture channel 0 clears the timer (if bit 4 is set).	
		0x1	Channel 0 Falling Edge. Falling edge of the signal on capture channel 0 clears the timer (if bit 4 is set).	
		0x2	Channel 1 Rising Edge. Rising edge of the signal on capture channel 1 clears the timer (if bit 4 is set).	
		0x3	Channel 1 Falling Edge. Falling edge of the signal on capture channel 1 clears the timer (if bit 4 is set).	
		0x4	Channel 2 Rising Edge. Rising edge of the signal on capture channel 2 clears the timer (if bit 4 is set).	
		0x5	Channel 2 Falling Edge. Falling edge of the signal on capture channel 2 clears the timer (if bit 4 is set).	
		0x6	Channel 3 Rising Edge. Rising edge of the signal on capture channel 3 clears the timer (if bit 4 is set).	
		0x57	Channel 3 Falling Edge. Falling edge of the signal on capture channel 3 clears the timer (if bit 4 is set).	
31:8	-	-	Reserved. Read value is undefined, only zero should be written.	-

16.7.12 PWM Control Register

The PWM Control Register is used to configure the match outputs as PWM outputs. Each match output can be independently set to perform either as PWM output or as match output whose function is controlled by the External Match Register (EMR). Note that different part number and package variations may provide different match output pin functions.

For each timer, a maximum of three single edge controlled PWM outputs can be selected on the MATn.2:0 outputs. One additional match register determines the PWM cycle length. When a match occurs in any of the other match registers, the PWM output is set to HIGH. The timer is reset by the match register that is configured to set the PWM cycle length. When the timer is reset to zero, all currently HIGH match outputs configured as PWM outputs are cleared.

Table 310: PWM Control Register (PWMC, offset 0x74)) bit description

Bit	Symbol	Value	Description	Reset value
0	PWMEN0		PWM mode enable for channel0.	0x0
		0	Match. CTIMERn_MAT0 is controlled by EM0.	
		1	PWM. PWM mode is enabled for CTIMERn_MAT0.	
1	PWMEN1		PWM mode enable for channel1.	0x0
		0	Match. CTIMERn_MAT01 is controlled by EM1.	
		1	PWM. PWM mode is enabled for CTIMERn_MAT1.	

Table 310: PWM Control Register (PWMC, offset 0x74)) bit description

Bit	Symbol	Value	Description	Reset value
2	PWMEN2		PWM mode enable for channel2.	0x0
		0	Match. CTIMERn_MAT2 is controlled by EM2.	
		1	PWM. PWM mode is enabled for CTIMERn_MAT2.	
3	PWMEN3		PWM mode enable for channel3. **Note:** It is recommended to use match channel 3 to set the PWM cycle.	0x0
		0	Match. CTIMERn_MAT3 is controlled by EM3.	
		1	PWM. PWM mode is enabled for CTIMERn_MAT3.	
31:4	-	-	Reserved. Read value is undefined, only zero should be written.	-

16.8 Functional description

Figure 38 shows a timer configured to reset the count and generate an interrupt on match. The prescaler is set to 2 and the match register set to 6. At the end of the timer cycle where the match occurs, the timer count is reset. This gives a full length cycle to the match value. The interrupt indicating that a match occurred is generated in the next clock after the timer reached the match value.

Figure 39 shows a timer configured to stop and generate an interrupt on match. The prescaler is again set to 2 and the match register set to 6. In the next clock after the timer reaches the match value, the timer enable bit in TCR is cleared, and the interrupt indicating that a match occurred is generated.

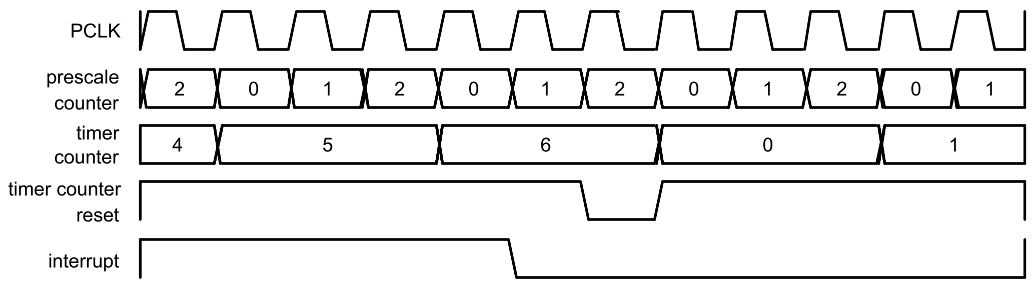


Fig 38. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled

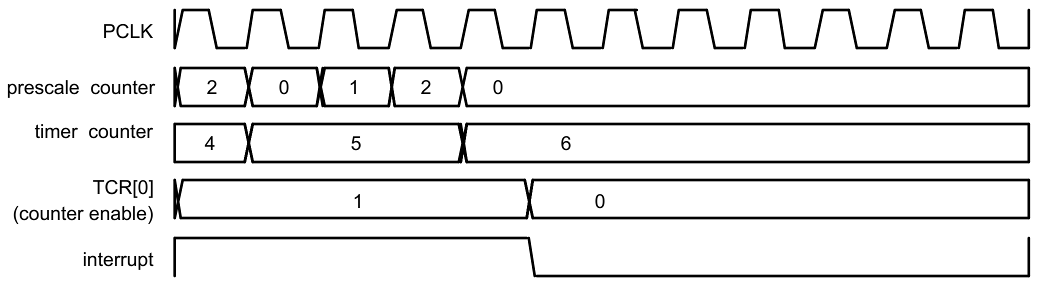


Fig 39. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled

16.8.1 Rules for single edge controlled PWM outputs

1. All single edge controlled PWM outputs go LOW at the beginning of each PWM cycle (timer is set to zero) unless their match value is equal to zero.
2. Each PWM output will go HIGH when its match value is reached. If no match occurs (i.e. the match value is greater than the PWM cycle length), the PWM output remains continuously LOW.
3. If a match value larger than the PWM cycle length is written to the match register, and the PWM signal is HIGH already, then the PWM signal will be cleared with the start of the next PWM cycle.

4. If a match register contains the same value as the timer reset value (the PWM cycle length), then the PWM output will be reset to LOW on the next clock tick after the timer reaches the match value. Therefore, the PWM output will always consist of a one clock tick wide positive pulse with a period determined by the PWM cycle length (i.e. the timer reload value).
5. If a match register is set to zero, then the PWM output will go to HIGH the first time the timer goes back to zero and will stay HIGH continuously.

Note: When the match outputs are selected to perform as PWM outputs, the timer reset (MRnR) and timer stop (MRnS) bits in the Match Control Register MCR must be set to zero except for the match register setting the PWM cycle length. For this register, set the MRnR bit to one to enable the timer reset when the timer value matches the value of the corresponding match register.

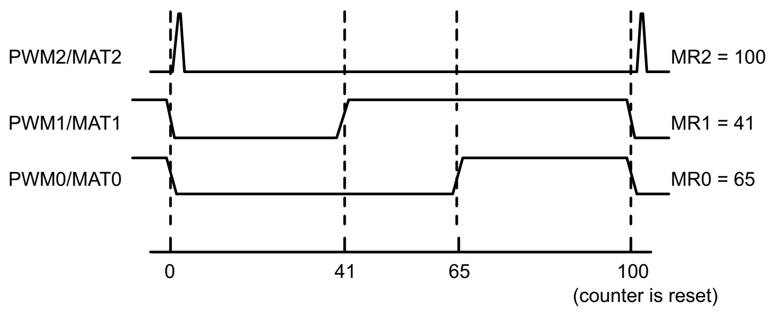


Fig 40. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.

16.8.2 DMA operation

DMA requests are generated by a match of the Timer Counter (TC) register value to either Match Register 0 (MR0) or Match Register 1 (MR1). This is not connected to the operation of the Match outputs controlled by the EMR register. Each match sets a DMA request flag, which is connected to the DMA controller. In order to have an effect, the DMA controller must be configured correctly.

When a timer is initially set up to generate a DMA request, the request may already be asserted before a match condition occurs. An initial DMA request may be avoided by having software write a one to the interrupt flag location, as if clearing a timer interrupt. See Section 16.7.1. A DMA request will be cleared automatically when it is acted upon by the DMA controller.

Note: because timer DMA requests are generated whenever the timer value is equal to the related Match Register value, DMA requests are always generated when the timer is running, unless the Match Register value is higher than the upper count limit of the timer. It is important not to select and enable timer DMA requests in the DMA block unless the timer is correctly configured to generate valid DMA requests.

UM11071

Chapter 17: LPC51U68 Windowed Watchdog Timer (WWDT)

Rev. 1.1 — 17 May 2018 User manual

17.1 How to read this chapter

The watchdog timer is available on all LPC51U68 devices.

17.2 Features

- Internally resets chip if not reloaded during the programmable time-out period.
- Optional windowed operation requires reload to occur between a minimum and maximum time-out period, both programmable.
- Optional warning interrupt can be generated at a programmable time prior to watchdog time-out.
- Programmable 24-bit timer with internal fixed pre-scaler.
- Selectable time period from 1,024 watchdog clocks ($T_{WDCLK} \times 256 \times 4$) to over 67 million watchdog clocks ($T_{WDCLK} \times 2^{24} \times 4$) in increments of 4 watchdog clocks.
- “Safe” watchdog operation. Once enabled, requires a hardware reset or a Watchdog reset to be disabled.
- Incorrect feed sequence causes immediate watchdog event if enabled.
- The watchdog reload value can optionally be protected such that it can only be changed after the “warning interrupt” time is reached.
- Flag to indicate Watchdog reset.
- The Watchdog clock (WDCLK) source is a selectable frequency in the range of 6 kHz to 1.5 MHz (see Section 6.5.44 for details of Watchdog oscillator configuration). The accuracy of this clock is limited to +/- 40% over temperature, voltage, and silicon processing variations. To determine the actual watchdog oscillator output, use the frequency measure block. See Section 6.2.3.
- The Watchdog timer can be configured to run in deep-sleep mode.
- Debug mode.

17.3 Basic configuration

Configuration of the WWDT is accomplished as follows:

- Turn on and configure the Watchdog oscillator. See the PDEN_WDT_OSC bit in the PDRUNCG0 register (Section 6.5.48), and the Watchdog oscillator control register (Section 6.5.44).
- Enable the register interface (WWDT bus clock): set the WWDT bit in the AHBCLKCTRL0 register, Table 115.
- For waking up from a WWDT interrupt, enable the watchdog interrupt for wake-up in the STARTER0 register (Table 155).

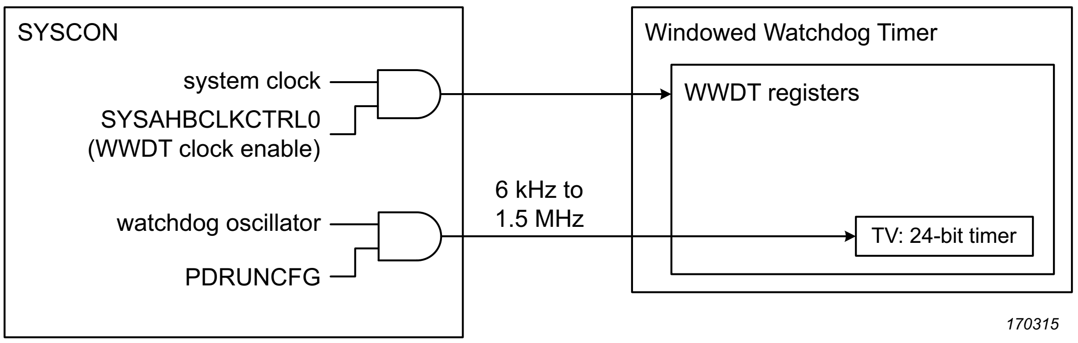


Fig 41. WWDT clocking

17.4 Pin description

The WWDT has no external pins.

17.5 General description

The purpose of the Watchdog Timer is to reset or interrupt the microcontroller within a programmable time if it enters an erroneous state. When enabled, a watchdog reset is generated if the user program fails to feed (reload) the Watchdog within a predetermined amount of time.

When a watchdog window is programmed, an early watchdog feed is also treated as a watchdog event. This allows preventing situations where a system failure may still feed the watchdog. For example, application code could be stuck in an interrupt service that contains a watchdog feed. Setting the window such that this would result in an early feed will generate a watchdog event, allowing for system recovery.

The Watchdog consists of a fixed (divide by 4) pre-scaler and a 24-bit counter which decrements when clocked. The minimum value from which the counter decrements is 0xFF. Setting a value lower than 0xFF causes 0xFF to be loaded in the counter. Hence the minimum Watchdog interval is ($T_{WDCLK} \times 256 \times 4$) and the maximum Watchdog interval is ($T_{WDCLK} \times 2^{24} \times 4$) in multiples of ($T_{WDCLK} \times 4$). The Watchdog should be used in the following manner:

- Enable and configure the Watchdog oscillator as described in Section 17.3 “Basic configuration”
- Set the Watchdog timer constant reload value in the TC register.
- Set the Watchdog timer operating mode in the MOD register.
- Set a value for the watchdog window time in the WINDOW register if windowed operation is desired.
- Set a value for the watchdog warning interrupt in the WARNINT register if a warning interrupt is desired.
- Enable the Watchdog by writing 0xAA followed by 0x55 to the FEED register.
- The Watchdog must be fed again before the Watchdog counter reaches zero in order to prevent a watchdog event. If a window value is programmed, the feed must also occur after the watchdog counter passes that value.

When the Watchdog Timer is configured so that a watchdog event will cause a reset and the counter reaches zero, the CPU will be reset, loading the stack pointer and program counter from the vector table as for an external reset. The Watchdog time-out flag (WDTOF) can be examined to determine if the Watchdog has caused the reset condition. The WDTOF flag must be cleared by software.

When the Watchdog Timer is configured to generate a warning interrupt, the interrupt will occur when the counter is no longer greater than the value defined by the WARNINT register.

17.5.1 Block diagram

The block diagram of the Watchdog is shown below in the Figure 42. The synchronization logic (APB bus clock to WDCLK) is not shown in the block diagram.

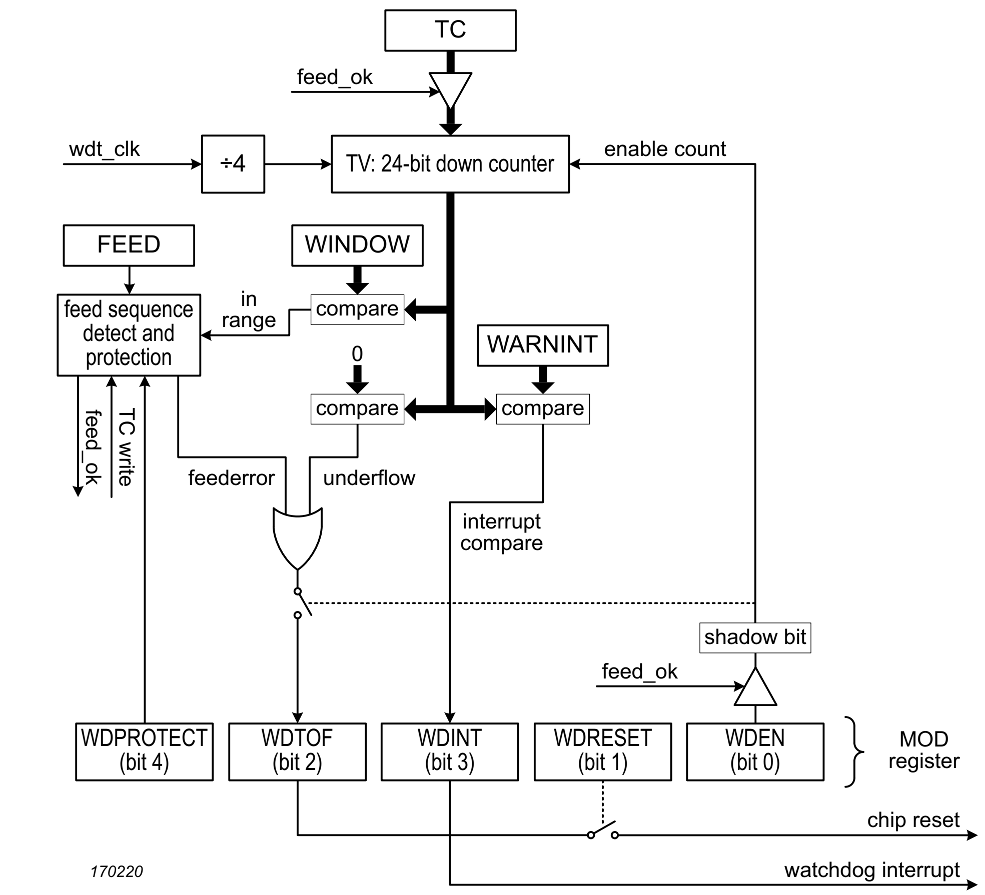


Fig 42. Windowed Watchdog timer block diagram

17.5.2 Clocking and power control

The watchdog timer block uses two clocks: APB bus clock and WDCLK. The APB bus clock is used for the APB accesses to the watchdog registers and is derived from the system clock (see Figure 9). The WDCLK is used for the watchdog timer counting and is derived from the watchdog oscillator.

The synchronization logic between the two clock domains works as follows: When the MOD and TC registers are updated by APB operations, the new value will take effect in 3 WDCLK cycles on the logic in the WDCLK clock domain.

When the watchdog timer is counting on WDCLK, the synchronization logic will first lock the value of the counter on WDCLK and then synchronize it with the APB bus clock, so that the CPU can read the TV register.

Remark: Because of the synchronization step, software must add a delay of three WDCLK clock cycles between the feed sequence and the time the WDPROTECT bit is enabled in the MOD register. The length of the delay depends on the selected watchdog clock WDCLK.

17.5.3 Using the WWDT lock features

The WWDT supports several lock features which can be enabled to ensure that the WWDT is running at all times:

- Disabling the WWDT clock source
- Changing the WWDT reload value

17.5.3.1 Disabling the WWDT clock source

If bit 5 in the WWDT MOD register is set, the WWDT clock source is locked and can not be disabled either by software or by hardware when sleep or deep-sleep modes are entered. Therefore, the user must ensure that the watchdog oscillator for each power mode is enabled **before** setting bit 5 in the MOD register.

17.5.3.2 Changing the WWDT reload value

If bit 4 is set in the WWDT MOD register, the watchdog time-out value (TC) can be changed only after the counter is below the value of WARNINT and WINDOW.

The reload overwrite lock mechanism can only be disabled by a reset of any type.

17.6 Register description

The Watchdog Timer contains the registers shown in Table 311.

The reset value reflects the data stored in used bits only. It does not include the content of reserved bits.

Table 311. Register overview: Watchdog timer (base address 0x4000 C000)

Name	Access	Offset	Description	Reset value	Section
MOD	R/W	0x000	Watchdog mode. This register contains the basic mode and status of the Watchdog Timer.	0x0	17.6.1
TC	R/W	0x004	Watchdog timer constant. This 24-bit register determines the time-out value.	0xFF	17.6.2
FEED	WO	0x008	Watchdog feed sequence. Writing 0xAA followed by 0x55 to this register reloads the Watchdog timer with the value contained in TC.	-	17.6.3
TV	RO	0x00C	Watchdog timer value. This 24-bit register reads out the current value of the Watchdog timer.	0xFF	17.6.4
-	-	0x010	Reserved	-	-
WARNINT	R/W	0x014	Watchdog Warning Interrupt compare value.	0x0	17.6.5
WINDOW	R/W	0x018	Watchdog Window compare value.	0xFF FFFF	17.6.6

17.6.1 Watchdog mode register

The MOD register controls the operation of the Watchdog. Note that a watchdog feed must be performed before any changes to the MOD register take effect.

Table 312. Watchdog mode register (MOD, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value
0	WDEN		Watchdog enable bit. Once this bit is set to one and a watchdog feed is performed, the watchdog timer will run permanently.	0x0
		0	Stop. The watchdog timer is stopped.	
		1	Run. The watchdog timer is running.	
1	WDRESET		Watchdog reset enable bit. Once this bit has been written with a 1 it cannot be re-written with a 0.	0x0
		0	Interrupt. A watchdog time-out will not cause a chip reset.	
		1	Reset. A watchdog time-out will cause a chip reset.	
2	WDTOF	-	Watchdog time-out flag. Set when the watchdog timer times out, by a feed error, or by events associated with WDPROTECT. Cleared by software writing a 0 to this bit position. Causes a chip reset if WDRESET = 1.	0x0 [1]
3	WDINT	-	Warning interrupt flag. Set when the timer is at or below the value in WARNINT. Cleared by software writing a 1 to this bit position. Note that this bit cannot be cleared while the WARNINT value is equal to the value of the TV register. This can occur if the value of WARNINT is 0 and the WDRESET bit is 0 when TV decrements to 0.	0x0
4	WDPROTECT		Watchdog update mode. This bit can be set once by software and is only cleared by a reset.	0x0
		0	Flexible. The watchdog time-out value (TC) can be changed at any time.	
		1	Threshold. The watchdog time-out value (TC) can be changed only after the counter is below the value of WARNINT and WINDOW.	
5	LOCK	-	Once this bit is set to one and a watchdog feed is performed, disabling or powering down the watchdog oscillator is prevented by hardware. This bit can be set once by software and is only cleared by any reset.	0x0
31:6	-	-	Reserved. Read value is undefined, only zero should be written.	-

[1] Only an external or power-on reset has this effect.

Once the **WDEN**, **WDPROTECT**, or **WDRESET** bits are set they can not be cleared by software. Both flags are cleared by an external reset or a Watchdog timer reset.

WDTOF The Watchdog time-out flag is set when the Watchdog times out, when a feed error occurs, or when PROTECT =1 and an attempt is made to write to the TC register. This flag is cleared by software writing a 0 to this bit.

WDINT The Watchdog interrupt flag is set when the Watchdog counter is no longer greater than the value specified by WARNINT. This flag is cleared when any reset occurs, and is cleared by software by writing a 0 to this bit.

In all power modes except deep power-down mode, a Watchdog reset or interrupt can occur when the watchdog is running and has an operating clock source. The watchdog oscillator can be configured to keep running in sleep and deep-sleep modes.

If a watchdog interrupt occurs in sleep or deep-sleep mode, and the WWDT interrupt is enabled in the NVIC, the device will wake up. Note that in deep-sleep mode, the WWDT interrupt must be enabled in the STARTER0 register in addition to the NVIC.

See the following registers:

Table 155 “Start enable register 0 (STARTER0, main syscon: offset 0x680) bit description”

Table 313. Watchdog operating modes selection

WDEN	WDRESET	Mode of Operation
0	X (0 or 1)	Debug/Operate without the Watchdog running.
1	0	Watchdog interrupt mode: the watchdog warning interrupt will be generated but watchdog reset will not. When this mode is selected, the watchdog counter reaching the value specified by WARNINT will set the WDINT flag and the Watchdog interrupt request will be generated.
1	1	Watchdog reset mode: both the watchdog interrupt and watchdog reset are enabled. When this mode is selected, the watchdog counter reaching the value specified by WARNINT will set the WDINT flag and the Watchdog interrupt request will be generated, and the watchdog counter reaching zero will reset the microcontroller. A watchdog feed prior to reaching the value of WINDOW will also cause a watchdog reset.

17.6.2 Watchdog Timer Constant register

The TC register determines the time-out value. Every time a feed sequence occurs the value in the TC is loaded into the Watchdog timer. The TC resets to 0x00 00FF. Writing a value below 0xFF will cause 0x00 00FF to be loaded into the TC. Thus the minimum time-out interval is $T_{WDCLK} \times 256 \times 4$.

If the WDPROTECT bit in MOD = 1, an attempt to change the value of TC before the watchdog counter is below the values of WARNINT and WINDOW will cause a watchdog reset and set the WDTOF flag.

Table 314. Watchdog Timer Constant register (TC, offset 0x04) bit description

Bit	Symbol	Description	Reset value
23:0	COUNT	Watchdog time-out value.	0x00 00FF
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

17.6.3 Watchdog Feed register

Writing 0xAA followed by 0x55 to this register will reload the Watchdog timer with the TC value. This operation will also start the Watchdog if it is enabled via the MOD register. Setting the WDEN bit in the MOD register is not sufficient to enable the Watchdog. A valid feed sequence must be completed after setting WDEN before the Watchdog is capable of generating a reset. Until then, the Watchdog will ignore feed errors.

After writing 0xAA to FEED, access to any Watchdog register other than writing 0x55 to FEED causes an immediate reset/interrupt when the Watchdog is enabled, and sets the WDTOF flag. The reset will be generated during the second APB bus clock following an incorrect access to a Watchdog register during a feed sequence.

It is good practice to disable interrupts around a feed sequence, if the application is such that an interrupt might result in rescheduling processor control away from the current task in the middle of the feed, and then lead to some other access to the WDT before control is returned to the interrupted task.

Table 315. Watchdog Feed register (FEED, offset 0x08) bit description

Bit	Symbol	Description	Reset value
7:0	FEED	Feed value should be 0xAA followed by 0x55.	-
31:8	-	Reserved. Read value is undefined, only zero should be written.	-

17.6.4 Watchdog Timer Value register

The TV register is used to read the current value of Watchdog timer counter.

When reading the value of the 24-bit counter, the lock and synchronization procedure takes up to 6 WDCLK cycles plus 6 APB bus clock cycles, so the value of TV is older than the actual value of the timer when it's being read by the CPU.

Table 316. Watchdog Timer Value register (TV, offset 0x0C) bit description

Bit	Symbol	Description	Reset value
23:0	COUNT	Counter timer value.	0x00 00FF
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

17.6.5 Watchdog Timer Warning Interrupt register

The WARNINT register determines the watchdog timer counter value that will generate a watchdog interrupt. When the watchdog timer counter is no longer greater than the value defined by WARNINT, an interrupt will be generated after the subsequent WDCLK.

A match of the watchdog timer counter to WARNINT occurs when the bottom 10 bits of the counter have the same value as the 10 bits of WARNINT, and the remaining upper bits of the counter are all 0. This gives a maximum time of 1,023 watchdog timer counts (4,096 watchdog clocks) for the interrupt to occur prior to a watchdog event. If WARNINT is 0, the interrupt will occur at the same time as the watchdog event.

Table 317. Watchdog Timer Warning Interrupt register (WARNINT, offset 0x14) bit description

Bit	Symbol	Description	Reset value
9:0	WARNINT	Watchdog warning interrupt compare value.	0x0
31:10	-	Reserved, only zero should be written.	-

17.6.6 Watchdog Timer Window register

The WINDOW register determines the highest TV value allowed when a watchdog feed is performed. If a feed sequence occurs when TV is greater than the value in WINDOW, a watchdog event will occur.

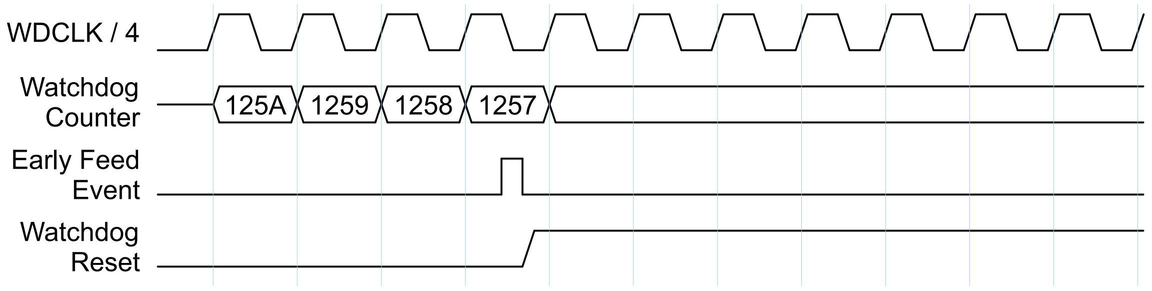
WINDOW resets to the maximum possible TV value, so windowing is not in effect.

Table 318. Watchdog Timer Window register (WINDOW, offset 0x18) bit description

Bit	Symbol	Description	Reset value
23:0	WINDOW	Watchdog window value.	0xFF FFFF
31:24	-	Reserved, only zero should be written.	-

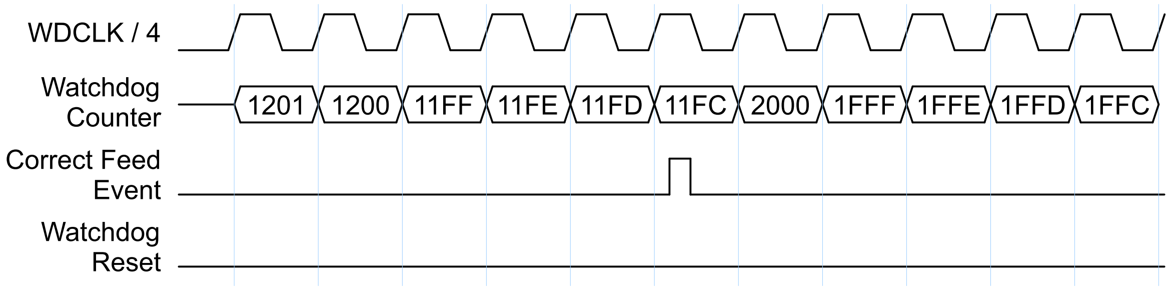
17.7 Functional description

The following figures illustrate several aspects of Watchdog Timer operation.



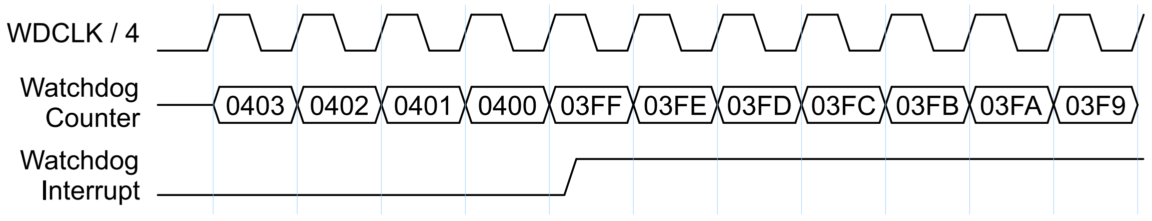
Conditions:
WINDOW = 0x1200
WARNINT = 0x3FF
TC = 0x2000

Fig 43. Early watchdog feed with windowed mode enabled



Conditions:
WINDOW = 0x1200
WARNINT = 0x3FF
TC = 0x2000

Fig 44. Correct watchdog feed with windowed mode enabled



Conditions:
WINDOW = 0x1200
WARNINT = 0x3FF
TC = 0x2000

Fig 45. Watchdog warning interrupt

UM11071

Chapter 18: LPC51U68 Real-Time Clock (RTC)

Rev. 1.1 — 17 May 2018 User manual

18.1 How to read this chapter

The RTC is available on all LPC51U68 devices.

18.2 Features

- The RTC and its independent oscillator operate directly from the device power pins, not using the on-chip regulator. The RTC oscillator has the following clock outputs:
  - 32 kHz clock, selectable for system clock and CLKOUT pin.
  - 1 Hz clock for RTC timing.
  - 1 kHz clock for high-resolution RTC timing.
- 32-bit, 1 Hz RTC counter and associated match register for alarm generation.
- Separate 16-bit high-resolution/wake-up timer clocked at 1 kHz for 1 ms resolution with a more that one minute maximum time-out period.
- RTC alarm and high-resolution/wake-up timer time-out each generate independent interrupt requests. Either time-out can wake up the part from any of the low power modes, including deep power-down.

18.3 Basic configuration

Configure the RTC as follows:

- Use the AHBCLKCTRL0 register (Table 115) to enable the clock to the RTC register interface and peripheral clock.
- For RTC software reset use the RTC CTRL register. See Table 321. The RTC is reset only by initial power-up of the device or when an RTC software reset is applied, it is not initialized by other system resets.
- The RTC provides an interrupt to the NVIC for the RTC_WAKE and RTC_ALARM functions, see Table 79.
- To enable the RTC interrupts for waking up from deep-sleep mode, enable the interrupts in the STARTER0 register (Table 155) and the NVIC.
- To enable the RTC interrupts for waking up from deep power-down, enable the appropriate RTC clock and wake-up in the RTC CTRL register (Table 321).
- If enabled, the RTC and its oscillator continue running in all reduced power modes as long as power is supplied to the device. So, the 32 kHz output is always available to be enabled for syscon clock generation (see Table 144). Once enabled, the 32 kHz clock can be selected for the system clock or be observed through the CLKOUT pin. The 1 Hz output is enabled in the RTC CTRL register (RTC_EN bit). Once the 1 Hz output is enabled, the 1 kHz output for the high-resolution wake-up timer can be enabled in the RTC CTRL register (RTC1KHZ_EN bit).
- If the 32 kHz output of the RTC is used by another part of the system, enable it via the EN bit in the RTCOSCCTRL register. See Table 144.

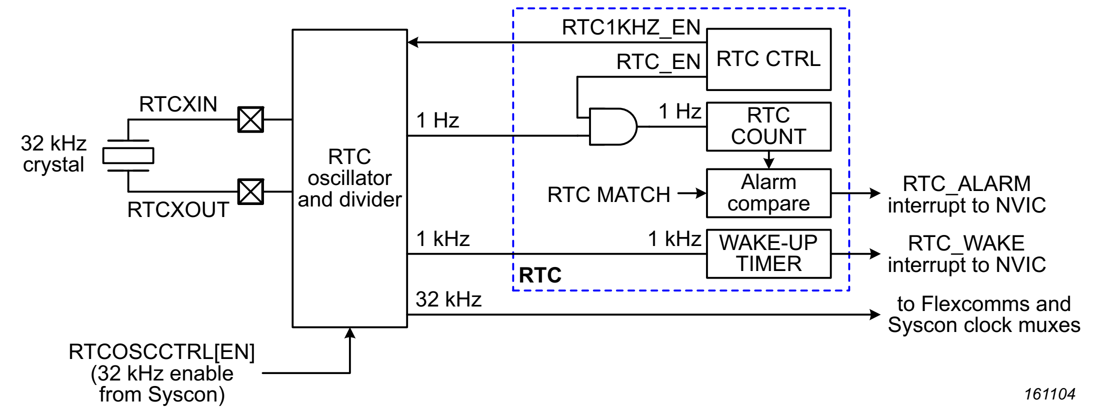

RTC1KHZ_EN
RTC_EN
RTC CTRL
RTCXIN
32 kHz crystal
RTCXOUT
RTC oscillator and divider
1 Hz
1 Hz
RTC COUNT
RTC MATCH
Alarm compare
RTC_ALARM interrupt to NVIC
1 kHz
1 kHz
WAKE-UP TIMER
RTC_WAKE interrupt to NVIC
RTC
32 kHz
to Flexcomms and Syscon clock muxes
RTCOSCCTRL[EN] (32 kHz enable from Syscon)
161104

Fig 46. RTC clocking

18.3.1 RTC timers

The RTC contains two timers:

1. The main RTC timer. This 32-bit timer uses a 1 Hz clock and is intended to run continuously as a real-time clock. When the timer value reaches a match value, an interrupt is raised. The alarm interrupt can also wake up the part from any low power mode if enabled.
2. The high-resolution/wake-up timer. This 16-bit timer uses a 1 kHz clock and operates as a one-shot down timer. Once the timer is loaded, it starts counting down to 0 at which point an interrupt is raised. The interrupt can wake up the part from any low power mode if enabled. This timer is intended to be used for timed wake-up from deep-sleep or deep power-down modes. The high-resolution wake-up timer can be disabled to conserve power if not used.

18.4 General description

18.4.1 Real-time clock

The real-time clock is a 32-bit up-counter which can be cleared or initialized by software. Once enabled, it counts continuously at a 1 Hz clock rate as long as the device is powered up and the RTC remains enabled.

The main purpose of the RTC is to count seconds and generate an alarm interrupt to the processor whenever the counter value equals the value programmed into the associated 32-bit match register.

If the part is in one of the reduced-power modes (deep-sleep, deep power-down) an RTC alarm interrupt can also wake up the part to exit the power mode and begin normal operation.

18.4.2 High-resolution/wake-up timer

The time interval required for many applications, including waking the part up from a low-power mode, will often demand a greater degree of resolution than the one-second minimum interval afforded by the main RTC counter. For these applications, a higher frequency secondary timer has been provided.

This secondary timer is an independent, stand-alone wake-up or general-purpose timer for timing intervals of up to 64 seconds with approximately one millisecond of resolution.

The High-Resolution/Wake-up Timer is a 16-bit down counter which is clocked at a 1 kHz rate when it is enabled. Writing any non-zero value to this timer will automatically enable the counter and launch a countdown sequence. When the counter is being used as a wake-up timer, this write can occur just prior to entering a reduced power mode.

When a starting count value is loaded, the High-Resolution/Wake-up Timer will turn on, count from the pre-loaded value down to zero, generate an interrupt and/or a wake-up command, and then turn itself off until re-launched by a subsequent software write.

18.4.3 RTC power

The RTC module and the oscillator that drives it, run directly from device power pins. As a result, the RTC will continue operating in deep power-down mode when power is internally turned off to the rest of the device.

18.5 Pin description

Table 319 gives a summary of pins related to the RTC.

Table 319. RTC pin description

Pin	Type	Description
RTCXIN	Input	RTC oscillator input.
RTCXOUT	Output	RTC oscillator output.

18.6 Register description

Reset Values pertain to initial power-up of the device or when an RTC software reset is applied (except where noted). This block is not initialized by any other system reset.

Table 320. Register overview: RTC (base address 0x4002 C000)

Name	Access	Offset	Description	SWRESET bit in CTRL = 1	Reset value	Section
CTRL	R/W	0x000	RTC control register	0x1	0x1	18.6.1
MATCH	R/W	0x004	RTC match register	0xFFFF FFFF	0xFFFF FFFF	18.6.2
COUNT	R/W	0x008	RTC counter register	0x0	0x0	18.6.3
WAKE	R/W	0x00C	High-resolution/wake-up timer control register	0x0	0x0	18.6.4

18.6.1 RTC CTRL register

This register controls which clock the RTC uses (1 kHz or 1 Hz) and enables the two RTC interrupts to wake up the part from deep power-down. To wake up the part from deep-sleep mode, enable the RTC interrupts in the system control block STARTLOGIC1 register.

Table 321. RTC control register (CTRL, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value
0	SWRESET		Software reset control	0x1
		0	Not in reset. The RTC is not held in reset. This bit must be cleared prior to configuring or initiating any operation of the RTC.	
		1	In reset. The RTC is held in reset. All register bits within the RTC will be forced to their reset value except the RTC_OSC_PD bit in this register. This bit must be cleared before writing to any register in the RTC - including writes to set any of the other bits within this register. Do not attempt to write to any bits of this register at the same time that the reset bit is being cleared.	
1	-	-	Reserved. Read value is undefined, only zero should be written.	-
2	ALARM1HZ		RTC 1 Hz timer alarm flag status.	0x0
		0	No match. No match has occurred on the 1 Hz RTC timer. Writing a 0 has no effect.	
		1	Match. A match condition has occurred on the 1 Hz RTC timer. This flag generates an RTC alarm interrupt request RTC_ALARM which can also wake up the part from any low power mode. Writing a 1 clears this bit.	
3	WAKE1KHZ		RTC 1 kHz timer wake-up flag status.	0x0
		0	Run. The RTC 1 kHz timer is running. Writing a 0 has no effect.	
		1	Time-out. The 1 kHz high-resolution/wake-up timer has timed out. This flag generates an RTC wake-up interrupt request which can also wake up the part from any low power mode. Writing a 1 clears this bit.	
4	ALARMDPD_EN		RTC 1 Hz timer alarm enable for deep power-down.	0x0
		0	Disable. A match on the 1 Hz RTC timer will not bring the part out of deep power-down mode.	
		1	Enable. A match on the 1 Hz RTC timer bring the part out of deep power-down mode.	
5	WAKEDPD_EN		RTC 1 kHz timer wake-up enable for deep power-down.	0x0
		0	Disable. A match on the 1 kHz RTC timer will not bring the part out of deep power-down mode.	
		1	Enable. A match on the 1 kHz RTC timer bring the part out of deep power-down mode.	
6	RTC1KHZ_EN		RTC 1 kHz clock enable. This bit can be set to 0 to conserve power if the 1 kHz timer is not used. This bit has no effect when the RTC is disabled (bit 7 of this register is 0).	0x0
		0	Disable. A match on the 1 kHz RTC timer will not bring the part out of deep power-down mode. The WAKE1KHZ flag clears when disabled.	
		1	Enable. The 1 kHz RTC timer is enabled.	
7	RTC_EN		RTC enable.	0x0
		0	Disable. The RTC 1 Hz and 1 kHz clocks are shut down and RTC operation is disabled. This bit should be 0 when writing to load a value in the RTC counter register.	
		1	Enable. The 1 Hz RTC clock is running and RTC operation is enabled. This bit must be set to initiate operation of the RTC. The first clock to the RTC counter occurs 1 s after this bit is set. To also enable the high-resolution, 1 kHz clock, set bit 6 in this register.	

Table 321. RTC control register (CTRL, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset value
8	RTC_OSC_PD		RTC oscillator power-down control.	0x0
		0	RTC oscillator is powered up and can output a clock if a crystal is correctly connected externally.	
		1	RTC oscillator is powered-down and outputs no clock.	
31:9	-	-	Reserved. Read value is undefined, only zero should be written.	-

18.6.2 RTC match register

Table 322. RTC match register (MATCH, offset 0x04) bit description

Bit	Symbol	Description	Reset value
31:0	MATVAL	Contains the match value against which the 1 Hz RTC timer will be compared to set the alarm flag RTC_ALARM and generate an alarm interrupt/wake-up if enabled.	0xFFFF FFFF

18.6.3 RTC counter register

Table 323. RTC counter register (COUNT, offset 0x08) bit description

Bit	Symbol	Description	Reset value
31:0	VAL	A read reflects the current value of the main, 1 Hz RTC timer. A write loads a new initial value into the timer. The RTC counter will count up continuously at a 1 Hz rate once the RTC Software Reset is removed (by clearing bit 0 of the CTRL register). **Remark:** No synchronization is provided to prevent a read of the counter register during a count transition. The suggested method to read a counter is to read the location twice and compare the results. If the values match, the time can be used. If they do not match, then the read should be repeated until two consecutive reads produce the same result. **Remark:** Only write to this register when the RTC_EN bit in the RTC CTRL Register is 0. The counter increments one second after the RTC_EN bit is set.	0x0

18.6.4 RTC high-resolution/wake-up register

Table 324. RTC high-resolution/wake-up register (WAKE, offset 0x0C) bit description

Bit	Symbol	Description	Reset value
15:0	VAL	A read reflects the current value of the high-resolution/wake-up timer. A write pre-loads a start count value into the wake-up timer and initializes a count-down sequence. Do not write to this register while counting is in progress.	0x0
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

UM11071

Chapter 19: LPC51U68 Multi-Rate Timer (MRT)

Rev. 1.1 — 17 May 2018 **User manual**

19.1 How to read this chapter

The MRT is available on all LPC51U68 parts.

19.2 Features

- 24-bit interrupt timer
- Four channels independently counting down from individually set values
- Repeat interrupt, one-shot interrupt, and one-shot bus stall modes

19.3 Basic configuration

Configuration of the MRT is accomplished as followings:

- In the AHBCLKCTRL1 register (Table 116), set the MRT bit to enable the clock to the register interface.
- Clear the MRT reset using the PRESETCTRL1 register (Table 109).
- The global MRT interrupt is connected to an interrupt slot in the NVIC (see Table 79).

19.4 Pin description

The MRT is not associated with any device pins.

19.5 General description

The Multi-Rate Timer (MRT) provides a repetitive interrupt timer with four channels. Each channel can be programmed with an independent time interval.

Each channel operates independently from the other channels in one of the following modes:

- Repeat interrupt mode. See Section 19.5.1.
- One-shot interrupt mode. See Section 19.5.2.
- One-shot stall mode. See Section 19.5.3.

The modes for each timer are set in the timer's control register. See Table 328.

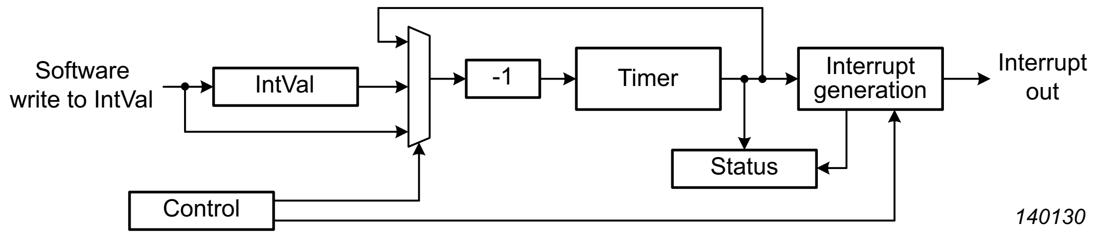


Fig 47. MRT block diagram

19.5.1 Repeat interrupt mode

The repeat interrupt mode generates repeated interrupts after a selected time interval. This mode can be used for software-based PWM or PPM applications.

When the timer n is in idle state, writing a non-zero value IVALUE to the INTVALn register immediately loads the time interval value IVALUE - 1, and the timer begins to count down from this value. When the timer reaches zero, an interrupt is generated, the value in the INTVALn register IVALUE - 1 is reloaded automatically, and the timer starts to count down again.

While the timer is running in repeat interrupt mode, the following actions can be performed:

- Change the interval value on the next timer cycle by writing a new value (>0) to the INTVALn register and setting the LOAD bit to 0. An interrupt is generated when the timer reaches zero. On the next cycle, the timer counts down from the new value.
- Change the interval value on-the-fly immediately by writing a new value (>0) to the INTVALn register and setting the LOAD bit to 1. The timer immediately starts to count down from the new timer interval value. An interrupt is generated when the timer reaches 0.
- Stop the timer at the end of time interval by writing a 0 to the INTVALn register and setting the LOAD bit to 0. An interrupt is generated when the timer reaches zero.
- Stop the timer immediately by writing a 0 to the INTVALn register and setting the LOAD bit to 1. No interrupt is generated when the INTVALn register is written.

19.5.2 One-shot interrupt mode

The one-shot interrupt generates one interrupt after a one-time count. With this mode, a single interrupt can be generated at any point. This mode can be used to introduce a specific delay in a software task.

When the timer is in the idle state, writing a non-zero value IVALUE to the INTVALn register immediately loads the time interval value IVALUE - 1, and the timer starts to count down. When the timer reaches 0, an interrupt is generated and the timer stops and enters the idle state.

While the timer is running in the one-shot interrupt mode, the following actions can be performed:

- Update the INTVALn register with a new time interval value (>0) and set the LOAD bit to 1. The timer immediately reloads the new time interval, and starts counting down from the new value. No interrupt is generated when the TIME_INTVALn register is updated.
- Write a 0 to the INTVALn register and set the LOAD bit to 1. The timer immediately stops counting and moves to the idle state. No interrupt is generated when the INTVALn register is updated.

19.5.3 One-shot stall mode

One-shot stall mode is similar to one-shot interrupt mode, except that it is intended for very short delays, for instance when the delay needed is less than the time it takes to get to an interrupt service routine. This mode is designed for very low software overhead, requiring only a single write to the INTVAL register (if the channel is already configured for one-shot stall mode). The MRT times the requested delay while stalling the bus write operation, concluding the write when the delay is complete. No interrupt or status polling is needed.

Bus stall mode can be used when a short delay is need between two software controlled events, or when a delay is expected before software can continue. Since in this mode there are no bus transactions while the MRT is counting down, the CPU consumes a minimum amount of power during that time.

Note that bus stall mode provides a minimum amount of time between the execution of the instruction that performs the write to INTVAL and the time that software continues. Other system events, such as interrupts or other bus masters accessing the APB bus where the MRT resides, can cause the delay to be longer.

19.6 Register description

The reset values shown in Table 325 are POR reset values.

Table 325. Register overview: MRT (base address 0x4000 D000)

Name	Access	Offset	Description	Reset value	Section
MRT Timer 0 registers					
INTVAL0	R/W	0x0	MRT0 Time interval value. This value is loaded into TIMER0.	0x0	19.6.1
TIMER0	RO	0x4	MRT0 Timer. This register reads the value of the down-counter.	0xFF FFFF	19.6.2
CTRL0	R/W	0x8	MRT0 Control. This register controls the MRT0 modes.	0x0	19.6.3
STAT0	R/W	0xC	MRT0 Status.	0x0	19.6.4
MRT Timer 1 registers					
INTVAL1	R/W	0x10	MRT1 Time interval value. This value is loaded into TIMER1.	0x0	19.6.1
TIMER1	R/W	0x14	MRT1 Timer. This register reads the value of the down-counter.	0xFF FFFF	19.6.2
CTRL1	R/W	0x18	MRT1 Control. This register controls the MRT1 modes.	0x0	19.6.3
STAT1	R/W	0x1C	MRT1 Status.	0x0	19.6.4
MRT Timer 2 registers					
INTVAL2	R/W	0x20	MRT2 Time interval value. This value is loaded into TIMER2.	0x0	19.6.1
TIMER2	R/W	0x24	MRT2 Timer. This register reads the value of the down-counter.	0xFF FFFF	19.6.2
CTRL2	R/W	0x28	MRT2 Control. This register controls the MRT2 modes.	0x0	19.6.3
STAT2	R/W	0x2C	MRT2 Status.	0x0	19.6.4
MRT Timer 3 registers					
INTVAL3	R/W	0x30	MRT3 Time interval value. This value is loaded into TIMER3.	0x0	19.6.1
TIMER3	R/W	0x34	MRT3 Timer . This register reads the value of the down-counter.	0xFF FFFF	19.6.2
CTRL3	R/W	0x38	MRT3 Control. This register controls the MRT modes.	0x0	19.6.3
STAT3	R/W	0x3C	MRT3 Status.	0x0	19.6.4
Global MRT registers					
MODCFG	R/W	0xF0	Module Configuration. This register provides information about this particular MRT instance, and allows choosing an overall mode for the idle channel feature.	0x0	19.6.5
IDLE_CH	RO	0xF4	Idle channel. This register returns the number of the first idle channel.	0x0	19.6.6
IRQ_FLAG	R/W	0xF8	Global interrupt flag,	0x0	19.6.7

19.6.1 Time interval register

This register contains the MRT load value and controls how the timer is reloaded. The load value is IVALUE -1.

Table 326. Time interval register (INTVAL[0:3], offset 0x000 (INTVAL0) to 0x030 (INTVAL3)) bit description

Bit	Symbol	Value	Description	Reset value
23:0	IVALUE		Time interval load value. This value is loaded into the TIMERn register and the MRT channel n starts counting down from IVALUE -1. If the timer is idle, writing a non-zero value to this bit field starts the timer immediately. If the timer is running, writing a zero to this bit field does the following: • If LOAD = 1, the timer stops immediately. • If LOAD = 0, the timer stops at the end of the time interval.	0x0
30:24	-	-	Reserved. Read value is undefined, only zero should be written.	-
31	LOAD		Determines how the timer interval value IVALUE -1 is loaded into the TIMERn register. This bit is write-only. Reading this bit always returns 0.	0x0
		0	No force load. The load from the INTVALn register to the TIMERn register is processed at the end of the time interval if the repeat mode is selected.	
		1	Force load. The INTVALn interval value IVALUE -1 is immediately loaded into the TIMERn register while TIMERn is running.	

19.6.2 Timer register

The timer register holds the current timer value. This register is read-only.

Table 327. Timer register (TIMER[0:3], offset 0x004 (TIMER0) to 0x034 (TIMER3)) bit description

Bit	Symbol	Description	Reset value
23:0	VALUE	Holds the current timer value of the down-counter. The initial value of the TIMERn register is loaded as IVALUE - 1 from the INTVALn register either at the end of the time interval or immediately in the following cases: INTVALn register is updated in the idle state. INTVALn register is updated with LOAD = 1. When the timer is in idle state, reading this bit fields returns -1 (0x00FF FFFF).	0x00FF FFFF
31:24	-	Reserved. Read value is undefined, only zero should be written.	0x0

19.6.3 Control register

The control register configures the mode for each MRT and enables the interrupt.

Table 328. Control register (CTRL[0:3], offset 0x08 (CTRL0) to 0x38 (CTRL3)) bit description

Bit	Symbol	Value	Description	Reset value
0	INTEN		Enable the TIMERn interrupt.	0x0
		0	Disabled. TIMERn interrupt is disabled.	
		1	Enabled. TIMERn interrupt is enabled.	
2:1	MODE		Selects timer mode.	0x0
		0x0	Repeat interrupt mode.	
		0x1	One-shot interrupt mode.	
		0x2	One-shot stall mode.	
		0x3	Reserved.	
31:3	-	-	Reserved.	0x0

19.6.4 Status register

This register indicates the status of each MRT.

Table 329. Status register (STAT[0:3], offset 0x0C (STAT0) to 0x3C (STAT3)) bit description

Bit	Symbol	Value	Description	Reset value
0	INTFLAG		Monitors the interrupt flag.	0x0
		0	No pending interrupt. Writing a zero is equivalent to no operation.	
		1	Pending interrupt. The interrupt is pending because TIMERn has reached the end of the time interval. If the INTEN bit in the CONTROLn is also set to 1, the interrupt for timer channel n and the global interrupt are raised. Writing a 1 to this bit clears the interrupt request.	
1	RUN		Indicates the state of TIMERn. This bit is read-only.	0x0
		0	Idle state. TIMERn is stopped.	
		1	Running. TIMERn is running.	
2	INUSE		Channel In Use flag. Operating details depend on the MULTITASK bit in the MODCFG register, and affects the use of IDLE_CH. See Section 19.6.6 “Idle channel register” for details of the two operating modes.	0x0
		0	This channel **is not** in use.	
		1	This channel **is** in use. Writing a 1 to this bit clears the status.	
31:3	-	-	Reserved.	0x0

19.6.5 Module Configuration register

The MODCFG register provides the configuration (number of channels and timer width) for this MRT. See Section 19.6.6 “Idle channel register” for details.

Table 330. Module Configuration register (MODCFG, offset 0xF0) bit description

Bit	Symbol	Value	Description	Reset Value
3:0	NOC	-	Identifies the number of channels in this MRT. (4 channels on this device.)	0x3
8:4	NOB	-	Identifies the number of timer bits in this MRT. (24 bits wide on this device.)	0x17
30:9	-	-	Reserved. Read value is undefined, only zero should be written.	-
31	MULTITASK		Selects the operating mode for the INUSE flags and the IDLE_CH register.	0x0
		0	Hardware status mode. In this mode, the INUSE(n) flags for all channels are reset.	
		1	Multi-task mode.	

19.6.6 Idle channel register

The idle channel register can be used to assist software in finding available channels in the MRT. This allows more flexibility by not giving hard assignments to software that makes use of the MRT, without the need to search for an available channel. Generally, IDLE_CH returns the lowest available channel number.

IDLE_CH can be used in two ways, controlled by the value of the MULTITASK bit in the MODCFG register. MULTITASK affects both the function of IDLE_CH, and the function of the INUSE bit for each MRT channel as follows:

- MULTITASK = 0: hardware status mode. The INUSE flags for all MRT channels are reset. IDLECH returns the lowest idle channel number. A channel is considered idle if its RUN flag = 0, and there is no interrupt pending for that channel.
- MULTITASK = 1: multi-task mode. In this mode, the INUSE flags allow more control over when MRT channels are released for further use. When IDLE_CH is read, returning a channel number of an idle channel, the INUSE flag for that channel is set by hardware. That channel will not be considered idle until its RUN flag = 0, there is no interrupt pending, and its INUSE flag = 0. This allows reserving an MRT channel with a single register read, and no need to start the channel before it is no longer considered idle by IDLE_CH. It also allows software to identify a specific MRT channel that it can use, then use it more than once without releasing it, removing the need to ask for an available channel for every use.

Table 331. Idle channel register (IDLE_CH, offset 0xF4) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved.	0x0
7:4	CHAN	Idle channel. Reading the CHAN bits, returns the lowest idle timer channel. The number is positioned such that it can be used as an offset from the MRT base address in order to access the registers for the allocated channel. If all timer channels are running, CHAN = 0xF. See text above for more details.	0x0
31:8	-	Reserved.	0x0

19.6.7 Global interrupt flag register

The global interrupt register combines the interrupt flags from the individual timer channels in one register. Setting and clearing each flag behaves in the same way as setting and clearing the INTFLAG bit in each of the STATUSn registers.

Table 332. Global interrupt flag register (IRQ_FLAG, offset 0xF8) bit description

Bit	Symbol	Value	Description	Reset value
0	GFLAG0		Monitors the interrupt flag of TIMER0.	0x0
		0	No pending interrupt. Writing a zero is equivalent to no operation.	
		1	Pending interrupt. The interrupt is pending because TIMER0 has reached the end of the time interval. If the INTEN bit in the CONTROL0 register is also set to 1, the interrupt for timer channel 0 and the global interrupt are raised. Writing a 1 to this bit clears the interrupt request.	
1	GFLAG1	-	Monitors the interrupt flag of TIMER1. See description of channel 0.	0x0
2	GFLAG2	-	Monitors the interrupt flag of TIMER2. See description of channel 0.	0x0
3	GFLAG3	-	Monitors the interrupt flag of TIMER3. See description of channel 0.	0x0
31:4	-	-	Reserved. Read value is undefined, only zero should be written.	0x0

20.1 How to read this chapter

The system tick timer (SysTick timer) is present on all LPC51U68 devices.

20.2 Basic configuration

Configuration of the system tick timer is accomplished as follows:

1. Pins: The system tick timer uses no external pins.
2. Power: The system tick timer is enabled through the SysTick control register). The system tick timer clock is fixed to half the frequency of the system clock.
3. Enable the clock source for the SysTick timer in the SYST_CSR register.

20.3 Features

- Simple 24-bit timer.
- Uses dedicated exception vector.
- Clocked internally by the system clock or the SYSTICKCLK.

20.4 General description

Block diagrams of the SysTick timer for each CPU is shown in Figure 48.

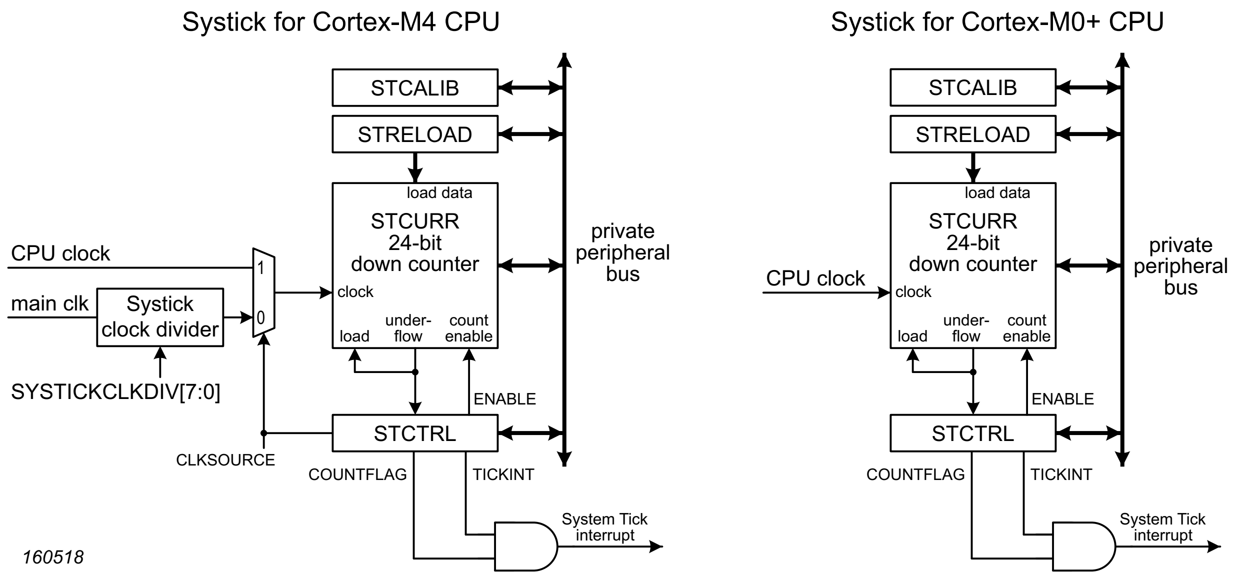


Fig 48. System tick timer block diagram

The SysTick timer is an integral part of Cortex-M0+. The SysTick timer is intended to generate a fixed 10 millisecond interrupt for use by an operating system or other system management software.

Since the SysTick timer is a part of the CPU, it facilitates porting of software by providing a standard timer that is available on ARM Cortex-based devices. The SysTick timer can be used for:

- An RTOS tick timer which fires at a programmable rate (for example 100 Hz) and invokes a SysTick routine.
- A high-speed alarm timer using the core clock.
- A simple counter. Software can use this to measure time to completion and time used.
- An internal clock source control based on missing/meeting durations. The COUNTFLAG bit-field in the control and status register can be used to determine if an action completed within a set duration, as part of a dynamic clock management control loop.

Refer to the appropriate ARM Cortex User Guide for details.

20.5 Register description

The systick timer registers are located on the private peripheral bus of each CPU (see Figure 9).

Table 333. Register overview: SysTick timer (base address 0xE000 E000)

Name	Access	Offset	Description	Reset value[1]	Section
SYST_CSR	R/W	0x010	System Timer Control and status register	0x0	20.5.1
SYST_RVR	R/W	0x014	System Timer Reload value register	0x0	20.5.2
SYST_CVR	R/W	0x018	System Timer Current value register	0x0	20.5.3
SYST_CALIB	RO	0x01C	System Timer Calibration value register	0x0	20.5.4

[1] Reset Value reflects the data stored in used bits only. It does not include content of reserved bits.

20.5.1 System Timer Control and status register

The SYST_CSR register contains control information for the SysTick timer and provides a status flag. This register is part of the CPU.

This register determines the clock source for the system tick timer.

Table 334. SysTick Timer Control and status register (SYST_CSR, offset 0x010) bit description

Bit	Symbol	Description	Reset value
0	ENABLE	System Tick counter enable. When 1, the counter is enabled. When 0, the counter is disabled.	0x0
1	TICKINT	System Tick interrupt enable. When 1, the System Tick interrupt is enabled. When 0, the System Tick interrupt is disabled. When enabled, the interrupt is generated when the System Tick counter counts down to 0.	0x0
2	CLKSOURCE	System Tick clock source selection. When 1, the system clock is selected. When 0, the output clock from the system tick clock divider (SYSTICKCLKDIV, see Figure 48 and Table 130) is selected as the reference clock. **Remark:** When the system tick clock divider is selected as the clock source, the CPU clock must be at least 2.5 times faster than the divider output.	0x0
15:3	-	Reserved. Read value is undefined, only zero should be written.	-
16	COUNTFLAG	Returns 1 if the SysTick timer counted to 0 since the last read of this register. Cleared automatically when read or when the counter is cleared.	0x0
31:17	-	Reserved. Read value is undefined, only zero should be written.	-

20.5.2 System Timer Reload value register

The SYST_RVR register is set to the value that will be loaded into the SysTick timer whenever it counts down to zero. This register is loaded by software as part of timer initialization. The SYST_CALIB register may be read and used as the value for SYST_RVR register if the CPU is running at the frequency intended for use with the SYST_CALIB value.

Table 335. System Timer Reload value register (SYST_RVR, offset 0x014) bit description

Bit	Symbol	Description	Reset value
23:0	RELOAD	This is the value that is loaded into the System Tick counter when it counts down to 0.	0x0
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

20.5.3 System Timer Current value register

The SYST_CVR register returns the current count from the System Tick counter when it is read by software.

Table 336. System Timer Current value register (SYST_CVR, offset 0x018) bit description

Bit	Symbol	Description	Reset value
23:0	CURRENT	Reading this register returns the current value of the System Tick counter. Writing any value clears the System Tick counter and the COUNTFLAG bit in SYST_CSR.	0x0
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

20.5.4 System Timer Calibration value register

The value of the SYST_CALIB register is read-only and is provided by the value of the SYSTCKCAL register in the system configuration block (see Table 101).

Table 337. System Timer Calibration value register (SYST_CALIB, offset 0x01C) bit description

Bit	Symbol	Description	Reset value
23:0	TENMS	Reload value from the SYSTCKCAL register in the SYSCON block. This field is loaded from the SYSTCKCAL register in Syscon.	0x0
29:24	-	Reserved. Read value is undefined, only zero should be written.	-
30	SKEW	Indicates whether the TENMS value will generate a precise 10 millisecond time, or an approximation. This bit is loaded from the SYSTCKCAL register in Syscon. When 0, the value of TENMS is considered to be precise. When 1, the value of TENMS is not considered to be precise.	0x0
31	NOREF	Indicates whether an external reference clock is available. This bit is loaded from the SYSTCKCAL register in Syscon. When 0, a separate reference clock is available. When 1, a separate reference clock is not available.	0x0

20.6 Functional description

The SysTick timer is a 24-bit timer that counts down to zero and generates an interrupt. The intent is to provide a fixed 10 millisecond time interval between interrupts. The SysTick timer is clocked from the CPU clock (the system clock, see Figure 9) or from the reference clock, which is fixed to half the frequency of the CPU clock. In order to generate recurring interrupts at a specific interval, the SYST_RVR register must be initialized with the correct value for the desired interval.

20.7 Example timer calculations

To use the system tick timer, do the following:

1. Program the SYST_RVR register with the reload value calculated as shown below to obtain the desired time interval.
2. Clear the SYST_CVR register by writing to it. This ensures that the timer will count from the SYST_RVR value rather than an arbitrary value when the timer is enabled.

The following examples illustrate selecting SysTick timer reload values for different system configurations. All of the examples calculate an interrupt interval of 10 milliseconds, as the SysTick timer is intended to be used, and there are no rounding errors.

System clock = 72 MHz

Program the SYST_CSR register with the value 0x7 which selects the system clock as the clock source and enables the SysTick timer and the SysTick timer interrupt.

SYST_RVR = (system clock frequency × 10 ms) −1 = (72 MHz × 10 ms) −1 = 720000 −1 = 719999 = 0x000A FC7F

System tick timer clock = 24 MHz

Program the SYST_CSR register with the value 0x3 which selects the clock from the system tick clock divider (see Section 6.5.31 “SYSTICK clock divider register”) as the clock source and enables the SysTick timer and the SysTick timer interrupt. Use DIV = 3.

SYST_RVR = (system tick timer clock frequency × 10 ms) –1 = (24 MHz × 10 ms) –1 = 240000 –1 = 239999 = 0x0003 A97F

System clock = 12 MHz

Program the SYST_CSR register with the value 0x7 which selects the system clock as the clock source and enables the SysTick timer and the SysTick timer interrupt.

In this case the system clock is derived from the FRO 12 MHz clock.

SYST_RVR = (system clock frequency × 10 ms) –1 = (12 MHz × 10 ms) –1 = 120000 –1 = 119999 = 0x0001 D4BF

UM11071

Chapter 21: LPC51U68 Micro-tick Timer (UTICK)

Rev. 1.1 — 17 May 2018 **User manual**

21.1 How to read this chapter

The Micro-Tick Timer is available on all LPC51U68 devices.

21.2 Features

- Ultra simple, ultra-low power timer that can run and wake up the device in reduced power modes other than deep power-down.
- Write once to start.
- Interrupt or software polling.
- Four capture registers that can be triggered by external pin transitions.

21.3 Basic configuration

Configure the Micro-Tick Timer as follows:

- Set the UTICK bit in the AHBCLKCTRL1 register (Table 116) to enable the clock to the Micro-Tick Timer register interface.
- The Micro-Tick Timer provides an interrupt to the NVIC, see Table 79.
- To enable Micro-Tick timer interrupts for waking up from deep-sleep mode, enable the interrupts in the STARTER0 register (Table 155) and the NVIC.
- Configure the pin functions of any Micro-tick Timer capture pins that will be used via IOCON, see Chapter 9.
- Enable the Watchdog oscillator that provides the Micro-Tick Timer clock in the syscon block by clearing the appropriate bit in PDRUNCFG0. See Section 6.5.48.

21.4 General description

Figure 49 shows a conceptual view of the Micro-tick Timer.

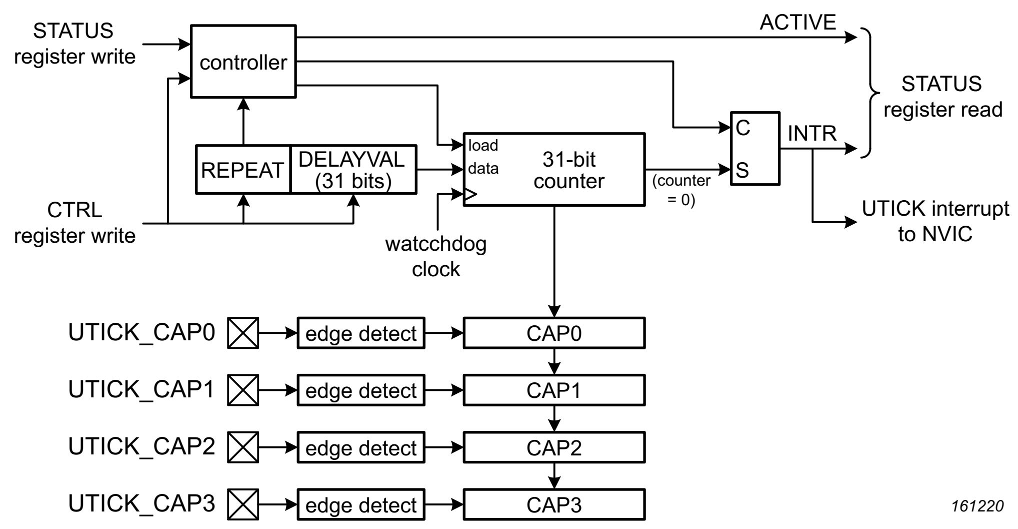


Fig 49. MIcro-Tick Timer block diagram

21.5 Pin description

Table 338 gives a summary of pins related to the Micro-tick Timer.

Table 338. Micro-tick Timer pin description

Pin	Type	Description
UTICK_CAP0, UTICK_CAP1, UTICK_CAP2, UTICK_CAP3	Input	Capture inputs. The selected transition on a capture pin can be configured to load the related CAP register with the value of counter.

21.6 Register description

The Micro-Tick Timer contains the registers shown in Table 339. Note that the Micro-Tick Timer operates from a different (typically slower) clock than the CPU and bus systems. This means there may be a synchronization delay when accessing Micro-Tick Timer registers.

Table 339. Register overview: Micro-Tick Timer (base address 0x4000 E000)

Name	Access	Offset	Description	Reset value	Section
CTRL	R/W	0x000	Control register.	0x0	21.6.1
STAT	R/W	0x004	Status register.	0x0	21.6.2
CFG	R/W	0x008	Capture configuration register.	0x0	21.6.3
CAPCLR	WO	0x00C	Capture clear register.	-	21.6.4
CAP0	RO	0x010	Capture register 0.	0x0	21.6.5
CAP1	RO	0x014	Capture register 1.	0x0	21.6.5
CAP2	RO	0x018	Capture register 2.	0x0	21.6.5
CAP3	RO	0x01C	Capture register 3.	0x0	21.6.5

21.6.1 CTRL register

This register controls the Micro-tick Timer. Any write to the CTRL register resets the counter, meaning a new interval will be measured if one was in progress.

Table 340. Control register (CTRL, offset 0x000) bit description

Bit	Symbol	Description	Reset value
30:0	DELAYVAL	Tick interval value. The delay will be equal to DELAYVAL + 1 periods of the timer clock. The minimum usable value is 1, for a delay of 2 timer clocks. A value of 0 stops the timer.	0x0
31	REPEAT	Repeat delay. 0 = One-time delay. 1 = Delay repeats continuously.	0x0

21.6.2 Status register

This register provides status for the Micro-tick Timer.

Table 341. Status register (STAT, offset 0x004) bit description

Bit	Symbol	Description	Reset value
0	INTR	Interrupt flag. 0 = No interrupt is pending. 1 = An interrupt is pending. A write of any value to this register clears this flag.	0x0
1	ACTIVE	Active flag. 0 = The Micro-Tick Timer is stopped. 1 = The Micro-Tick Timer is currently active.	0x0
31:2	-	Reserved	-

21.6.3 Capture configuration register

This register allows enabling Micro-tick capture functions and selects the polarity of the capture triggers.

Table 342. Capture configuration register (CFG, offset 0x008) bit description

Bit	Symbol	Description	Reset value
0	CAPEN0	Enable Capture 0. 1 = Enabled, 0 = Disabled.	0x0
1	CAPEN1	Enable Capture 1. 1 = Enabled, 0 = Disabled.	0x0
2	CAPEN2	Enable Capture 2. 1 = Enabled, 0 = Disabled.	0x0
3	CAPEN3	Enable Capture 3. 1 = Enabled, 0 = Disabled.	0x0
7:4	-	Reserved	-
8	CAPPOL0	Capture Polarity 0. 0 = Positive edge capture, 1 = Negative edge capture.	0x0
9	CAPPOL1	Capture Polarity 1. 0 = Positive edge capture, 1 = Negative edge capture.	0x0
10	CAPPOL2	Capture Polarity 2. 0 = Positive edge capture, 1 = Negative edge capture.	0x0
11	CAPPOL3	Capture Polarity 3. 0 = Positive edge capture, 1 = Negative edge capture.	0x0
31:12	-	Reserved	-

21.6.4 Capture clear register

This read-only register allows clearing previous capture values, allowing new captures to take place.

Table 343. Capture clear register (CAPCLR, offset 0x00C) bit description

Bit	Symbol	Description	Reset value
0	CAPCLR0	Clear capture 0. Writing 1 to this bit clears the CAP0 register value.	-
1	CAPCLR1	Clear capture 1. Writing 1 to this bit clears the CAP1 register value.	-
2	CAPCLR2	Clear capture 2. Writing 1 to this bit clears the CAP2 register value.	-
3	CAPCLR3	Clear capture 3. Writing 1 to this bit clears the CAP3 register value.	-
31:4	-	Reserved	-

21.6.5 Capture registers

This register contains the Micro-tick time value based on any previously capture events.Each Capture register is associated with one of the capture trigger inputs.

Table 344. Capture registers (CAP[0:3], offsets [0x010:0x01C]) bit description

Bit	Symbol	Description	Reset value
30:0	CAP_VALUE	Capture value for the related capture event (UTICK_CAPn. Note: the value is 1 lower than the actual value of the Micro-tick Timer at the moment of the capture event.	0x0
31	VALID	Capture Valid. When 1, a value has been captured based on a transition of the related UTICK_CAPn pin. Cleared by writing to the related bit in the CAPCLR register.	0x0

UM11071

Chapter 22: LPC51U68 USB 2.0 device controller

Rev. 1.1 — 17 May 2018 User manual

22.1 How to read this chapter

The USB block is available on selected LPC51U68 devices.

22.2 Basic configuration

Initial configuration of the USB is accomplished as follows:

- Pins: Configure the USB pins in the IOCON register block.
- In the AHBCLKCTRL1 register, enable the clock to the USB controller register interface (see Section 6.5.17).
- Power: Enable the power to the USB PHY in the PDRUNCFG0 register (Section 6.5.48).
- Configure the USB main clock (see Section 6.5.27 and Section 6.5.35).
- Configure the USB wake-up signal (see Section 22.7.6) if needed.

22.3 Features

- USB2.0 full-speed device controller.
- Supports 10 physical (5 logical) endpoints including two control endpoints if physical, one control endpoint if logical.
- Single and double-buffering supported.
- Each non-control endpoint supports bulk, interrupt, or isochronous endpoint types.
- Supports wake-up from deep-sleep mode on USB activity and remote wake-up.
- Supports SoftConnect internally.
- Link Power Management (LPM) supported.

22.4 General description

The Universal Serial Bus (USB) is a four-wire bus that supports communication between a host and one or more (up to 127) peripherals. The host controller allocates the USB bandwidth to attached devices through a token-based protocol. The bus supports hot plugging and dynamic configuration of the devices. All transactions are initiated by the host controller.

The host schedules transactions in 1 ms frames. Each frame contains a Start-Of-Frame (SOF) marker and transactions that transfer data to or from device endpoints. Each device can have a maximum of 16 logical or 32 physical endpoints. The device controller supports up to 10 physical endpoints. There are four types of transfers defined for the endpoints. Control transfers are used to configure the device.

Interrupt transfers are used for periodic data transfer. Bulk transfers are used when the latency of transfer is not critical. Isochronous transfers have guaranteed delivery time but no error correction.

For more information on the Universal Serial Bus, see the USB Implementers Forum website.

The USB device controller enables full-speed (12 Mb/s) data exchange with a USB host controller.

Figure 50 shows the block diagram of the USB device controller.

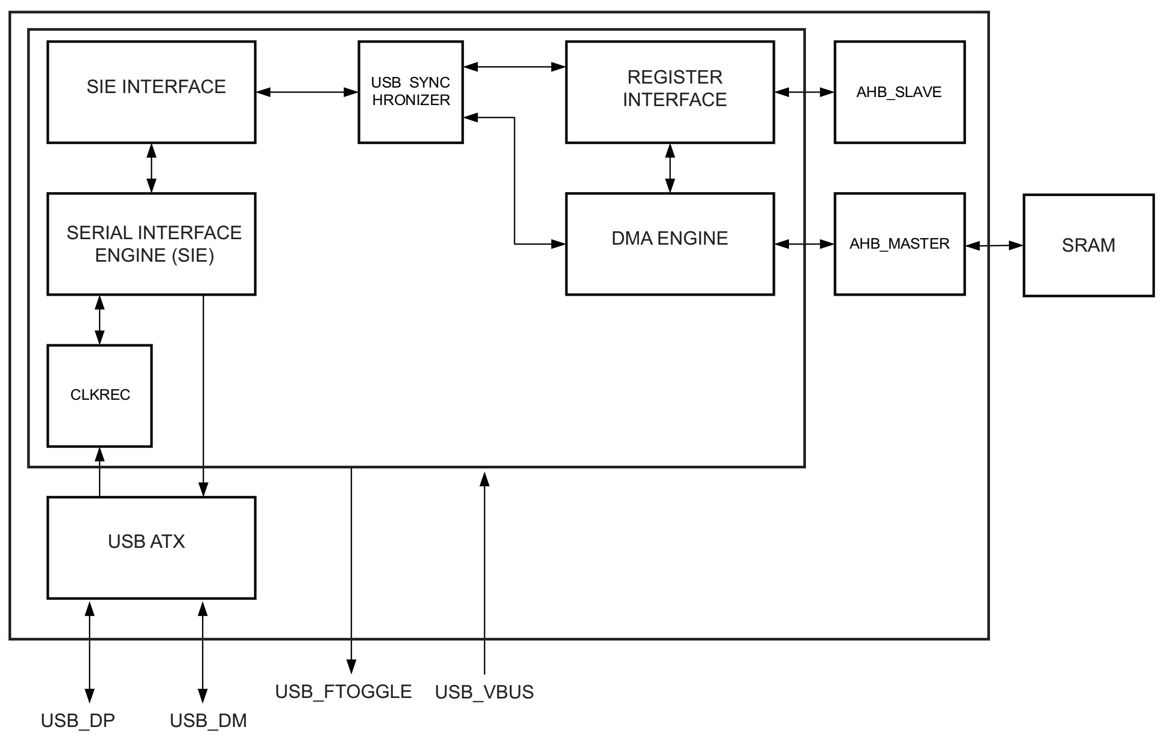


Fig 50. USB block diagram

The USB Device Controller has a built-in analog transceiver (ATX). The USB ATX sends/receives the bi-directional USB0_DP and USB0_DM signals of the USB bus.

The SIE implements the full USB protocol layer. It is completely hard-wired for speed and needs no software intervention. It handles transfer of data between the endpoint buffers in USB RAM and the USB bus. The functions of this block include: synchronization pattern recognition, parallel/serial conversion, bit stuffing/de-stuffing, CRC checking/generation, PID verification/generation, address recognition, and handshake evaluation/generation.

22.4.1 USB software interface

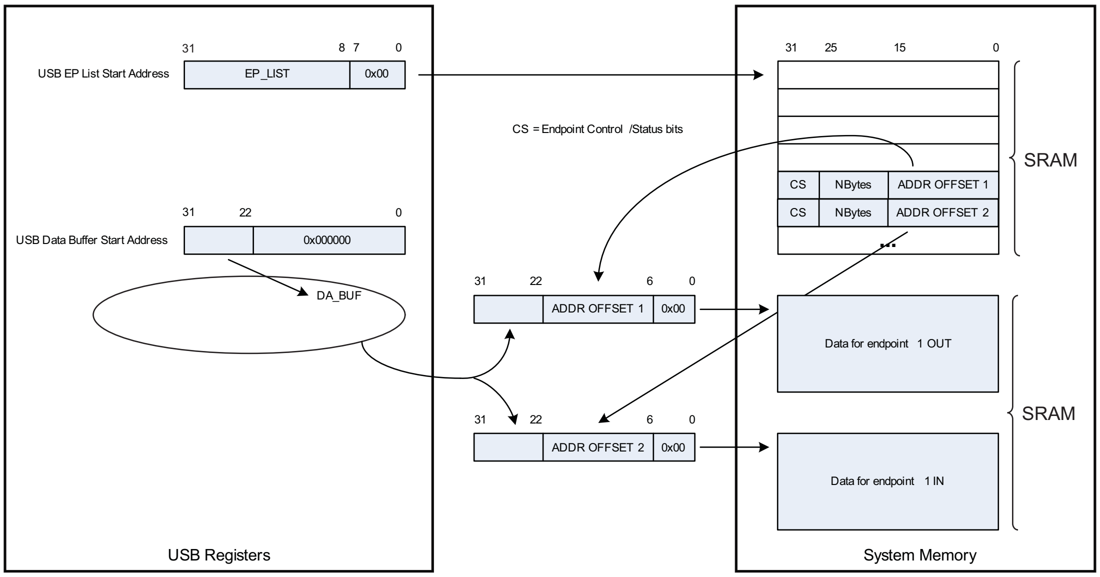


Fig 51. USB software interface

22.4.2 Fixed endpoint configuration

Table 345 shows the supported endpoint configurations. The packet size is configurable up to the maximum value shown in Table 345 for each type of end point.

Table 345. Fixed endpoint configuration

Logical endpoint	Physical endpoint	Endpoint type	Direction	Max packet size (byte)	Double buffer
0	0	Control	Out	64	No
0	1	Control	In	64	No
1	2	Interrupt/Bulk/Isochronous	Out	64/64/1023	Yes
1	3	Interrupt/Bulk/Isochronous	In	64/64/1023	Yes
2	4	Interrupt/Bulk/Isochronous	Out	64/64/1023	Yes
2	5	Interrupt/Bulk/Isochronous	In	64/64/1023	Yes
3	6	Interrupt/Bulk/Isochronous	Out	64/64/1023	Yes
3	7	Interrupt/Bulk/Isochronous	In	64/64/1023	Yes
4	8	Interrupt/Bulk/Isochronous	Out	64/64/1023	Yes
4	9	Interrupt/Bulk/Isochronous	In	64/64/1023	Yes

22.4.3 SoftConnect

The softConnect signal is implemented internally. An external pull-up resistor between USB_DP and VDD is not necessary.

Software can control the pull-up by setting the DCON bit in the DEVCMDSTAT register. If the DCON bit is set to 1, the USB_DP line is pulled up to VDD through an internal 1.5 KOhm pull-up resistor.

22.4.4 Interrupts

The USB controller has two interrupt lines, a general USB interrupt and a USB activity wake-up interrupt (USB_NEEDCLK). An general interrupt is generated by the hardware if both the interrupt status bit and the corresponding interrupt enable bit are set. The interrupt status bit is set by hardware if the interrupt condition occurs (irrespective of the interrupt enable bit setting).

22.4.5 Suspend and resume

The USB protocol insists on power management by the USB device. This becomes even more important if the device draws power from the bus (bus-powered device). The following constraints should be met by the bus-powered device.

- A device in the non-configured state should draw a maximum of 100mA from the USB bus.
- A configured device can draw only up to what is specified in the Max Power field of the configuration descriptor. The maximum value is 500 mA.
- A suspended device should draw a maximum of 500 μA.

A device will go into the L2 suspend state if there is no activity on the USB bus for more than 3 ms. A suspended device wakes up, if there is transmission from the host (host-initiated wake up). The USB controller also supports software initiated remote wake-up. To initiate remote wake-up, software on the device must enable all clocks and clear the suspend bit. This will cause the hardware to generate a remote wake-up signal upstream.

The USB controller supports Link Power Management (LPM). Link Power Management defines an additional link power management state L1 that supplements the existing L2 state by utilizing most of the existing suspend/resume infrastructure but provides much faster transitional latencies between L1 and L0 (On).

The assertion of USB suspend signal indicates that there was no activity on the USB bus for the last 3 ms. At this time an interrupt is sent to the processor on which the software can start preparing the device for suspend.

If there is no activity for the next 2 ms, the USB NEEDCLK signal will go low. This indicates that the USB main clock can be switched off.

When activity is detected on the USB bus, the USB suspend signal is deactivated and USB NEEDCLK signal is activated. This process is fully combinatorial and hence no USB main clock is required to activate the USB NEEDCLK signal.

22.4.6 Frame toggle output

The USB0_FRAME output pin reflects the 1 kHz clock derived from the incoming Start of Frame tokens sent by the USB host.

22.4.7 Clocking

The USB device controller has the following clock connections:

- USB main clock: The USB main clock is a 48 MHz clock used for USB functions (see Section 6.5.27 and Section 6.5.35). If the FRO is used as the USB clock source, it can be configured to adjust automatically to the USB bus rate (see Section 6.5.43).
- AHB clock: This is the AHB system bus clock. The minimum frequency of the AHB clock is 6 MHz when the USB device controller is receiving or transmitting USB packets.

22.5 Pin description

The device controller can access one USB port.

Table 346. USB device pin description

Name	Direction	Description
USB0_VBUS	I	VBUS status input. When this function is not enabled via its corresponding IOCON register, it is driven HIGH internally.
USB0_FRAME	O	USB 1 ms SoF (Start of Frame) signal.
USB0_DP	I/O	Positive differential data. If the device is put into deep power-down mode, this pin should not be allowed to float, i.e. externally pulled up or down.
USB0_DM	I/O	Negative differential data. If the device is put into deep power-down mode, this pin should not be allowed to float, i.e. externally pulled up or down.
USB0_LEDN	O	Configured LED indicator (active low).

22.6 Register description

Table 347. Register overview: USB (base address 0x4008 4000)

Name	Access	Offset	Description	Reset value	Section
DEVCMDSTAT	R/W	0x000	USB Device Command/Status register	0x800	22.6.1
INFO	R/W	0x004	USB Info register	0x0	22.6.2
EPLISTSTART	R/W	0x008	USB EP Command/Status List start address	0x0	22.6.3
DATABUFSTART	R/W	0x00C	USB Data buffer start address	0x0	22.6.4
LPM	R/W	0x010	USB Link Power Management register	0x0	22.6.5
EPSKIP	R/W	0x014	USB Endpoint skip	0x0	22.6.6
EPINUSE	R/W	0x018	USB Endpoint Buffer in use	0x0	22.6.7
EPBUFCFG	R/W	0x01C	USB Endpoint Buffer Configuration register	0x0	22.6.8
INTSTAT	R/W	0x020	USB interrupt status register	0x0	22.6.9
INTEN	R/W	0x024	USB interrupt enable register	0x0	22.6.10
INTSETSTAT	R/W	0x028	USB set interrupt status register	0x0	22.6.11
EPTOGGLE	RO	0x034	USB Endpoint toggle register	0x0	22.6.12

22.6.1 USB Device Command/Status register

Table 348. USB Device Command/Status register (DEVCMDSTAT, offset 0x00) bit description

Bit	Symbol	Value	Description	Reset value	Access
6:0	DEV_ADDR	-	USB device address. After bus reset, the address is reset to 0x00. If the enable bit is set, the device will respond on packets for function address DEV_ADDR. When receiving a SetAddress Control Request from the USB host, software must program the new address before completing the status phase of the SetAddress Control Request.	0x0	R/W
7	DEV_EN	-	USB device enable. If this bit is set, the HW will start responding on packets for function address DEV_ADDR.	0x0	R/W
8	SETUP	-	SETUP token received. If a SETUP token is received and acknowledged by the device, this bit is set. As long as this bit is set all received IN and OUT tokens will be NAKed by HW. SW must clear this bit by writing a one. If this bit is zero, HW will handle the tokens to the CTRL EP0 as indicated by the CTRL EP0 IN and OUT data information programmed by SW.	0x0	R/W1C
9	FORCE_ NEEDCLK		Forces the NEEDCLK output to always be on:	0x0	R/W
		0	USB_NEEDCLK has normal function.		
		1	USB_NEEDCLK always 1. Clock will not be stopped in case of suspend.		
10	-	-	Reserved.	0x0	RO
11	LPM_SUP		LPM Supported:	0x1	R/W
		0	LPM not supported.		
		1	LPM supported.		
12	INTONNAK_AO		Interrupt on NAK for interrupt and bulk OUT EP	0x0	R/W
		0	Only acknowledged packets generate an interrupt		
		1	Both acknowledged and NAKed packets generate interrupts.		
13	INTONNAK_AI		Interrupt on NAK for interrupt and bulk IN EP	0x0	R/W
		0	Only acknowledged packets generate an interrupt		
		1	Both acknowledged and NAKed packets generate interrupts.		
14	INTONNAK_CO		Interrupt on NAK for control OUT EP	0x0	R/W
		0	Only acknowledged packets generate an interrupt		
		1	Both acknowledged and NAKed packets generate interrupts.		
15	INTONNAK_CI		Interrupt on NAK for control IN EP	0x0	R/W
		0	Only acknowledged packets generate an interrupt		
		1	Both acknowledged and NAKed packets generate interrupts.		
16	DCON	-	Device status - connect. The connect bit must be set by SW to indicate that the device must signal a connect. The pull-up resistor on USB0_DP will be enabled when this bit is set and the VBUSDEBOUNCED bit is one.	0x0	R/W

Table 348. USB Device Command/Status register (DEVCMDSTAT, offset 0x00) bit description

Bit	Symbol	Value	Description	Reset value	Access
17	DSUS	-	Device status - suspend. The suspend bit indicates the current suspend state. It is set to 1 when the device hasn't seen any activity on its upstream port for more than 3 milliseconds. It is reset to 0 on any activity. When the device is suspended (Suspend bit DSUS = 1) and the software writes a 0 to it, the device will generate a remote wake-up. This will only happen when the device is connected (Connect bit = 1). When the device is not connected or not suspended, a writing a 0 has no effect. Writing a 1 never has an effect.	0x0	R/W
18	-	-	Reserved.	0x0	RO
19	LPM_SUS	-	Device status - LPM Suspend. This bit represents the current LPM suspend state. It is set to 1 by HW when the device has acknowledged the LPM request from the USB host and the Token Retry Time of 10 μs has elapsed. When the device is in the LPM suspended state (LPM suspend bit = 1) and the software writes a zero to this bit, the device will generate a remote walk-up. Software can only write a zero to this bit when the LPM_REWP bit is set to 1. HW resets this bit when it receives a host initiated resume. HW only updates the LPM_SUS bit when the LPM_SUPP bit is equal to one.	0x0	R/W
20	LPM_REWP	-	LPM Remote Wake-up Enabled by USB host. HW sets this bit to one when the bRemoteWake bit in the LPM extended token is set to 1. HW will reset this bit to 0 when it receives the host initiated LPM resume, when a remote wake-up is sent by the device or when a USB bus reset is received. Software can use this bit to check if the remote wake-up feature is enabled by the host for the LPM transaction.	0x0	RO
23:21	-	-	Reserved.	0x0	RO
24	DCON_C	-	Device status - connect change. The Connect Change bit is set when the device's pull-up resistor is disconnected because VBus disappeared. The bit is reset by writing a one to it.	0x0	R/W1C
25	DSUS_C	-	Device status - suspend change. The suspend change bit is set to 1 when the suspend bit toggles. The suspend bit can toggle because: - The device goes in the suspended state - The device is disconnected - The device receives resume signaling on its upstream port. The bit is reset by writing a one to it.	0x0	R/W1C
26	DRES_C	-	Device status - reset change. This bit is set when the device received a bus reset. On a bus reset the device will automatically go to the default state (unconfigured and responding to address 0). The bit is reset by writing a one to it.	0x0	R/W1C
27	-	-	Reserved.	0x0	RO
28	VBUSDEBOUNCED	-	This bit indicates if Vbus is detected or not. The bit raises immediately when Vbus becomes high. It drops to zero if Vbus is low for at least 3 ms. If this bit is high and the DCon bit is set, the HW will enable the pull-up resistor to signal a connect.	0x0	RO
31:29	-	-	Reserved.	0x0	RO

22.6.2 USB Info register

Table 349. USB Info register (INFO, offset 0x04) bit description

Bit	Symbol	Value	Description	Reset value	Access
10:0	FRAME_NR	-	Frame number. This contains the frame number of the last successfully received SOF. In case no SOF was received by the device at the beginning of a frame, the frame number returned is that of the last successfully received SOF. In case the SOF frame number contained a CRC error, the frame number returned will be the corrupted frame number as received by the device.	0x0	RO
14:11	ERR_CODE		The error code which last occurred:	0x0	R/W
		0x0	No error		
		0x1	PID encoding error		
		0x2	PID unknown		
		0x3	Packet unexpected		
		0x4	Token CRC error		
		0x5	Data CRC error		
		0x6	Time out		
		0x7	Babble		
		0x8	Truncated EOP		
		0x9	Sent/Received NAK		
		0xA	Sent Stall		
		0xB	Overrun		
		0xC	Sent empty packet		
		0xD	Bitstuff error		
		0xE	Sync error		
		0xF	Wrong data toggle		
15	-	-	Reserved.	0x0	RO
31:16	-	-	Reserved	-	RO

22.6.3 USB EP Command/Status List start address

This 32-bit register indicates the start address of the USB EP Command/Status List.

Only a subset of these bits is programmable by software. The 8 least-significant bits are hardcoded to zero because the list must start on a 256 byte boundary. Bits 31 to 8 can be programmed by software.

Table 350. USB EP Command/Status List start address (EPLISTSTART, offset 0x08) bit description

Bit	Symbol	Description	Reset value	Access
7:0	-	Reserved	0x0	RO
31:8	EP_LIST	Start address of the USB EP Command/Status List.	0x0	R/W

22.6.4 USB Data buffer start address

This register indicates the page of the AHB address where the endpoint data can be located.

Table 351. USB Data buffer start address (DATABUFSTART, offset 0x0C) bit description

Bit	Symbol	Description	Reset value	Access
21:0	-	Reserved	0x0	R
31:22	DA_BUF	Start address of the buffer pointer page where all endpoint data buffers are located.	0x0	R/W

22.6.5 USB Link Power Management register

Table 352. Link Power Management register (LPM, offset 0x10) bit description

Bit	Symbol	Description	Reset value	Access
3:0	HIRD_HW	Host Initiated Resume Duration - HW. This is the HIRD value from the last received LPM token	0x0	RO
7:4	HIRD_SW	Host Initiated Resume Duration - SW. This is the time duration required by the USB device system to come out of LPM initiated suspend after receiving the host initiated LPM resume.	0x0	R/W
8	DATA_PENDING	As long as this bit is set to one and LPM supported bit is set to one, HW will return a NYET handshake on every LPM token it receives. If LPM supported bit is set to one and this bit is zero, HW will return an ACK handshake on every LPM token it receives. If SW has still data pending and LPM is supported, it must set this bit to 1.	0x0	R/W
31:9	RESERVED	Reserved	0x0	RO

22.6.6 USB Endpoint skip

Table 353. USB Endpoint skip (EPSKIP, offset 0x14) bit description

Bit	Symbol	Description	Reset value	Access
29:0	SKIP	Endpoint skip: Writing 1 to one of these bits, will indicate to HW that it must deactivate the buffer assigned to this endpoint and return control back to software. When HW has deactivated the endpoint, it will clear this bit, but it will not modify the EPINUSE bit. An interrupt will be generated when the Active bit goes from 1 to 0. Note: In case of double-buffering, HW will only clear the Active bit of the buffer indicated by the EPINUSE bit.	0x0	R/W
31:30	-	Reserved	0x0	RO

22.6.7 USB Endpoint Buffer in use

Table 354. USB Endpoint Buffer in use (EPINUSE, offset 0x18) bit description

Bit	Symbol	Description	Reset value	Access
1:0	-	Reserved. Fixed to zero because the control endpoint zero is fixed to single-buffering for each physical endpoint.	0x0	RO
9:2	BUF	Buffer in use: This register has one bit per physical endpoint. 0: HW is accessing buffer 0. 1: HW is accessing buffer 1.	0x0	R/W
31:10	-	Reserved	0x0	RO

22.6.8 USB Endpoint Buffer Configuration

Table 355. USB Endpoint Buffer Configuration (EPBUFCFG, offset 0x1C) bit description

Bit	Symbol	Description	Reset value	Access
1:0	-	Reserved. Fixed to zero because the control endpoint zero is fixed to single-buffering for each physical endpoint.	0x0	RO
9:2	BUF_SB	Buffer usage: This register has one bit per physical endpoint. 0: Single-buffer. 1: Double-buffer. If the bit is set to single-buffer (0), it will not toggle the corresponding EPINUSE bit when it clears the active bit. If the bit is set to double-buffer (1), HW will toggle the EPINUSE bit when it clears the Active bit for the buffer.	0x0	R/W
31:10	-	Reserved	0x0	RO

22.6.9 USB interrupt status register

Table 356. USB interrupt status register (INTSTAT, offset 0x20) bit description

Bit	Symbol	Description	Reset value	Access
0	EP0OUT	Interrupt status register bit for the Control EP0 OUT direction. This bit will be set if NBytes transitions to zero or the skip bit is set by software or a SETUP packet is successfully received for the control EP0. If the IntOnNAK_CO is set, this bit will also be set when a NAK is transmitted for the Control EP0 OUT direction. Software can clear this bit by writing a one to it.	0x0	R/W1C
1	EP0IN	Interrupt status register bit for the Control EP0 IN direction. This bit will be set if NBytes transitions to zero or the skip bit is set by software. If the IntOnNAK_CI is set, this bit will also be set when a NAK is transmitted for the Control EP0 IN direction. Software can clear this bit by writing a one to it.	0x0	R/W1C
2	EP1OUT	Interrupt status register bit for the EP1 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the IntOnNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP1 OUT direction. Software can clear this bit by writing a one to it.	0x0	R/W1C
3	EP1IN	Interrupt status register bit for the EP1 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the IntOnNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP1 IN direction. Software can clear this bit by writing a one to it.	0x0	R/W1C
4	EP2OUT	Interrupt status register bit for the EP2 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the IntOnNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP2 OUT direction. Software can clear this bit by writing a one to it.	0x0	R/W1C
5	EP2IN	Interrupt status register bit for the EP2 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the IntOnNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP2 IN direction. Software can clear this bit by writing a one to it.	0x0	R/W1C

Table 356. USB interrupt status register (INTSTAT, offset 0x20) bit description

Bit	Symbol	Description	Reset value	Access
6	EP3OUT	Interrupt status register bit for the EP3 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the IntOnNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP3 OUT direction. Software can clear this bit by writing a one to it.	0x0	R/W1C
7	EP3IN	Interrupt status register bit for the EP3 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the IntOnNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP3 IN direction. Software can clear this bit by writing a one to it.	0x0	R/W1C
8	EP4OUT	Interrupt status register bit for the EP4 OUT direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the IntOnNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP4 OUT direction. Software can clear this bit by writing a one to it.	0x0	R/W1C
9	EP4IN	Interrupt status register bit for the EP4 IN direction. This bit will be set if the corresponding Active bit is cleared by HW. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the IntOnNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP4 IN direction. Software can clear this bit by writing a one to it.	0x0	R/W1C
29:10	-	Reserved	0x0	RO
30	FRAME_INT	Frame interrupt. This bit is set to one every millisecond when the VbusDebounced bit and the DCON bit are set. This bit can be used by software when handling isochronous endpoints. Software can clear this bit by writing a one to it.	0x0	R/W1C
31	DEV_INT	Device status interrupt. This bit is set by HW when one of the bits in the Device Status Change register are set. Software can clear this bit by writing a one to it.	0x0	R/W1C

22.6.10 USB interrupt enable register

Table 357. USB interrupt enable register (INTEN, offset 0x24) bit description

Bit	Symbol	Description	Reset value	Access
9:0	EP_INT_EN	If this bit is set and the corresponding USB interrupt status bit is set, a HW interrupt is generated on the interrupt line indicated by the corresponding USB interrupt routing bit.	0x0	R/W
29:10	-	Reserved	0x0	RO
30	FRAME_INT_EN	If this bit is set and the corresponding USB interrupt status bit is set, a HW interrupt is generated on the interrupt line indicated by the corresponding USB interrupt routing bit.	0x0	R/W
31	DEV_INT_EN	If this bit is set and the corresponding USB interrupt status bit is set, a HW interrupt is generated on the interrupt line indicated by the corresponding USB interrupt routing bit.	0x0	R/W

22.6.11 USB set interrupt status register

Table 358. USB set interrupt status register (INTSETSTAT, offset 0x28) bit description

Bit	Symbol	Description	Reset value	Access
9:0	EP_SET_INT	If software writes a one to one of these bits, the corresponding USB interrupt status bit is set. When this register is read, the same value as the USB interrupt status register is returned.	0x0	R/W
29:10	-	Reserved	0x0	RO
30	FRAME_SET_INT	If software writes a one to one of these bits, the corresponding USB interrupt status bit is set. When this register is read, the same value as the USB interrupt status register is returned.	0x0	R/W
31	DEV_SET_INT	If software writes a one to one of these bits, the corresponding USB interrupt status bit is set. When this register is read, the same value as the USB interrupt status register is returned.	0x0	R/W

22.6.12 USB Endpoint toggle

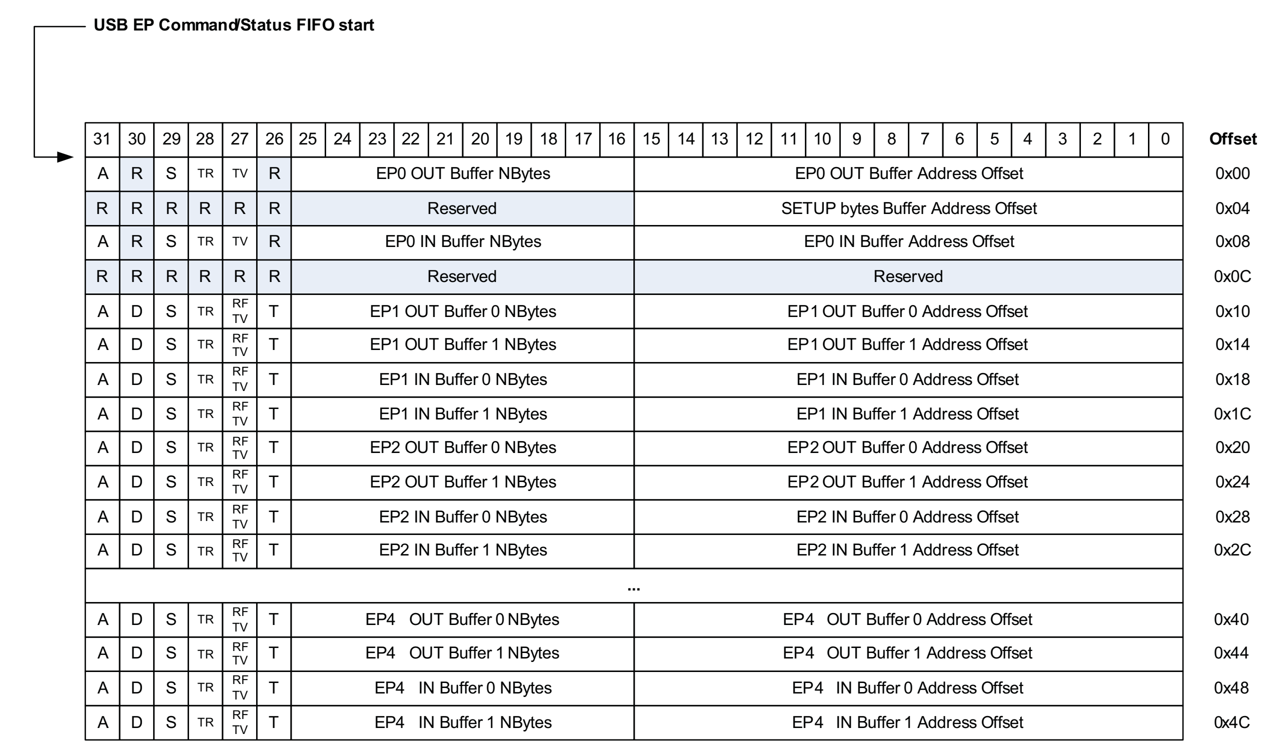
Table 359. USB Endpoint toggle (EPTOGGLE, offset 0x34) bit description

Bit	Symbol	Description	Reset value	Access
9:0	TOGGLE	Endpoint data toggle: This field indicates the current value of the data toggle for the corresponding endpoint.	0x0	RO
31:10	-	Reserved	0x0	RO

22.7 Functional description

22.7.1 Endpoint command/status list

Figure 52 gives an overview on how the Endpoint List is organized in memory. The USB EP Command/Status List start register points to the start of the list that contains all the endpoint information in memory. The order of the endpoints is fixed as shown in the picture.


USB EP Command/Status FIFO start

31	30	29	28	27	26	25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	Offset
A	R	S	TR	TV	R	EP0 OUT Buffer NBytes	EP0 OUT Buffer Address Offset	0x00
R	R	R	R	R	R	Reserved	SETUP bytes Buffer Address Offset	0x04
A	R	S	TR	TV	R	EP0 IN Buffer NBytes	EP0 IN Buffer Address Offset	0x08
R	R	R	R	R	R	Reserved	Reserved	0x0C
A	D	S	TR	RF TV	T	EP1 OUT Buffer 0 NBytes	EP1 OUT Buffer 0 Address Offset	0x10
A	D	S	TR	RF TV	T	EP1 OUT Buffer 1 NBytes	EP1 OUT Buffer 1 Address Offset	0x14
A	D	S	TR	RF TV	T	EP1 IN Buffer 0 NBytes	EP1 IN Buffer 0 Address Offset	0x18
A	D	S	TR	RF TV	T	EP1 IN Buffer 1 NBytes	EP1 IN Buffer 1 Address Offset	0x1C
A	D	S	TR	RF TV	T	EP2 OUT Buffer 0 NBytes	EP2 OUT Buffer 0 Address Offset	0x20
A	D	S	TR	RF TV	T	EP2 OUT Buffer 1 NBytes	EP2 OUT Buffer 1 Address Offset	0x24
A	D	S	TR	RF TV	T	EP2 IN Buffer 0 NBytes	EP2 IN Buffer 0 Address Offset	0x28
A	D	S	TR	RF TV	T	EP2 IN Buffer 1 NBytes	EP2 IN Buffer 1 Address Offset	0x2C
...								
A	D	S	TR	RF TV	T	EP4 OUT Buffer 0 NBytes	EP4 OUT Buffer 0 Address Offset	0x40
A	D	S	TR	RF TV	T	EP4 OUT Buffer 1 NBytes	EP4 OUT Buffer 1 Address Offset	0x44
A	D	S	TR	RF TV	T	EP4 IN Buffer 0 NBytes	EP4 IN Buffer 0 Address Offset	0x48
A	D	S	TR	RF TV	T	EP4 IN Buffer 1 NBytes	EP4 IN Buffer 1 Address Offset	0x4C

Fig 52. Endpoint command/status list (see also Table 360)

Table 360. Endpoint commands

Symbol	Access	Description
A	R/W	Active The buffer is enabled. HW can use the buffer to store received OUT data or to transmit data on the IN endpoint. Software can only set this bit to ‘1’. As long as this bit is set to one, software is not allowed to update any of the values in this 32-bit word. In case software wants to deactivate the buffer, it must write a one to the corresponding “skip” bit in the USB Endpoint skip register. Hardware can only write this bit to zero. It will do this when it receives a short packet or when the NBytes field transitions to zero or when software has written a one to the “skip” bit.
D	R/W	Disabled 0: The selected endpoint is enabled. 1: The selected endpoint is disabled. If a USB token is received for an endpoint that has the disabled bit set, hardware will ignore the token and not return any data or handshake. When a bus reset is received, software must set the disable bit of all endpoints to 1. Software can only modify this bit when the active bit is zero.
S	R/W	Stall 0: The selected endpoint is not stalled. 1: The selected endpoint is stalled. The Active bit has always higher priority than the Stall bit. This means that a Stall handshake is only sent when the active bit is zero and the stall bit is one. Software can only modify this bit when the active bit is zero.
TR	R/W	Toggle Reset When software sets this bit to one, the HW will set the toggle value equal to the value indicated in the “toggle value” (TV) bit. For the control endpoint zero, this is not needed to be used because the hardware resets the endpoint toggle to one for both directions when a setup token is received. For the other endpoints, the toggle can only be reset to zero when the endpoint is reset.
RF / TV	R/W	Rate Feedback mode / Toggle value For bulk endpoints and isochronous endpoints this bit is reserved and must be set to zero. For the control endpoint zero this bit is used as the toggle value. When the toggle reset bit is set, the data toggle is updated with the value programmed in this bit. When the endpoint is used as an interrupt endpoint, it can be set to the following values. 0: Interrupt endpoint in ‘toggle mode’. 1: Interrupt endpoint in ‘rate feedback mode’. This means that the data toggle is fixed to zero for all data packets. When the interrupt endpoint is in ‘rate feedback mode’, the TR bit must always be set to zero.

Table 360. Endpoint commands

Symbol	Access	Description
T	R/W	Endpoint Type 0: Generic endpoint. The endpoint is configured as a bulk or interrupt endpoint. 1: Isochronous endpoint
NBytes	R/W	For OUT endpoints this is the number of bytes that can be received in this buffer. For IN endpoints this is the number of bytes that must be transmitted. HW decrements this value with the packet size every time when a packet is successfully transferred. Note: If a short packet is received on an OUT endpoint, the active bit will be cleared and the NBytes value indicates the remaining buffer space that is not used. Software calculates the received number of bytes by subtracting the remaining NBytes from the programmed value.
Address Offset	R/W	Bits 21 to 6 of the buffer start address. The address offset is updated by hardware after each successful reception/transmission of a packet. Hardware increments the original value with the integer value when the packet size is divided by 64. Examples: • If an isochronous packet of 200 bytes is successfully received, the address offset is incremented by 3. • If a packet of 64 bytes is successfully received, the address offset is incremented by 1. • If a packet of less than 64 bytes is received, the address offset is not incremented.

Remark: When receiving a SETUP token for endpoint zero, the HW will only read the SETUP bytes Buffer Address offset to know where it has to store the received SETUP bytes. The hardware will ignore all other fields. In case the SETUP stage contains more than 8 bytes, it will only write the first 8 bytes to memory. A USB compliant host must never send more than 8 bytes during the SETUP stage.

For EP0 transfers, the hardware will do auto handshake as long as the ACTIVE bit is set in EP0_IN/OUT command list. Unlike other endpoints, the hardware will not clear the ACTIVE bit after transfer is done. Thus, the software should manually clear the bit whenever it receives new setup packet and set it only after it has queued the data for control transfer. See Figure 53 “Flowchart of control endpoint 0 - OUT direction”.

22.7.2 Control endpoint 0

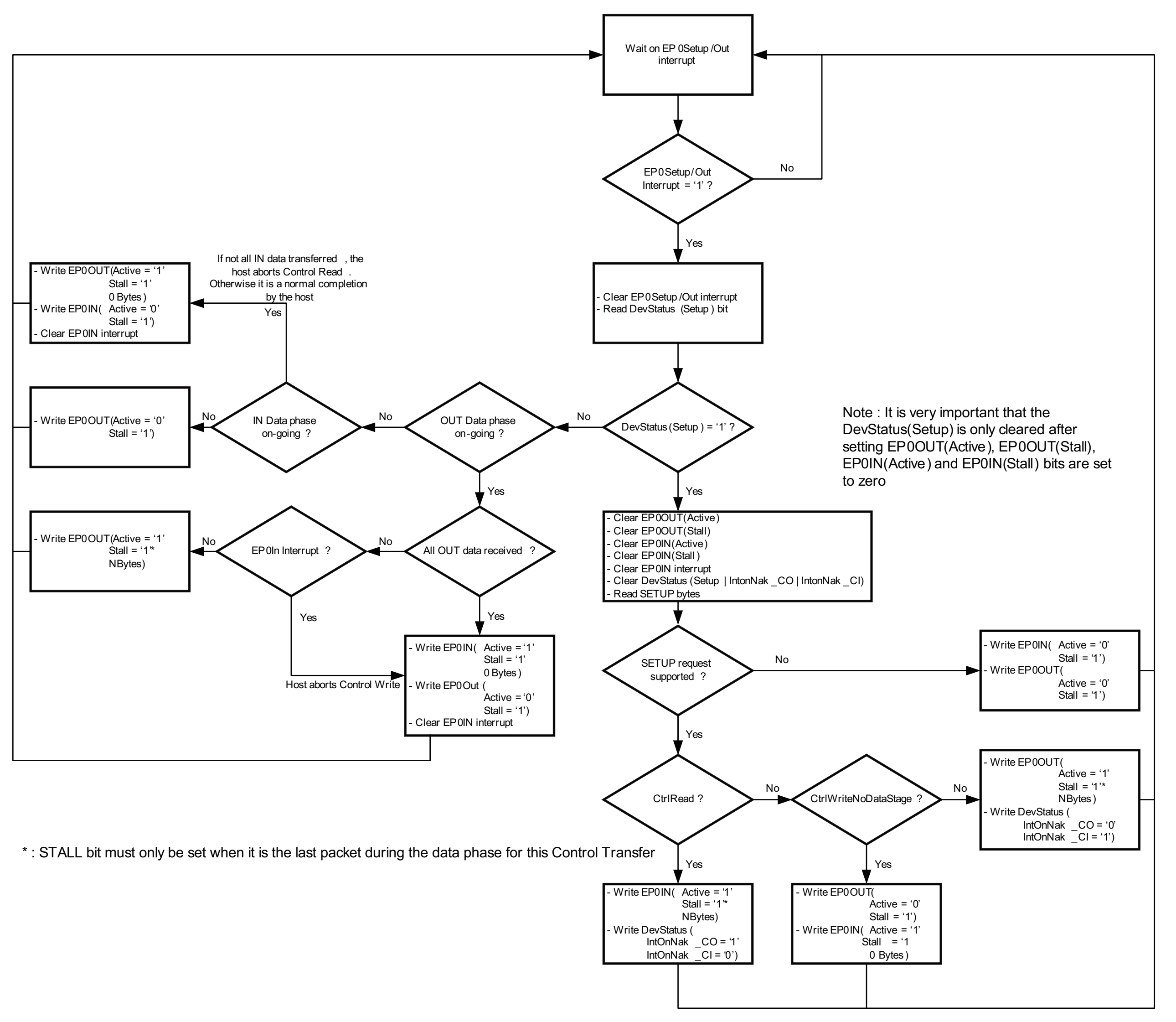

Wait on EP 0Setup /Out interrupt
EP0Setup/Out Interrupt = '1' ?
No
Yes
- Clear EP0Setup /Out interrupt
- Read DevStatus (Setup) bit
DevStatus (Setup) = '1' ?
No
Yes
Note : It is very important that the DevStatus(Setup) is only cleared after setting EP0OUT(Active), EP0OUT(Stall), EP0IN(Active) and EP0IN(Stall) bits are set to zero
- Clear EP0OUT(Active)
- Clear EP0OUT(Stall)
- Clear EP0IN(Active)
- Clear EP0IN(Stall)
- Clear EP0IN interrupt
- Clear DevStatus (Setup | IntonNak _CO | IntonNak _CI)
- Read SETUP bytes
SETUP request supported ?
No
Yes
- Write EP0IN(Active = '0' Stall = '1')
- Write EP0OUT(Active = '0' Stall = '1')
CtrlRead ?
No
Yes
CtrlWriteNoDataStage ?
No
Yes
- Write EP0OUT(Active = '1' Stall = '1'* NBytes)
- Write DevStatus (IntOnNak _CO = '0' IntOnNak _CI = '1')
- Write EP0IN(Active = '1' Stall = '1'* NBytes)
- Write DevStatus (IntOnNak _CO = '1' IntOnNak _CI = '0')
- Write EP0OUT(Active = '0' Stall = '1')
- Write EP0IN(Active = '1' Stall = '1 0 Bytes)
OUT Data phase on-going ?
No
Yes
IN Data phase on-going ?
No
Yes
If not all IN data transferred , the host aborts Control Read . Otherwise it is a normal completion by the host
- Write EP0OUT(Active = '1' Stall = '1' 0 Bytes)
- Write EP0IN(Active = '0' Stall = '1')
- Clear EP0IN interrupt
- Write EP0OUT(Active = '0' Stall = '1')
All OUT data received ?
No
Yes
EP0In Interrupt ?
No
Yes
- Write EP0OUT(Active = '1' Stall = '1'* NBytes)
Host aborts Control Write
- Write EP0IN(Active = '1' Stall = '1' 0 Bytes)
- Write EP0Out (Active = '0' Stall = '1')
- Clear EP0IN interrupt
* : STALL bit must only be set when it is the last packet during the data phase for this Control Transfer

Fig 53. Flowchart of control endpoint 0 - OUT direction

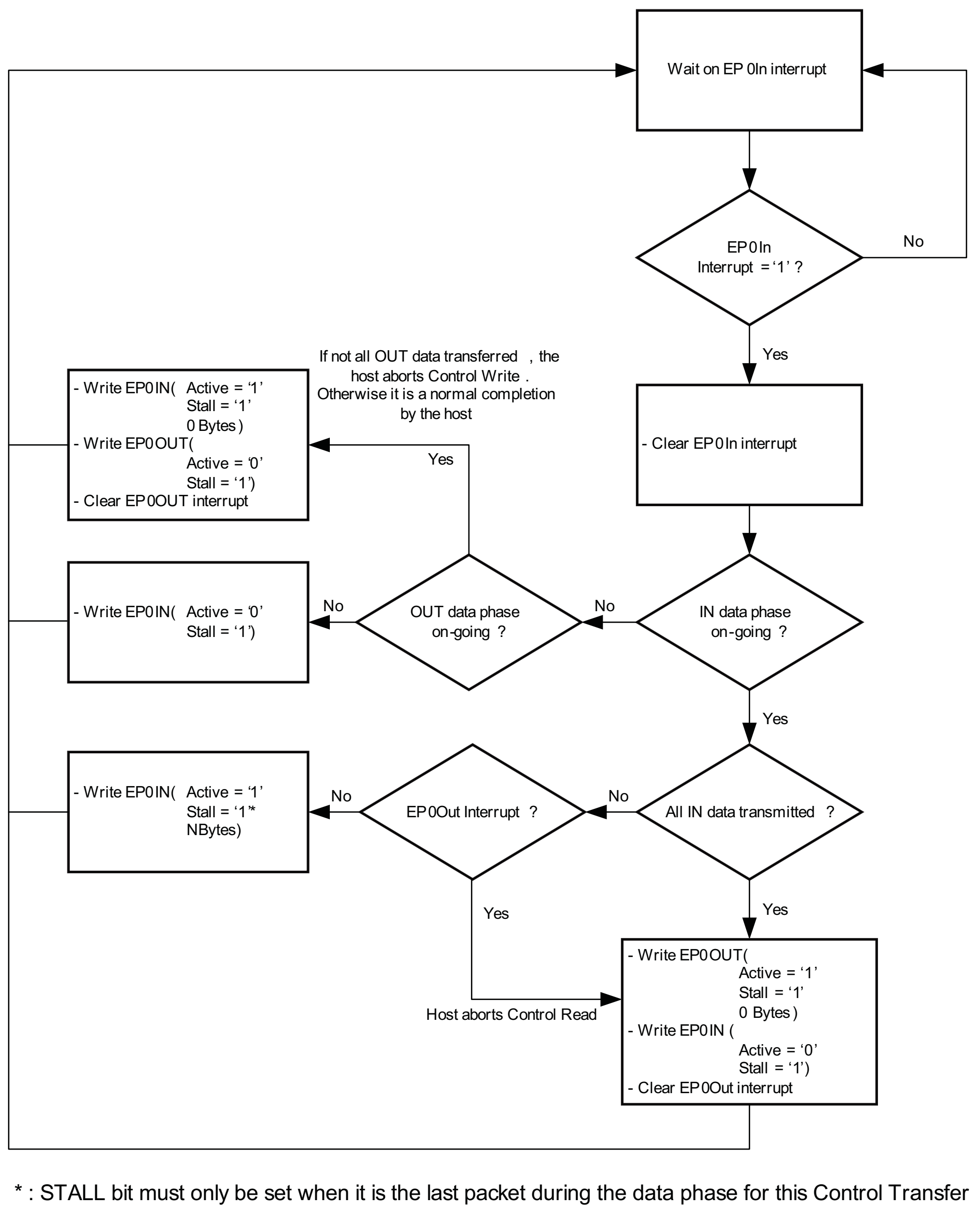

Wait on EP 0In interrupt
EP0In Interrupt = '1' ?
No
Yes
- Clear EP0In interrupt
IN data phase on-going ?
No
OUT data phase on-going ?
Yes
If not all OUT data transferred , the host aborts Control Write . Otherwise it is a normal completion by the host
- Write EP0IN(Active = '1' Stall = '1' 0 Bytes) - Write EP0OUT(Active = '0' Stall = '1') - Clear EP0OUT interrupt
No
- Write EP0IN(Active = '0' Stall = '1')
Yes
All IN data transmitted ?
No
EP0Out Interrupt ?
No
- Write EP0IN(Active = '1' Stall = '1'* NBytes)
Yes
Host aborts Control Read
Yes
- Write EP0OUT(Active = '1' Stall = '1' 0 Bytes) - Write EP0IN (Active = '0' Stall = '1') - Clear EP0Out interrupt
* : STALL bit must only be set when it is the last packet during the data phase for this Control Transfer

Fig 54. Flowchart of control endpoint 0 - IN direction

22.7.3 Generic endpoint: single-buffering

To enable single-buffering, software must set the corresponding "USB EP Buffer Config" bit to zero. In the "USB EP Buffer in use" register, software can indicate which buffer is used in this case.

When software wants to transfer data, it programs the different bits in the Endpoint command/status entry and sets the active bits. The hardware will transmit/receive multiple packets for this endpoint until the NBytes value is equal to zero. When NBytes goes to zero, hardware clears the active bit and sets the corresponding interrupt status bit.

Software must wait until hardware has cleared the Active bit to change some of the command/status bits. This prevents hardware from overwriting a new value programmed by software with some old values that were still cached.

If software wants to disable the active bit before the hardware has finished handling the complete buffer, it can do this by setting the corresponding endpoint skip bit in USB endpoint skip register.

22.7.4 Generic endpoint: double-buffering

To enable double-buffering, software must set the corresponding "USB EP Buffer Config" bit to one. The "USB EP Buffer in use" register indicates which buffer will be used by HW when the next token is received.

When HW clears the active bit of the current buffer in use, it will switch the buffer in use. Software can also force HW to use a certain buffer by writing to the "USB EP Buffer in use" bit.

22.7.5 Special cases

22.7.5.1 Use of the Active bit

The use of the Active bit is a bit different between OUT and IN endpoints.

When data must be received for the OUT endpoint, the software will set the Active bit to one and program the NBytes field to the maximum number of bytes it can receive.

When data must be transmitted for an IN endpoint, the software sets the Active bit to one and programs the NBytes field to the number of bytes that must be transmitted.

22.7.5.2 Generation of a STALL handshake

Special care must be taken when programming the endpoint to send a STALL handshake. A STALL handshake is only sent in the following situations:

- The endpoint is enabled (Disabled bit = 0)
- The active bit of the endpoint is set to 0. (No packet needs to be received/transmitted for that endpoint).
- The stall bit of the endpoint is set to one.

22.7.5.3 Clear Feature (endpoint halt)

When a non-control endpoint has returned a STALL handshake, the host will send a Clear Feature (Endpoint Halt) for that endpoint. When the device receives this request, the endpoint must be un-stalled and the toggle bit for that endpoint must be reset back to zero. In order to do that the software must program the following items for the endpoint that is indicated.

If the endpoint is used in single-buffer mode, program the following:

- Set STALL bit (S) to 0.
- Set toggle reset bit (TR) to 1 and set toggle value bit (TV) to 0.

If the endpoint is used in double-buffer mode, program the following:

- Set the STALL bit of both buffer 0 and buffer 1 to 0.
- Read the buffer in use bit for this endpoint.
- Set the toggle reset bit (TR) to 1 and set the toggle value bit (TV) to 0 for the buffer indicated by the buffer in use bit.

22.7.5.4 Set configuration

When a SetConfiguration request is received with a configuration value different from zero, the device software must enable all endpoints that will be used in this configuration and reset all the toggle values. To do so, it must generate the procedure explained in Section 22.7.5.3 for every endpoint that will be used in this configuration.
For all endpoints that are not used in this configuration, it must set the Disabled bit (D) to one.

22.7.6 USB wake-up

22.7.6.1 Waking up from deep-sleep mode on USB activity

To allow the chip to wake up from deep-sleep mode on USB activity, complete the following steps:

1. Set bit USB_NEED_CLK in the DEVCMDSTAT register (Section 22.6.1) to 0 (default) to enable automatic control of the USB NEEDCLK signal.
2. Wait until USB activity is suspended by polling the DSUS bit in the DSVCMD_STAT register (DSUS = 1).
3. The USB NEEDCLK signal will be deasserted after another 2 ms. Poll the USBCLKST register until the USB NEEDCLK status bit is 0
4. Once the USBCLKST register returns 0, enable the USB activity wake-up interrupt in the NVIC and clear it (see Table 79).
5. Set bit 1 in the USBCLKCTRL register to 1 to trigger the USB activity wake-up interrupt on the rising edge of the USB NEEDCLK signal.
6. Enable the wake-up from deep-sleep mode on this interrupt by enabling the USB NEEDCLK signal in the STARTER0 register (Section 6.5.51, bit 19).
7. Enter deep-sleep mode via the power mode entry API (see Section 8.4.3).

The chip will automatically wake up and resume execution on USB activity.

22.7.6.2 Remote wake-up

To issue a remote wake-up when the USB activity is suspended, complete the following steps:

1. Set bit FORCE_NEEDCLK in the DEVCMDSTAT register to 0 (Section 22.6.1, default) to enable automatic control of the USB NEEDCLK signal.
2. When it is time to issue a remote wake-up, turn on the USB clock and enable the USB clock source.
3. Force the USB clock on by writing a 1 to bit FORCE_NEEDCLK (Section 22.6.1) in the DEVCMDSTAT register.
4. Write a 0 to the DSUS bit in the DSVCMD_STAT register.

5. Wait until the USB leaves the suspend state by polling the DSUS bit in the DSVCMD_STAT register (DSUS =0).
6. Clear the FORCE_NEEDCLK bit (Section 22.6.1, bit 0) in the DEVCMDSTAT to enable automatic USB clock control.

UM11071

Chapter 23: LPC51U68 Flexcomm Interface serial communication

Rev. 1.1 — 17 May 2018 **User manual**

23.1 How to read this chapter

Multiple Flexcomm Interfaces are available on all LPC51U68 parts.

23.2 Introduction

Each Flexcomm Interface provides one peripheral function from a choice of several, chosen by the user. This chapter describes the overall Flexcomm Interface and how to choose peripheral functions. Details of the different peripherals are found in separate chapters for each type.

23.3 Features

Each Flexcomm Interface provides a choice of peripheral functions, one of which must be chosen by the user before the function can be configured and used.

- USART with asynchronous operation or synchronous master or slave operation.
- SPI master or slave, with up to 4 slave selects.
- I^2C, including separate master, slave, and monitor functions.
- Some Flexcomm Interfaces may also provide an I^2S function. If so, there may be from one to four I^2S channel pairs, one of which may optionally be a master and the rest slaves, configured together for either transmit or receive.
- Data for USART, SPI, and I^2S traffic uses the Flexcomm Interface FIFO. The I^2C function does not use the FIFO.

23.4 Basic configuration

The Flexcomm Interface is configured as follows:

1. Peripheral clock: Make sure that the related Flexcomm Interface is enabled in the AHBCLKCTRL1 register (Section 6.5.17).
2. Flexcomm Interface clock: Select a clock source for the related Flexcomm Interface. Options are shown in Figure 9. Also see Section 6.5.28.
3. Select the desired Flexcomm Interface function by writing to the PSELID register of the related Flexcomm Interface (Section 23.7.1).
4. See specific peripheral chapters for information on configuring those peripherals: UART (Chapter 24), SPI (Chapter 25), I2C (Chapter 26), I2S (Chapter 27).

   Remark: The Flexcomm Interface function clock frequency should not be higher than 48 MHz.

23.5 Architecture

The overall structure of one Flexcomm Interface is shown in Figure 55.

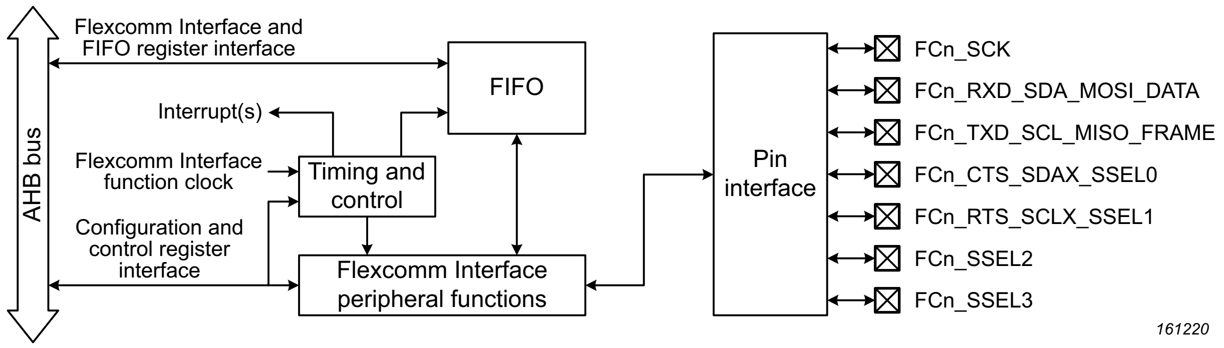


Fig 55. Flexcomm Interface block diagram

23.5.1 Function Summary

LPC51U68 devices include Flexcomm Interfaces and functions as shown in Table 361. Specific part numbers and package variations may include a subset of this list.

Table 361: Flexcomm Interface base addresses and functions

Flexcomm number	Base address	USART (Chapter 24)	SPI (Chapter 25)	I^{2}C (Chapter 26)	I^{2}S (Chapter 27)
0	0x4008 6000	Yes	Yes	Yes	-
1	0x4008 7000	Yes	Yes	Yes, special I^{2}C pins available	-
2	0x4008 8000	Yes	Yes	Yes	-
3	0x4008 9000	Yes	Yes	Yes	-
4	0x4008 A000	Yes	Yes	Yes, special I^{2}C pins available	-
5	0x4009 6000	Yes	Yes	Yes	-
6	0x4009 7000	Yes	Yes	Yes	Yes, 1 channel pair
7	0x4009 8000	Yes	Yes	Yes	Yes, 1 channel pair

23.5.2 Choosing a peripheral function

A specific peripheral function, from among those supported by a particular Flexcomm Interface, is selected by software writing to the PSELID register. Reading the PSELID register provides information on which peripheral functions are available on that Flexcomm Interface.

Once a specific peripheral function has been selected, the PID register will supply an identifier for the selected peripheral. Software may use this information to confirm the selection before proceeding.

23.5.3 FIFO usage

Refer to the chapter for a specific peripheral function for information on how the FIFO is used (see Table 361).

23.5.4 DMA

The Flexcomm Interface generates DMA requests if desired, based on a selectable FIFO level. Refer to the chapter for a specific peripheral function for information on how the FIFO is used (see Table 361).

23.5.5 AHB bus access

Generally, the bus interface to the registers contained in the Flexcomm Interface (including its serial peripheral functions) support only word writes. Byte and halfword writes should not be used. An exception is that the FIFOWR register, when the Flexcomm Interface is configured for use as an SPI interface, allows byte and halfword writes. This allows support for control information embedded in DMA buffers, for example. See Section 25.6.14 "FIFO write data register" for more information.

23.6 Pin description

Each Flexcomm Interface allows up to 7 pin connections. Specific uses of a Flexcomm Interface typically do not use all of these, and some Flexcomm Interface instances may not provide a means to connect all functions to device pins. Pin usage for a specific peripheral function is described in the chapter for that peripheral.

Table 362: Flexcomm Interface Pin Description

Pin	Type	Description
SCK	I/O	Clock input or output for the USART function in synchronous modes.
	I/O	Clock input or output for the SPI function.
	I/O	Clock input or output for the I^{2}S function (if present).
RXD_SDA_MOSI or RXD_SDA_MOSI_DATA	Input	Receive data input for the USART function.
	I/O	SDA (data) input/output for the I^{2}C function.
	I/O	Master data output/slave data input for the SPI function.
	I/O	Data input or output for the I^{2}S function (if present).
TXD_SCL_MISO or TXD_SCL_MISO_WS	Output	Transmit data output for the USART function.
	I/O	SCL input/output for the I^{2}C function.
	I/O	Master data input/slave data output for the SPI function.
	I/O	WS (also known as LRCLK) input or output for the I^{2}S function (if present).
CTS_SDA_SSEL0	Input	Clear To Send input for the USART function.
	I/O	SDA (data) input/output for the I^{2}C function.
	I/O	Slave Select 0 input or output for the SPI function.
RTS_SCL_SSEL1	Output	Request To Send output for the USART function.
	I/O	SCL (clock) input/output for the I^{2}C function.
	I/O	Slave Select 1 input or output for the SPI function.
SSEL2	I/O	Slave Select 2 input or output for the SPI function.
SSEL3	I/O	Slave Select 3 input or output for the SPI function.

23.7 Register description

Each Flexcomm Interface contains registers that are related to configuring the Flexcomm Interface to do a specific peripheral function and other registers related to peripheral FIFOs and data access. The latter depend somewhat on the chosen peripheral functions and are described in the chapters for each specific function (USART, SPI, I2C, and I^{2}S if present in a specific Flexcomm Interface). The Flexcomm Interface registers that identify and configure the Flexcomm Interface are shown in Table 363.

The base addresses of all Flexcomm Interfaces may be found in Table 361.

Table 363: Register map for the first channel pair within one Flexcomm Interface

Name	Access	Offset	Description	Reset Value [1]	Section
PSELID	R/W	0xFF8	Peripheral Select and Flexcomm Interface ID register.	0x0	23.7.1
PID	RO	0xFFC	Peripheral identification register.	Section 23.7.2	23.7.2

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

23.7.1 Peripheral Select and Flexcomm Interface ID register

The PSELID register identifies the Flexcomm Interface and provides information about which peripheral functions are supported by each Flexcomm Interface. It also provides the means to select one peripheral function for each Flexcomm Interface.

Table 364. Peripheral Select and Flexcomm Interface ID register (PSELID - offset 0xFF8) bit description

Bit	Symbol	Value	Description	Reset Value
2:0	PERSEL		Peripheral Select. This field is writable by software.	0x0
		0x0	No peripheral selected.	
		0x1	USART function selected.	
		0x2	SPI function selected.	
		0x3	I^{2}C function selected.	
		0x4	I^{2}S transmit function selected.	
		0x5	I^{2}S receive function selected.	
		0x6	Reserved	
		0x7	Reserved	
3	LOCK		Lock the peripheral select. This field is writable by software.	0x0
		0	Peripheral select can be changed by software.	
		1	Peripheral select is locked and cannot be changed until this Flexcomm Interface or the entire device is reset.	
4	USARTPRESENT		USART present indicator. This field is Read-only.	0x0
		0	This Flexcomm Interface does not include the USART function.	
		1	This Flexcomm Interface includes the USART function.	
5	SPIPRESENT		SPI present indicator. This field is Read-only.	0x0
		0	This Flexcomm Interface does not include the SPI function.	
		1	This Flexcomm Interface includes the SPI function.	
6	I2CPRESENT		I^{2}C present indicator. This field is Read-only.	0x0
		0	This Flexcomm Interface does not include the I^{2}C function.	
		1	This Flexcomm Interface includes the I^{2}C function.	

Table 364. Peripheral Select and Flexcomm Interface ID register (PSELID - offset 0xFF8) bit description

Bit	Symbol	Value	Description	Reset Value
7	I2SPRESENT		I^2S present indicator. This field is Read-only.	0x0
		0	This Flexcomm Interface does not include the I^2S function.	
		1	This Flexcomm Interface includes the I^2S function.	
11:8	-		Reserved. Read value is undefined, only zero should be written.	-
31:12	ID		Flexcomm Interface ID.	0x00101

23.7.2 Peripheral identification register

This register is read-only and will read as 0 until a specific Flexcomm Interface function is selected via the PID register. Once the Flexcomm Interface is configured for a function, this register confirms the selection by returning the module ID for that function, and identifies the revision of that function. A software driver could make use of this information register to implement module type or revision specific behavior.

Table 365. Peripheral identification register (PID - offset 0xFFC) bit description

Bit	Symbol	Description	Reset Value
7:0	-	-	0x0
11:8	Minor_Rev	Minor revision of module implementation.	See specific device chapter
15:12	Major_Rev	Major revision of module implementation.	See specific device chapter
31:16	ID	Module identifier for the selected function.	See specific device chapter

UM11071

Chapter 24: LPC51U68 USARTs

Rev. 1.1 — 17 May 2018 User manual

24.1 How to read this chapter

USART functions are available on all LPC51U68 devices as a selectable function in each Flexcomm Interface peripheral. Up to 8 Flexcomm Interfaces are available.

24.2 Features

- 7, 8, or 9 data bits and 1 or 2 stop bits.
- Synchronous mode with master or slave operation. Includes data phase selection and continuous clock option.
- Multiprocessor/multidrop (9-bit) mode with software address compare.
- RS-485 transceiver output enable.
- Parity generation and checking: odd, even, or none.
- Software selectable oversampling from 5 to 16 clocks in asynchronous mode.
- One transmit and one receive data buffer.
- The USART function supports separate transmit and receive FIFO with 16 entries each.
- RTS/CTS for hardware signaling for automatic flow control. Software flow control can be performed using Delta CTS detect, Transmit Disable control, and any GPIO as an RTS output.
- Break generation and detection.
- Receive data is 2 of 3 sample "voting". Status flag set when one sample differs.
- Built-in Baud Rate Generator.
- Auto-baud mode for automatic baud rate detection.
- Special operating mode allows operation at up to 9600 baud using the 32 kHz RTC oscillator as the USART clock. This mode can be used while the device is in deep-sleep mode and can wake-up the device when a character is received.
- A fractional rate divider is shared among all USARTs.
- Interrupts available for FIFO receive level reached, FIFO transmit level reached, FIFO overflow or underflow, Transmitter Idle, change in receiver break detect, Framing error, Parity error, Delta CTS detect, and receiver sample noise detected (among others).
- USART transmit and receive functions can operated with the system DMA controller.
- Loopback mode for testing of data and flow control.

24.3 Basic configuration

Initial configuration of a USART peripheral is accomplished as follows:

- If needed, use the PRESETCTRL1 register (Table 109) to reset the Flexcomm Interface that is about to have a specific peripheral function selected.

- Select the desired Flexcomm Interface function by writing to the PSELID register of the related Flexcomm Interface (Section 23.7.1).
- Configure the FIFOs for operation.
- Configure USART for receiving and transmitting data:
  - In the AHBCLKCTRL1 register (Table 116), set the appropriate bit for the related Flexcomm Interface in order to enable the clock to the register interface.
  - Enable or disable the related Flexcomm Interface interrupt in the NVIC (see Table 79).
  - Configure the related Flexcomm Interface pin functions via IOCON, see Chapter 9.
  - Configure the Flexcomm Interface clock and USART baud rate. See Section 24.3.1.

    Remark: The Flexcomm Interface function clock frequency should not be above 48 MHz.
- Configure the USART to wake up the part from low power modes. See Section 24.3.2.
- Configure the USART to receive and transmit data in synchronous slave mode. See Section 24.3.2.

24.3.1 Configure the Flexcomm Interface clock and USART baud rate

Each Flexcomm Interface has a separate clock selection, which can include a shared fractional divider (also see Section 24.7.2.3 “32 kHz mode”). The function clock and the fractional divider for the baud rate calculation are set up in the SYSCON block as follows:

1. If a fractional value is needed to obtain a particular baud rate, program the fractional rate divider (FRG, controlled by Syscon register FRGCTRL). The fractional divider value is the fraction of MULT/DIV. The MULT and DIV values are programmed in the FRGCTRL register. The DIV value must be programmed with the fixed value of 256.

   Flexcomm Interface clock = (FRG input clock) / (1+(MULT / DIV))

   The following rules apply for MULT and DIV:

   - Always set DIV to 256 by programming the FRGCTRL register with the value of 0xFF.
   - Set the MULT to any value between 0 and 255.

   See Table 135 for more information on the FRG.
2. In asynchronous mode: configure the baud rate divider BRGVAL in the BRG register. The baud rate divider divides the Flexcomm Interface function clock (FCLK) to create the clock needed to produce the desired baud rate.

   Generally: baud rate = [FCLK / oversample rate] / BRG divide

   With specific register values: baud rate = [FCLK / (OSRVAL+1)] / (BRGVAL + 1)

   Generally: BRG divide = [FCLK / oversample rate] / baud rate

   With specific register values: BRGVAL = [[FCLK / (OSRVAL + 1)] / baud rate] - 1

   See Section 24.6.6 “USART Baud Rate Generator register”.
3. In synchronous master mode: The serial clock is Un_SCLK = FCLK / (BRGVAL+1).

The USART can also be clocked by the 32 kHz RTC oscillator. Set the MODE32K bit to enable this 32 kHz mode. See also Section 24.7.2.3 “32 kHz mode”.

For details on the clock configuration see:

Section 24.7.2 "Clocking and baud rates"

24.3.2 Configure the USART for wake-up

A USART can wake up the system from sleep mode in asynchronous or synchronous mode on any enabled USART interrupt.

In deep-sleep mode, there are two options for configuring USART for wake-up:

- If the USART is configured for synchronous slave mode, the USART block can create an interrupt on a received signal even when the USART block receives no on-chip clocks - that is in deep-sleep mode.

  As long as the USART receives a clock signal from the master, it can receive up to one byte in the RXDAT register while in deep-sleep mode. Any interrupt raised as part of the receive data process can then wake up the part.
- If the 32 kHz mode is enabled, the USART can run in asynchronous mode using the 32 kHz RTC oscillator and create interrupts.

24.3.2.1 Wake-up from sleep mode

- Configure the USART in either asynchronous mode or synchronous mode. See Table 369.
- Enable the USART interrupt in the NVIC.
- Any enabled USART interrupt wakes up the part from sleep mode.

24.3.2.2 Wake-up from deep-sleep mode

- Configure the USART in synchronous slave mode. See Table 369. The SCLK function must be connected to a pin and also connect the pin to the master. Alternatively, the 32 kHz mode can be enabled and the USART operated in asynchronous mode with the 32 kHz RTC oscillator.
- Enable the USART interrupt in the STARTER0 register. See Table 155.
- Enable the USART interrupt in the NVIC.
- The USART wakes up the part from deep-sleep mode on all events that cause an interrupt and are enabled. Typical wake-up events are:
  - A start bit has been received.
  - Received data becomes available.
  - In synchronous mode, data is available in the FIFO to be transmitted, and a serial clock from the master has been received.
  - A change in the state of the CTS pin if the CTS function is connected.
  - **Remark:** By enabling or disabling specific USART interrupts, you can customize when the wake-up occurs.

Wake-up for DMA only

The device can optionally be woken up only far enough to perform needed DMA before returning to deep-sleep mode, without the CPU waking up at all. To accomplish this:

- Set up the appropriate USART function to use DMA, and set the related WAKE bit (WAKETX for the transmit function, and WAKERX for the receive function) in the FIFOCFG register.
- Configure the DMA controller appropriately, including a transfer complete interrupt.
- Disable the related USART interrupt in the NVIC.
- Enable the DMA interrupt in the NVIC.
- Enable FCWAKE and WAKEDMA in the HWWAKE register in Syscon (see Section 6.5.54).

24.4 Pin description

The USART receive, transmit, and control signals are movable Flexcomm Interface functions and are assigned to external pins through via IOCON. See the IOCON description (Chapter 9) to assign the USART functions to pins on the device package.

Table 366. USART pin description

Pin	Type	Name used in Pin Configuration chapter	Description
TXD	O	FCn_TXD_SCL_MISO_WS	Transmitter output for USART on Flexcomm Interface n. Serial transmit data.
RXD	I	FCn_RXD_SDA_MOSI_DATA	Receiver input for USART on Flexcomm Interface n. Serial receive data.
$\overline{\text{RTS}}$	O	FCn_RTS_SCL_SSEL1	Request To Send output for USART on Flexcomm Interface n. This signal supports inter-processor communication through the use of hardware flow control. This signal can also be configured to act as an output enable for an external RS-485 transceiver. RTS is active when the USART RTS signal is configured to appear on a device pin.
$\overline{\text{CTS}}$	I	FCn_CTS_SDA_SSEL0	Clear To Send input for USART on Flexcomm Interface n. Active low signal indicates that the external device that is in communication with the USART is ready to accept data. This feature is active when enabled by the CTSEn bit in CFG register and when configured to appear on a device pin. When deasserted (high) by the external device, the USART will complete transmitting any character already in progress, then stop until CTS is again asserted (low).
SCLK	I/O	FCn_SCK	Serial clock input/output for USART on Flexcomm Interface n in synchronous mode. Clock input or output in synchronous mode. **Remark:** When the USART is configured as a master, such that SCK is an output, it must actually be connected to a pin in order for the USART to work properly.

Recommended IOCON settings are shown in Table 367. See Chapter 9 for definitions of pin types.

Table 367: Suggested USART pin settings

IOCON bit(s)	Type D pin	Type A pin	Type I pin
10	OD: Set to 0 unless open-drain output is desired.	Same as type D.	I2CFILTER: Set to 1.
9	SLEW: Generally set to 0.	Not used, set to 0.	I2CDRIVE: Set to 0.
8	FILTEROFF: Generally set to 1.	Same as type D.	Same as type D.
7	DIGIMODE: Set to 1.	Same as type D.	Same as type D.
6	INVERT: Set to 0.	Same as type D.	Same as type D.
5	Not used, set to 0.	Not used, set to 0.	I2CSLEW: Set to 1.
4:3	MODE: Set 0 (pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input).	Same as type D.	Not used, set to 0.
2:0	FUNC: Must select the correct function for this peripheral.	Same as type D.	Same as type D.
General comment	A good choice for USART input or output.	A reasonable choice for USART input or output.	Not recommended for USART functions that can be outputs in the chosen mode.

24.5 General description

The USART receiver block monitors the serial input line, Un_RXD, for valid input. The receiver shift register assembles characters as they are received, after which they are passed to the receiver FIFO to await access by the CPU or DMA controller.

The USART transmitter block accepts data written by the CPU or DMA controller to the transmit FIFO. When the transmitter is available, the transmit shift register takes that data, formats it, and serializes it to the serial output, Un_TXD.

The Baud Rate Generator block divides the incoming clock to create an oversample clock (typically 16x) in the standard asynchronous operating mode. The BRG clock input source is the shared Fractional Rate Generator that runs from the USART function clock. The 32 kHz operating mode generates a specially timed internal clock based on the RTC oscillator frequency.

In synchronous slave mode, data is transmitted and received using the serial clock directly. In synchronous master mode, data is transmitted and received using the baud rate clock without division.

Status information from the transmitter and receiver is provided via the STAT register. Many of the status flags are able to generate interrupts, as selected by software. The INTSTAT register provides a view of all interrupts that are both enabled and pending.

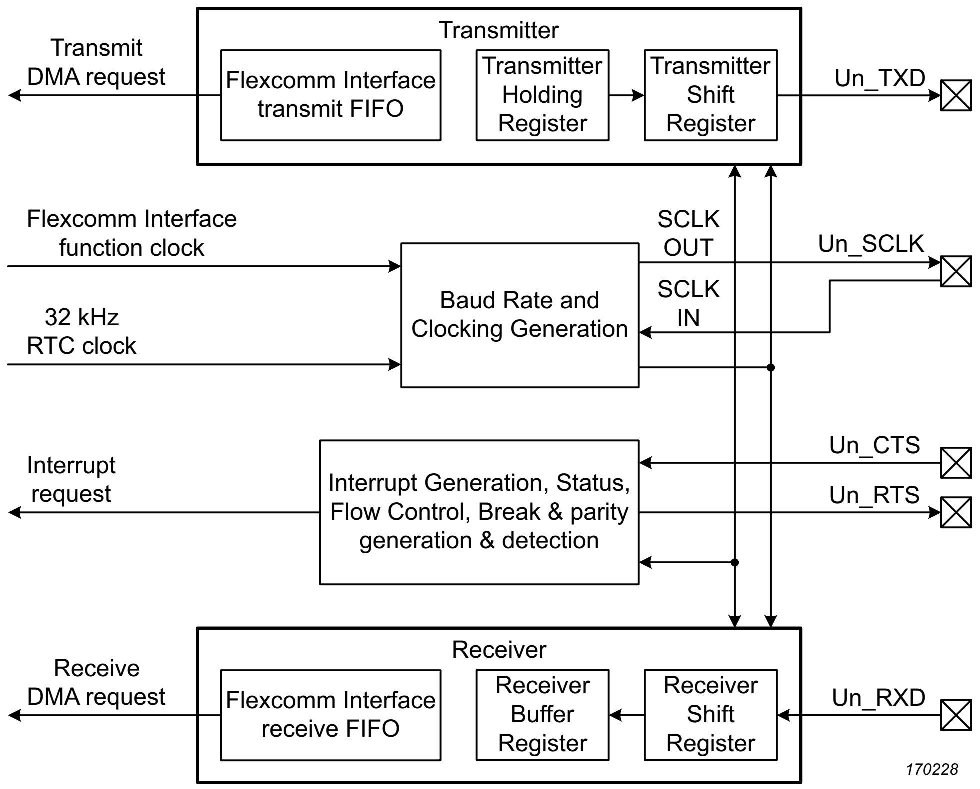

Transmit DMA request
Transmitter
Flexcomm Interface transmit FIFO
Transmitter Holding Register
Transmitter Shift Register
Un_TXD
Flexcomm Interface function clock
32 kHz RTC clock
Baud Rate and Clocking Generation
SCLK OUT
SCLK IN
Un_SCLK
Interrupt request
Interrupt Generation, Status, Flow Control, Break & parity generation & detection
Un_CTS
Un_RTS
Receive DMA request
Receiver
Flexcomm Interface receive FIFO
Receiver Buffer Register
Receiver Shift Register
Un_RXD
170228

Fig 56. USART block diagram

24.6 Register description

The reset value reflects the data stored in used bits only. It does not include the content of reserved bits. Address offsets are within the related Flexcomm Interface address space **after** the USART function has been selected for that Flexcomm Interface (see Section 23.5.1 for a summary of Flexcomm Interface addresses).

Table 368: USART register overview

Name	Access	Offset	Description	Reset value	Section
Registers for the USART function:					
CFG	R/W	0x000	USART Configuration register. Basic USART configuration settings that typically are not changed during operation.	0x0	24.6.1
CTL	R/W	0x004	USART Control register. USART control settings that are more likely to change during operation.	0x0	24.6.2
STAT	R/W	0x008	USART Status register. The complete status value can be read here. Writing ones clears some bits in the register. Some bits can be cleared by writing a 1 to them.	0x0A	24.6.3
INTENSET	R/W	0x00C	Interrupt Enable read and Set register for USART (not FIFO) status. Contains individual interrupt enable bits for each potential USART interrupt. A complete value may be read from this register. Writing a 1 to any implemented bit position causes that bit to be set.	0x0	24.6.4
INTENCLR	WO	0x010	Interrupt Enable Clear register. Allows clearing any combination of bits in the INTENSET register. Writing a 1 to any implemented bit position causes the corresponding bit to be cleared.	-	24.6.5
BRG	R/W	0x020	Baud Rate Generator register. 16-bit integer baud rate divisor value.	0x0	24.6.6
INTSTAT	RO	0x024	Interrupt status register. Reflects interrupts that are currently enabled.	0x0	24.6.7
OSR	R/W	0x028	Oversample selection register for asynchronous communication.	0xF	24.6.8
ADDR	R/W	0x02C	Address register for automatic address matching.	0x0	24.6.9
Registers for FIFO control and data access:					
FIFOCFG	R/W	0xE00	FIFO configuration and enable register.	0x0	24.6.10
FIFOSTAT	R/W	0xE04	FIFO status register.	0x30	24.6.11
FIFOTRIG	R/W	0xE08	FIFO trigger level settings for interrupt and DMA request.	0x0	24.6.12
FIFOINTENSET	R/W1S	0xE10	FIFO interrupt enable set (enable) and read register.	0x0	24.6.13
FIFOINTENCLR	R/W1C	0xE14	FIFO interrupt enable clear (disable) and read register.	0x0	24.6.14
FIFOINTSTAT	RO	0xE18	FIFO interrupt status register.	0x0	24.6.15
FIFOWR	WO	0xE20	FIFO write data.	-	24.6.16
FIFORD	RO	0xE30	FIFO read data.	-	24.6.17
FIFORDNOPOP	RO	0xE40	FIFO data read with no FIFO pop.	-	24.6.18
ID register:					
ID	RO	0xFFC	USART module Identification. This value appears in the shared Flexcomm Interface peripheral ID register when USART is selected.	0xE010 0000	24.6.19

24.6.1 USART Configuration register

The CFG register contains communication and mode settings for aspects of the USART that would normally be configured once in an application.

Remark: Only the CFG register can be written when the ENABLE bit = 0. CFG can be set up by software with ENABLE = 1, then the rest of the USART can be configured.

Remark: If software needs to change configuration values, the following sequence should be used: 1) Make sure the USART is not currently sending or receiving data. 2) Disable the USART by writing a 0 to the Enable bit (0 may be written to the entire register). 3) Write the new configuration value, with the ENABLE bit set to 1.

Table 369. USART Configuration register (CFG, offset 0x000) bit description

Bit	Symbol	Value	Description	Reset Value
0	ENABLE		USART Enable.	0x0
		0	Disabled. The USART is disabled and the internal state machine and counters are reset. While Enable = 0, all USART interrupts and DMA transfers are disabled. When Enable is set again, CFG and most other control bits remain unchanged. When re-enabled, the USART will immediately be ready to transmit because the transmitter has been reset and is therefore available.	
		1	Enabled. The USART is enabled for operation.	
1	-	-	Reserved. Read value is undefined, only zero should be written.	-
3:2	DATALEN		Selects the data size for the USART.	0x0
		0x0	7 bit Data length.	
		0x1	8 bit Data length.	
		0x2	9 bit data length. The 9th bit is commonly used for addressing in multidrop mode. See the ADDRDET bit in the CTL register.	
		0x3	Reserved.	
5:4	PARITYSEL		Selects what type of parity is used by the USART.	0x0
		0x0	No parity.	
		0x1	Reserved.	
		0x2	Even parity. Adds a bit to each character such that the number of 1s in a transmitted character is even, and the number of 1s in a received character is expected to be even.	
		0x3	Odd parity. Adds a bit to each character such that the number of 1s in a transmitted character is odd, and the number of 1s in a received character is expected to be odd.	
6	STOPLEN		Number of stop bits appended to transmitted data. Only a single stop bit is required for received data.	0x0
		0	1 stop bit.	
		1	2 stop bits. This setting should only be used for asynchronous communication.	
7	MODE32K		Selects standard or 32 kHz clocking mode.	0x0
		0	Disabled. USART uses standard clocking.	
		1	Enabled. USART uses the 32 kHz clock from the RTC oscillator as the clock source to the BRG, and uses a special bit clocking scheme.	
8	LINMODE		LIN break mode enable.	0x0
		0	Disabled. Break detect and generate is configured for normal operation.	
		1	Enabled. Break detect and generate is configured for LIN bus operation.	

Table 369. USART Configuration register (CFG, offset 0x000) bit description *…continued*

Bit	Symbol	Value	Description	Reset Value
9	CTSEN		CTS Enable. Determines whether CTS is used for flow control. CTS can be from the input pin, or from the USART's own RTS if loopback mode is enabled.	0x0
		0	No flow control. The transmitter does not receive any automatic flow control signal.	
		1	Flow control enabled. The transmitter uses the CTS input (or RTS output in loopback mode) for flow control purposes.	
10	-	-	Reserved. Read value is undefined, only zero should be written.	-
11	SYNCEN		Selects synchronous or asynchronous operation.	0x0
		0	Asynchronous mode.	
		1	Synchronous mode.	
12	CLKPOL		Selects the clock polarity and sampling edge of received data in synchronous mode.	0x0
		0	Falling edge. Un_RXD is sampled on the falling edge of SCLK.	
		1	Rising edge. Un_RXD is sampled on the rising edge of SCLK.	
13	-	-	Reserved. Read value is undefined, only zero should be written.	-
14	SYNCMST		Synchronous mode Master select.	0x0
		0	Slave. When synchronous mode is enabled, the USART is a slave.	
		1	Master. When synchronous mode is enabled, the USART is a master.	
15	LOOP		Selects data loopback mode.	0x0
		0	Normal operation.	
		1	Loopback mode. This provides a mechanism to perform diagnostic loopback testing for USART data. Serial data from the transmitter (Un_TXD) is connected internally to serial input of the receive (Un_RXD). Un_TXD and Un_RTS activity will also appear on external pins if these functions are configured to appear on device pins. The receiver RTS signal is also looped back to CTS and performs flow control if enabled by CTSEN.	
17:16	-	-	Reserved. Read value is undefined, only zero should be written.	-
18	OETA		Output Enable Turnaround time enable for RS-485 operation.	0x0
		0	Disabled. If selected by OESEL, the Output Enable signal deasserted at the end of the last stop bit of a transmission.	
		1	Enabled. If selected by OESEL, the Output Enable signal remains asserted for one character time after the end of the last stop bit of a transmission. OE will also remain asserted if another transmit begins before it is deasserted.	
19	AUTOADDR		Automatic Address matching enable.	0x0
		0	Disabled. When addressing is enabled by ADDRDET, address matching is done by software. This provides the possibility of versatile addressing (e.g. respond to more than one address).	
		1	Enabled. When addressing is enabled by ADDRDET, address matching is done by hardware, using the value in the ADDR register as the address to match.	
20	OESEL		Output Enable Select.	0x0
		0	Standard. The RTS signal is used as the standard flow control function.	
		1	RS-485. The RTS signal configured to provide an output enable signal to control an RS-485 transceiver.	
21	OEPOL		Output Enable Polarity.	0x0
		0	Low. If selected by OESEL, the output enable is active low.	
		1	High. If selected by OESEL, the output enable is active high.	

Table 369. USART Configuration register (CFG, offset 0x000) bit description *…continued*

Bit	Symbol	Value	Description	Reset Value
22	RXPOL		Receive data polarity.	0x0
		0	Standard. The RX signal is used as it arrives from the pin. This means that the RX rest value is 1, start bit is 0, data is not inverted, and the stop bit is 1.	
		1	Inverted. The RX signal is inverted before being used by the USART. This means that the RX rest value is 0, start bit is 1, data is inverted, and the stop bit is 0.	
23	TXPOL		Transmit data polarity.	0x0
		0	Standard. The TX signal is sent out without change. This means that the TX rest value is 1, start bit is 0, data is not inverted, and the stop bit is 1.	
		1	Inverted. The TX signal is inverted by the USART before being sent out. This means that the TX rest value is 0, start bit is 1, data is inverted, and the stop bit is 0.	
31:24	-	-	Reserved. Read value is undefined, only zero should be written.	-

24.6.2 USART Control register

The CTL register controls aspects of USART operation that are more likely to change during operation.

Table 370. USART Control register (CTL, offset 0x004) bit description

Bit	Symbol	Value	Description	Reset Value
0	-	-	Reserved. Read value is undefined, only zero should be written.	-
1	TXBRKEN		Break Enable.	0x0
		0	Normal operation.	
		1	Continuous break. Continuous break is sent immediately when this bit is set, and remains until this bit is cleared. A break may be sent without danger of corrupting any currently transmitting character if the transmitter is first disabled (TXDIS in CTL is set) and then waiting for the transmitter to be disabled (TXDISINT in STAT = 1) before writing 1 to TXBRKEN.	
2	ADDRDET		Enable address detect mode.	0x0
		0	Disabled. The USART presents all incoming data.	
		1	Enabled. The USART receiver ignores incoming data that does not have the most significant bit of the data (typically the 9th bit) = 1. When the data MSB bit = 1, the receiver treats the incoming data normally, generating a received data interrupt. Software can then check the data to see if this is an address that should be handled. If it is, the ADDRDET bit is cleared by software and further incoming data is handled normally.	
5:3	-	-	Reserved. Read value is undefined, only zero should be written.	-
6	TXDIS		Transmit Disable.	0x0
		0	Not disabled. USART transmitter is not disabled.	
		1	Disabled. USART transmitter is disabled after any character currently being transmitted is complete. This feature can be used to facilitate software flow control.	
7	-	-	Reserved. Read value is undefined, only zero should be written.	-

Table 370. USART Control register (CTL, offset 0x004) bit description

Bit	Symbol	Value	Description	Reset Value
8	CC		Continuous Clock generation. By default, SCLK is only output while data is being transmitted in synchronous mode.	0x0
		0	Clock on character. In synchronous mode, SCLK cycles only when characters are being sent on Un_TXD or to complete a character that is being received.	
		1	Continuous clock. SCLK runs continuously in synchronous mode, allowing characters to be received on Un_RxD independently from transmission on Un_TXD).	
9	CLRCCONRX		Clear Continuous Clock.	0x0
		0	No effect. No effect on the CC bit.	
		1	Auto-clear. The CC bit is automatically cleared when a complete character has been received. This bit is cleared at the same time.	
15:10	-	-	Reserved. Read value is undefined, only zero should be written.	-
16	AUTOBAUD		Auto-baud enable.	0x0
		0	Disabled. USART is in normal operating mode.	
		1	Enabled. USART is in auto-baud mode. This bit should only be set when the USART receiver is idle. The first start bit of RX is measured and used the update the BRG register to match the received data rate. AUTOBAUD is cleared once this process is complete, or if there is an AERR.	
31:17	-	-	Reserved. Read value is undefined, only zero should be written.	-

24.6.3 USART Status register

The STAT register primarily provides a set of USART status flags (not including FIFO status) for software to read. Flags other than read-only flags may be cleared by writing ones to corresponding bits of STAT. Interrupt status flags that are read-only and cannot be cleared by software, can be masked using the INTENCLR register (see Table 373).

The error flags for received noise, parity error, and framing error are set immediately upon detection and remain set until cleared by software action in STAT.

Table 371. USART Status register (STAT, offset 0x008) bit description

Bit	Symbol	Description	Reset value	Access [1]
0	-	Reserved. Read value is undefined, only zero should be written.	-	-
1	RXIDLE	Receiver Idle. When 0, indicates that the receiver is currently in the process of receiving data. When 1, indicates that the receiver is not currently in the process of receiving data.	0x1	RO
2	-	Reserved. Read value is undefined, only zero should be written.	-	-
3	TXIDLE	Transmitter Idle. When 0, indicates that the transmitter is currently in the process of sending data.When 1, indicate that the transmitter is not currently in the process of sending data.	0x1	RO
4	CTS	This bit reflects the current state of the CTS signal, regardless of the setting of the CTSEN bit in the CFG register. This will be the value of the CTS input pin unless loopback mode is enabled.	-	RO
5	DELTACTS	This bit is set when a change in the state is detected for the CTS flag above. This bit is cleared by software.	0x0	W1C

Table 371. USART Status register (STAT, offset 0x008) bit description

Bit	Symbol	Description	Reset value	Access [1]
6	TXDISSTAT	Transmitter Disabled Status flag. When 1, this bit indicates that the USART transmitter is fully idle after being disabled via the TXDIS bit in the CFG register (TXDIS = 1).	0x0	RO
9:7	-	Reserved. Read value is undefined, only zero should be written.	-	-
10	RXBRK	Received Break. This bit reflects the current state of the receiver break detection logic. It is set when the Un_RXD pin remains low for 16 bit times. Note that FRAMERRINT will also be set when this condition occurs because the stop bit(s) for the character would be missing. RXBRK is cleared when the Un_RXD pin goes high.	0x0	RO
11	DELTARXBRK	This bit is set when a change in the state of receiver break detection occurs. Cleared by software.	0x0	R/W1C
12	START	This bit is set when a start is detected on the receiver input. Its purpose is primarily to allow wake-up from deep-sleep mode immediately when a start is detected. Cleared by software.	0x0	R/W1C
13	FRAMERRINT	Framing Error interrupt flag. This flag is set when a character is received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source.	0x0	R/W1C
14	PARITYERRINT	Parity Error interrupt flag. This flag is set when a parity error is detected in a received character.	0x0	R/W1C
15	RXNOISEINT	Received Noise interrupt flag. Three samples of received data are taken in order to determine the value of each received data bit, except in synchronous mode. This acts as a noise filter if one sample disagrees. This flag is set when a received data bit contains one disagreeing sample. This could indicate line noise, a baud rate or character format mismatch, or loss of synchronization during data reception.	0x0	R/W1C
16	ABERR	Auto baud Error. An auto baud error can occur if the BRG counts to its limit before the end of the start bit that is being measured, essentially an auto baud time-out.	0x0	R/W1C
31:17	-	Reserved. Read value is undefined, only zero should be written.	-	-

[1] RO = Read-only, W1C = write 1 to clear.

24.6.4 USART Interrupt Enable read and set register

The INTENSET register is used to enable various USART interrupt sources (not including FIFO interrupts). Enable bits in INTENSET are mapped in locations that correspond to the flags in the STAT register. Interrupt enables may also be read back from this register. Writing ones to implemented bits in this register causes those bits to be set. The INTENCLR register is used to clear bits in this register.

Table 372. USART Interrupt Enable read and set register (INTENSET, offset 0x00C) bit description

Bit	Symbol	Description	Reset Value
2:0	-	Reserved. Read value is undefined, only zero should be written.	-
3	TXIDLEEN	When 1, enables an interrupt when the transmitter becomes idle (TXIDLE = 1).	0x0
4	-	Reserved. Read value is undefined, only zero should be written.	-
5	DELTACTSEN	When 1, enables an interrupt when there is a change in the state of the CTS input.	0x0
6	TXDISEN	When 1, enables an interrupt when the transmitter is fully disabled as indicated by the TXDISINT flag in STAT. See description of the TXDISINT bit for details.	0x0

Table 372. USART Interrupt Enable read and set register (INTENSET, offset 0x00C) bit description *…continued*

Bit	Symbol	Description	Reset Value
10:7	-	Reserved. Read value is undefined, only zero should be written.	-
11	DELTARXBRKEN	When 1, enables an interrupt when a change of state has occurred in the detection of a received break condition (break condition asserted or deasserted).	0x0
12	STARTEN	When 1, enables an interrupt when a received start bit has been detected.	0x0
13	FRAMERREN	When 1, enables an interrupt when a framing error has been detected.	0x0
14	PARITYERREN	When 1, enables an interrupt when a parity error has been detected.	0x0
15	RXNOISEEN	When 1, enables an interrupt when noise is detected. See description of the RXNOISEINT bit in Table 371.	0x0
16	ABERREN	When 1, enables an interrupt when an auto baud error occurs.	0x0
31:17	-	Reserved. Read value is undefined, only zero should be written.	-

24.6.5 USART Interrupt Enable Clear register

The INTENCLR register is used to clear bits in the INTENSET register.

Table 373. USART Interrupt Enable clear register (INTENCLR, offset 0x010) bit description

Bit	Symbol	Description	Reset value
2:0	-	Reserved. Read value is undefined, only zero should be written.	-
3	TXIDLECLR	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
4	-	Reserved. Read value is undefined, only zero should be written.	-
5	DELTACTSCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
6	TXDISCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
10:7	-	Reserved. Read value is undefined, only zero should be written.	-
11	DELTARXBRKCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
12	STARTCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
13	FRAMERRCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
14	PARITYERRCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
15	RXNOISECLR	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
16	ABERRCLR	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
31:17	-	Reserved. Read value is undefined, only zero should be written.	-

24.6.6 USART Baud Rate Generator register

The Baud Rate Generator is a simple 16-bit integer divider controlled by the BRG register. The BRG register contains the value used to divide the Flexcomm Interface clock (FCLK) in order to produce the clock used for USART internal operations.

A 16-bit value allows producing standard baud rates from 300 baud and lower at the highest frequency of the device, up to 921,600 baud from a base clock as low as 14.7456 MHz.

Typically, the baud rate clock is 16 times the actual baud rate. This overclocking allows for centering the data sampling time within a bit cell, and for noise reduction and detection by taking three samples of incoming data.

Note that in 32 kHz mode, the baud rate generator is still used and must be set to 0 if 9600 baud is required.

For more information on USART clocking, see Section 24.7.2 and Section 24.3.1.

Remark: To change a baud rate after a USART is running, the following sequence should be used:

1. Make sure the USART is not currently sending or receiving data.
2. Disable the USART by writing a 0 to the Enable bit (0 may be written to the entire register).
3. Write the new BRGVAL.
4. Write to the CFG register to set the Enable bit to 1.

Table 374. USART Baud Rate Generator register (BRG, offset 0x020) bit description

Bit	Symbol	Description	Reset value
15:0	BRGVAL	This value is used to divide the USART input clock to determine the baud rate, based on the input clock from the FRG. 0 = FCLK is used directly by the USART function. 1 = FCLK is divided by 2 before use by the USART function. 2 = FCLK is divided by 3 before use by the USART function. ... 0xFFFF = FCLK is divided by 65,536 before use by the USART function.	0x0
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

24.6.7 USART Interrupt Status register

The read-only INTSTAT register provides a view of those interrupt flags that are currently enabled. This can simplify software handling of interrupts. See Table 371 for detailed descriptions of the interrupt flags.

Table 375. USART Interrupt Status register (INTSTAT, offset 0x024) bit description

Bit	Symbol	Description	Reset value
2:0	-	Reserved. Read value is undefined, only zero should be written.	-
3	TXIDLE	Transmitter Idle status.	0x0
4	-	Reserved. Read value is undefined, only zero should be written.	-
5	DELTACTS	This bit is set when a change in the state of the CTS input is detected.	0x0
6	TXDISINT	Transmitter Disabled Interrupt flag.	0x0
10:7	-	Reserved. Read value is undefined, only zero should be written.	-
11	DELTARXBRK	This bit is set when a change in the state of receiver break detection occurs.	0x0
12	START	This bit is set when a start is detected on the receiver input.	0x0
13	FRAMERRINT	Framing Error interrupt flag.	0x0
14	PARITYERRINT	Parity Error interrupt flag.	0x0
15	RXNOISEINT	Received Noise interrupt flag.	0x0
16	ABERRINT	Auto baud Error Interrupt flag.	0x0
31:17	-	Reserved. Read value is undefined, only zero should be written.	-

24.6.8 Oversample selection register

The OSR register allows selection of oversampling in asynchronous modes. The oversample value is the number of BRG clocks used to receive one data bit. The default is industry standard 16x oversampling.

Changing the oversampling can sometimes allow better matching of baud rates in cases where the function clock rate is not a multiple of 16 times the expected maximum baud rate. For all modes where the OSR setting is used, the USART receiver takes three consecutive samples of input data in the approximate middle of the bit time. Smaller values of OSR can make the sampling position within a data bit less accurate and may potentially cause more noise errors or incorrect data.

Table 376. Oversample selection register (OSR, offset 0x028) bit description

Bit	Symbol	Description	Reset value
3:0	OSRVAL	Oversample Selection Value. 0 to 3 = not supported 0x4 = 5 function clocks are used to transmit and receive each data bit. 0x5 = 6 function clocks are used to transmit and receive each data bit. ... 0xF= 16 function clocks are used to transmit and receive each data bit.	0xF
31:4	-	Reserved, the value read from a reserved bit is not defined.	-

24.6.9 Address register

The ADDR register holds the address for hardware address matching in address detect mode with automatic address matching enabled.

Table 377. Address register (ADDR, offset 0x02C) bit description

Bit	Symbol	Description	Reset value
7:0	ADDRESS	8-bit address used with automatic address matching. Used when address detection is enabled (ADDRDET in CTL = 1) and automatic address matching is enabled (AUTOADDR in CFG = 1).	0x0
31:8	-	Reserved, the value read from a reserved bit is not defined.	-

24.6.10 FIFO Configuration register

This register configures FIFO usage. A peripheral function within the Flexcomm Interface must be selected prior to configuring the FIFO.

Table 378. FIFO Configuration register (FIFOCFG - offset 0xE00) bit description

Bit	Symbol	Value	Description	Reset value	Access
0	ENABLETX		Enable the transmit FIFO.	0x0	R/W
		0	The transmit FIFO is not enabled.		
		1	The transmit FIFO is enabled.		
1	ENABLERX		Enable the receive FIFO.	0x0	R/W
		0	The receive FIFO is not enabled.		
		1	The receive FIFO is enabled.		
3:2	-	-	Reserved. Read value is undefined, only zero should be written.	-	-
5:4	SIZE		FIFO size configuration. This is a read-only field. 0x0 = FIFO is configured as 16 entries of 8 bits. 0x1, 0x2, 0x3 = not applicable to USART.	-	RO
11:6	-	-	Reserved. Read value is undefined, only zero should be written.	-	-
12	DMATX		DMA configuration for transmit.	0x0	R/W
		0	DMA is not used for the transmit function.		
		1	Generate a DMA request for the transmit function if the FIFO is not full. Generally, data interrupts would be disabled if DMA is enabled.		
13	DMARX		DMA configuration for receive.	0x0	R/W
		0	DMA is not used for the receive function.		
		1	Generate a DMA request for the receive function if the FIFO is not empty. Generally, data interrupts would be disabled if DMA is enabled.		
14	WAKETX		Wake-up for transmit FIFO level. This allows the device to be woken from reduced power modes (up to deep-sleep, as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion.See Section 6.5.54 "Hardware Wake-up control register".	0x0	R/W
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the transmit FIFO level reaches the value specified by TXLVL in FIFOTRIG, even when the TXLVL interrupt is not enabled.		
15	WAKERX		Wake-up for receive FIFO level. This allows the device to be woken from reduced power modes (up to deep-sleep, as long as the peripheral function works in that power mode) without enabling the RXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion. See Section 6.5.54 "Hardware Wake-up control register".	0x0	R/W
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the receive FIFO level reaches the value specified by RXLVL in FIFOTRIG, even when the RXLVL interrupt is not enabled.		

Table 378. FIFO Configuration register (FIFOCFG - offset 0xE00) bit description

Bit	Symbol	Value	Description	Reset value	Access
16	EMPTYTX	-	Empty command for the transmit FIFO. When a 1 is written to this bit, the TX FIFO is emptied.	-	WO
17	EMPTYRX	-	Empty command for the receive FIFO. When a 1 is written to this bit, the RX FIFO is emptied.	-	WO
31:18	-	-	Reserved. Read value is undefined, only zero should be written.	-	-

24.6.11 FIFO status register

This register provides status information for the FIFO and also indicates an interrupt from the peripheral function.

Table 379. FIFO status register (FIFOSTAT - offset 0xE04) bit description

Bit	Symbol	Description	Reset value	Access
0	TXERR	TX FIFO error. Will be set if a transmit FIFO error occurs. This could be an overflow caused by pushing data into a full FIFO, or by an underflow if the FIFO is empty when data is needed. Cleared by writing a 1 to this bit.	0x0	R/W1C
1	RXERR	RX FIFO error. Will be set if a receive FIFO overflow occurs, caused by software or DMA not emptying the FIFO fast enough. Cleared by writing a 1 to this bit.	0x0	R/W1C
2	-	Reserved. Read value is undefined, only zero should be written.	-	
3	PERINT	Peripheral interrupt. When 1, this indicates that the peripheral function has asserted an interrupt. The details can be found by reading the peripheral's STAT register.	0x0	RO
4	TXEMPTY	Transmit FIFO empty. When 1, the transmit FIFO is empty. The peripheral may still be processing the last piece of data.	0x1	RO
5	TXNOTFULL	Transmit FIFO not full. When 1, the transmit FIFO is not full, so more data can be written. When 0, the transmit FIFO is full and another write would cause it to overflow.	0x1	RO
6	RXNOTEMPTY	Receive FIFO not empty. When 1, the receive FIFO is not empty, so data can be read. When 0, the receive FIFO is empty.	0x0	RO
7	RXFULL	Receive FIFO full. When 1, the receive FIFO is full. Data needs to be read out to prevent the peripheral from causing an overflow.	0x0	RO
12:8	TXLVL	Transmit FIFO current level. A 0 means the TX FIFO is currently empty, and the TXEMPTY and TXNOTFULL flags will be 1. Other values tell how much data is actually in the TX FIFO at the point where the read occurs. If the TX FIFO is full, the TXEMPTY and TXNOTFULL flags will be 0.	0x0	RO
15:13	-	Reserved. Read value is undefined, only zero should be written.	-	
20:16	RXLVL	Receive FIFO current level. A 0 means the RX FIFO is currently empty, and the RXFULL and RXNOTEMPTY flags will be 0. Other values tell how much data is actually in the RX FIFO at the point where the read occurs. If the RX FIFO is full, the RXFULL and RXNOTEMPTY flags will be 1.	0x0	RO
31:21	-	Reserved. Read value is undefined, only zero should be written.	-	-

24.6.12 FIFO trigger level settings register

This register allows selecting when FIFO-level related interrupts occur.

Table 380. FIFO trigger level settings register (FIFOTRIG - offset 0xE08) bit description

Bit	Symbol	Value	Description	Reset value
0	TXLVLENA		Transmit FIFO level trigger enable. The FIFO level trigger will cause an interrupt if enabled in FIFOINTENSET. This field is not used for DMA requests (see DMATX in FIFOCFG).	0x0
		0	Transmit FIFO level does not generate a FIFO level trigger.	
		1	An interrupt will be generated if the transmit FIFO level reaches the value specified by the TXLVL field in this register.	
1	RXLVLENA		Receive FIFO level trigger enable. This trigger will become an interrupt if enabled in FIFOINTENSET. This field is not used for DMA requests (see DMARX in FIFOCFG).	0x0
		0	Receive FIFO level does not generate a FIFO level trigger.	
		1	An interrupt will be generated if the receive FIFO level reaches the value specified by the RXLVL field in this register.	
10:2	-	-	Reserved. Read value is undefined, only zero should be written.	-
11:8	TXLVL		Transmit FIFO level trigger point. This field is used only when TXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode See Section 6.5.54 “Hardware Wake-up control register”. 0 = generate an interrupt when the TX FIFO becomes empty. 1 = generate an interrupt when the TX FIFO level decreases to one entry. ... 15 = generate an interrupt when the TX FIFO level decreases to 15 entries (is no longer full).	0x0
15:12	-	-	Reserved. Read value is undefined, only zero should be written.	-
19:16	RXLVL		Receive FIFO level trigger point. The RX FIFO level is checked when a new piece of data is received. This field is used only when RXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode See Section 6.5.54 “Hardware Wake-up control register”. 0 = generate an interrupt when the RX FIFO has one entry (is no longer empty). 1 = generate an interrupt when the RX FIFO has two entries. ... 15 = generate an interrupt when the RX FIFO increases to 16 entries (has become full).	0x0
31:20	-	-	Reserved. Read value is undefined, only zero should be written.	-

24.6.13 FIFO interrupt enable set and read

This register is used to enable various interrupt sources. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in this register causes those bits to be set. The FIFOINTENCLR register is used to clear bits in this register.

Table 381. FIFO interrupt enable set and read register (FIFOINTENSET - offset 0xE10) bit description

Bit	Symbol	Value	Description	Reset value
0	TXERR		Determines whether an interrupt occurs when a transmit error occurs, based on the TXERR flag in the FIFOSTAT register.	0x0
		0	No interrupt will be generated for a transmit error.	
		1	An interrupt will be generated when a transmit error occurs.	
1	RXERR		Determines whether an interrupt occurs when a receive error occurs, based on the RXERR flag in the FIFOSTAT register.	0x0
		0	No interrupt will be generated for a receive error.	
		1	An interrupt will be generated when a receive error occurs.	
2	TXLVL		Determines whether an interrupt occurs when a the transmit FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0x0
		0	No interrupt will be generated based on the TX FIFO level.	
		1	If TXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the TX FIFO level decreases to the level specified by TXLVL in the FIFOTRIG register.	
3	RXLVL		Determines whether an interrupt occurs when a the receive FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0x0
		0	No interrupt will be generated based on the RX FIFO level.	
		1	If RXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the when the RX FIFO level increases to the level specified by RXLVL in the FIFOTRIG register.	
31:4	-	-	Reserved. Read value is undefined, only zero should be written.	-

24.6.14 FIFO interrupt enable clear and read

The FIFOINTENCLR register is used to clear interrupt enable bits in FIFOINTENSET. The complete set of interrupt enables may also be read from this register as well as FIFOINTENSET.

Table 382. FIFO interrupt enable clear and read (FIFOINTENCLR - offset 0xE14) bit description

Bit	Symbol	Description	Reset value
0	TXERR	Writing a one to this bit disables the TXERR interrupt.	0x0
1	RXERR	Writing a one to this bit disables the RXERR interrupt.	0x0
2	TXLVL	Writing a one to this bit disables the interrupt caused by the transmit FIFO reaching the level specified by the TXLVL field in the FIFOTRIG register.	0x0
3	RXLVL	Writing a one to this bit disables the interrupt caused by the receive FIFO reaching the level specified by the RXLVL field in the FIFOTRIG register.	0x0
31:4		Reserved. Read value is undefined, only zero should be written.	-

24.6.15 FIFO interrupt status register

The read-only FIFOINTSTAT register provides a view of those interrupt flags that are both pending and currently enabled. This can simplify software handling of interrupts. Refer to the descriptions of interrupts in Section 24.6.11 and Section 24.6.12 for details.

Table 383. FIFO interrupt status register (FIFOINTSTAT - offset 0xE18) bit description

Bit	Symbol	Description	Reset value
0	TXERR	TX FIFO error.	0x0
1	RXERR	RX FIFO error.	0x0
2	TXLVL	Transmit FIFO level interrupt.	0x0
3	RXLVL	Receive FIFO level interrupt.	0x0
4	PERINT	Peripheral interrupt.	0x0
31:5	-	Reserved. Read value is undefined, only zero should be written.	-

24.6.16 FIFO write data register

The FIFOWR register is used to write values to be transmitted to the FIFO.

Table 384. FIFO write data register (FIFOWR - offset 0xE20) bit description

Bit	Symbol	Description	Reset value
8:0	TXDATA	Transmit data to the FIFO.	-

24.6.17 FIFO read data register

The FIFORD register is used to read values that have been received by the FIFO.

Table 385. FIFO read data register (FIFORD - offset 0xE30) bit description

Bit	Symbol	Description	Reset value
8:0	RXDATA	Received data from the FIFO. The number of bits used depends on the DATALEN and PARITYSEL settings.	-
12:9	-	Reserved, the value read from a reserved bit is not defined.	-
13	FRAMERR	Framing Error status flag. This bit reflects the status for the data it is read along with from the FIFO, and indicates that the character was received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source.	-
14	PARITYERR	Parity Error status flag. This bit reflects the status for the data it is read along with from the FIFO. This bit will be set when a parity error is detected in a received character.	-
15	RXNOISE	Received Noise flag. See description of the RxNoiseInt bit in Table 371.	-
31:16	-	Reserved, the value read from a reserved bit is not defined.	-

24.6.18 FIFO data read with no FIFO pop

This register acts in exactly the same way as FIFORD, except that it supplies data from the top of the FIFO without popping the FIFO (i.e. leaving the FIFO state unchanged). This could be used to allow system software to observe incoming data without interfering with the peripheral driver.

Table 386. FIFO data read with no FIFO pop (FIFORDNOPOP - offset 0xE40) bit description

Bit	Symbol	Description	Reset value
8:0	RXDATA	Received data from the FIFO.	-
12:9	-	Reserved, the value read from a reserved bit is not defined.	-
13	FRAMERR	Framing Error status flag.	-

Table 386. FIFO data read with no FIFO pop (FIFORDNOPOP - offset 0xE40) bit description

Bit	Symbol	Description	Reset value
14	PARITYERR	Parity Error status flag.	-
15	RXNOISE	Received Noise flag.	-
31:16	-	Reserved, the value read from a reserved bit is not defined.	-

24.6.19 Module identification register

The ID register identifies the type and revision of the USART module. A generic SW driver can make use of this information register to implement module type or revision specific behavior.

Table 387. Module identification register (ID - offset 0xFFC) bit description

Bit	Symbol	Description	Reset value
7:0	APERTURE	Aperture: encoded as (aperture size/4K) -1, so 0x00 means a 4K aperture.	0x0
11:8	MINOR_REV	Minor revision of module implementation, starting at 0. Software compatibility is expected between minor revisions.	-
15:12	MAJOR_REV	Major revision of module implementation, starting at 0. There may not be software compatibility between major revisions.	-
31:16	ID	Unique module identifier for this IP block.	0xE010

24.7 Functional description

24.7.1 AHB bus access

The bus interface to the USART registers contained in the Flexcomm Interface support only word writes. Byte and halfword writes are not supported in conjunction with the USART function.

24.7.2 Clocking and baud rates

In order to use the USART, clocking details must be defined such as setting up the clock source selection, the BRG, and setting up the FRG if it is the selected clock source.

Also see Section 24.3.1 “Configure the Flexcomm Interface clock and USART baud rate”.

24.7.2.1 Fractional Rate Generator (FRG)

The Fractional Rate Generator can be used to obtain more precise baud rates when the function clock is not a good multiple of standard (or otherwise desirable) baud rates.

The FRG is typically set up to produce an integer multiple of the highest required baud rate, or a very close approximation. The BRG is then used to obtain the actual baud rate needed.

The FRG register controls the Fractional Rate Generator, which provides the base clock that may be used by any Flexcomm Interface. The Fractional Rate Generator creates a lower rate output clock by suppressing selected input clocks. When not needed, the value of 0 can be set for the FRG, which will then not divide the input clock.

The FRG output clock is defined as the input clock divided by 1 + (MULT / 256), where MULT is in the range of 1 to 255. This allows producing an output clock that ranges from the input clock divided by 1+1/256 to 1+255/256 (just more than 1 to just less than 2). Any further division can be done specific to each USART block by the integer BRG divider contained in each USART.

The base clock produced by the FRG cannot be perfectly symmetrical, so the FRG distributes the output clocks as evenly as is practical. Since USARTs normally uses 16x overclocking, the jitter in the fractional rate clock in these cases tends to disappear in the ultimate USART output.

For setting up the fractional divider, see Section 6.5.28 and Section 6.5.36.

24.7.2.2 Baud Rate Generator (BRG)

The Baud Rate Generator (see Section 24.6.6) is used to divide the base clock to produce a rate 16 times the desired baud rate. Typically, standard baud rates can be generated by integer divides of higher baud rates.

24.7.2.3 32 kHz mode

In order to use a 32 kHz clock to operate a USART at any reasonable speed, a number of adaptations need to be made. First, 16x overclocking has to be abandoned. Otherwise, the maximum data rate would be very low. For the same reason, multiple samples of each data bit must be reduced to one. Finally, special clocking has to be used for individual bit times because 32 kHz is not particularly close to an even multiple of any standard baud rate.

When 32 kHz mode is enabled, clocking comes from the RTC oscillator. The FRG is bypassed, and the BRG can be used to divide down the default 9600 baud to lower rates. Other adaptations required to make the USART work for rates up to 9600 baud are done internally. Rate error will be less than one half percent in this mode, provided the RTC oscillator is operating at the intended frequency of 32.768 kHz.

24.7.3 DMA

A DMA request is provided for each USART direction, and can be used in lieu of interrupts for transferring data by configuring the DMA controller and FIFO level triggering appropriately. The DMA controller provides an acknowledgement signal that clears the related request when it completes handling a that request. The transmitter DMA request is asserted when the transmitter can accept more data. The receiver DMA request is asserted when received data is available to be read.

When DMA is used to perform USART data transfers, other mechanisms can be used to generate interrupts when needed. For instance, completion of the configured DMA transfer can generate an interrupt from the DMA controller. Also, interrupts for special conditions, such as a received break, can still generate useful interrupts.

24.7.4 Synchronous mode

In synchronous mode, a master generates a clock as defined by the clock selection and BRG, which is used to transmit and receive data. As a slave, the external clock is used to transmit and receive data. There is no overclocking in either case.

24.7.5 Flow control

The USART supports both hardware and software flow control.

24.7.5.1 Hardware flow control

The USART supports hardware flow control using RTS and/or CTS signalling. If RTS is configured to appear on a device pin so that it can be sent to an external device, it indicates to an external device the ability of the receiver to receive more data. It can also be used internally to throttle the transmitter from the receiver, which can be especially useful if loopback mode is enabled.

If connected to a pin, and if enabled to do so, the CTS input can allow an external device to throttle the USART transmitter. Both internal and external CTS can be used separately or together.

Figure 57 shows an overview of RTS and CTS within the USART.

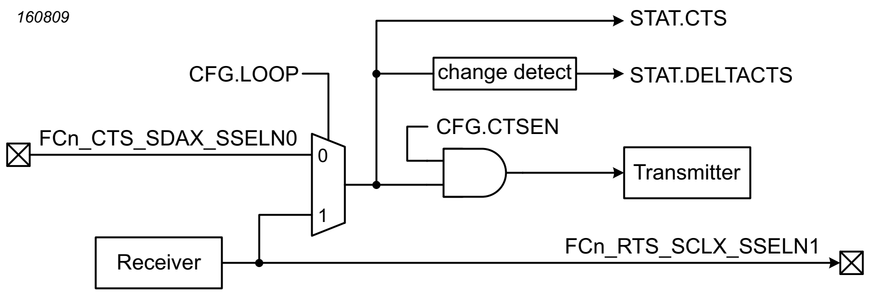


Fig 57. Hardware flow control using RTS and CTS

24.7.5.2 Software flow control

Software flow control could include XON / XOFF flow control, or other mechanisms. these are supported by the ability to check the current state of the CTS input, and/or have an interrupt when CTS changes state (via the CTS and DELTACTS bits, respectively, in the STAT register), and by the ability of software to gracefully turn off the transmitter (via the TXDIS bit in the CTL register).

24.7.6 Auto-baud function

The auto-baud functions attempts to measure the start bit time of the next received character. For this to work, the measured character must have a 1 in the least significant bit position, so that the start bit is bounded by a falling and rising edge. Before an auto-baud operation is requested, the BRG value must be set to 0. The measurement is made using the current clocking settings, including the oversampling configuration. The result is that a value is stored in the BRG register that is as close as possible to the correct setting for the sampled character and the current clocking settings. The sampled character is provided in the RXDAT and RXDATSTAT registers, allowing software to double check for the expected character.

Auto-baud includes a time-out that is flagged by ABERR if no character is received at the expected time. It is recommended that auto-baud only be enabled when the USART receiver is idle. Once enabled, either data will become available in the FIFO or ABERR will be asserted at some point, at which time software should turn off auto-baud.

Auto-baud has no meaning and should not be enabled when the USART is in synchronous mode.

24.7.7 RS-485 support

RS-485 support requires some form of address recognition and data direction control.

This USART has provisions for hardware address recognition (see the AUTOADDR bit in the CFG register in Section 24.6.1 and the ADDR register in Section 24.6.9), as well as software address recognition (see the ADDRDET bit in the CTL register in Section 24.6.2).

Automatic data direction control with the RTS pin can be set up using the OESEL, OEPOL, and OETA bits in the CFG register (Section 24.6.1). Data direction control can also be implemented in software using a GPIO pin.

24.7.8 Oversampling

Typical industry standard USARTs use a 16x oversample clock to transmit and receive asynchronous data. This is the number of BRG clocks used for one data bit. The Oversample Select Register (OSR) allows this USART to use a 16x down to a 5x oversample clock. There is no oversampling in synchronous modes.

Reducing the oversampling can sometimes help in getting better baud rate matching when the baud rate is very high, or the function clock is very low. For example, the closest actual rate near 115,200 baud with a 12 MHz function clock and 16x oversampling is 107,143 baud, giving a rate error of 7%. Changing the oversampling to 15x gets the actual rate to 114,286 baud, a rate error of 0.8%. Reducing the oversampling to 13x gets the actual rate to 115,385 baud, a rate error of only 0.16%.

There is a cost for altering the oversampling. In asynchronous modes, the USART takes three samples of incoming data on consecutive oversample clocks, as close to the center of a bit time as can be done. When the oversample rate is reduced, the three samples spread out and occupy a larger proportion of a bit time. For example, with 5x oversampling, there is one oversample clock, then three data samples taken, then one more oversample clock before the end of the bit time. Since the oversample clock is running asynchronously from the input data, skew of the input data relative to the expected timing has little room for error. At 16x oversampling, there are several oversample clocks before actual data sampling is done, making the sampling more robust. Generally speaking, it is recommended to use the highest oversampling where the rate error is acceptable in the system.

24.7.9 Break generation and detection

A line break may be sent at any time, regardless of other USART activity. Received break is also detected at any time, including during reception of a character. Received break is signaled when the RX input remains low for 16 bit times. Both the beginning and end of a received break are noted by the DELTARXBRK status flag, which can be used as an interrupt. See Section 24.7.10 for details of LIN mode break.

In order to avoid corrupting any character currently being transmitted, it is recommended that the USART transmitter be disabled by setting the TXDIS bit in the CTL register, then waiting for the TXDISSTAT flag to be set prior to sending a break. Then a 1 may be written to the TXBRKEN bit in the CTL register. This sends a break until TXBRKEN is cleared, allowing any length break to be sent.

24.7.10 LIN bus

The only difference between standard operation and LIN mode is that LIN mode alters the way that break generation and detection is performed (see Section 24.7.9 for details of the standard break). When a break is requested by setting the TXBRKEN bit in the CTL register, then sending a dummy character, a 13 bit time break is sent. A received break is flagged when the RX input remains low for 11 bit times. As for non-LIN mode, a received character is also flagged, and accompanied by a framing error status.

As a LIN slave, the auto-baud feature can be used to synchronize to a LIN sync byte, and will return the value of the sync byte as confirmation of success.

Wake-up for LIN can potentially be handled in a number of ways, depending on the system, and what clocks may be running in a slave device. For instance, as long as the USART is receiving internal clocks allowing it to function, it can be set to wake up the CPU for any interrupt, including a received start bit. If there are no clocks running, the GPIO function of the USART RX pin can be programmed to wake up the device.

UM11071

Chapter 25: LPC51U68 Serial Peripheral Interfaces (SPI)

Rev. 1.1 — 17 May 2018 **User manual**

25.1 How to read this chapter

SPI functions are available on all LPC51U68 devices as a selectable function in each Flexcomm Interface peripheral. Up to 8 Flexcomm Interfaces are available.

25.2 Features

- Master and slave operation.
- Data transmits of 4 to 16 bits supported directly. Larger frames supported by software.
- The SPI function supports separate transmit and receive FIFOs with 8 entries each.
- Supports DMA transfers: SPIn transmit and receive functions can be operated with the system DMA controller.
- Data can be transmitted to a slave without the need to read incoming data. This can be useful while setting up an SPI memory.
- Up to four Slave Select input/outputs with selectable polarity and flexible usage.

Remark: Texas Instruments SSI and National Microwire modes are not supported.

25.3 Basic configuration

Initial configuration of an SPI peripheral is accomplished as follows:

- If needed, use the PRESETCTRL1 register (Table 109) to reset the Flexcomm Interface that is about to have a specific peripheral function selected.
- Select the desired Flexcomm Interface function by writing to the PSELID register of the related Flexcomm Interface (Section 23.7.1).
- Configure the FIFOs for operation.
- Configure the SPI for receiving and transmitting data:
  - In the AHBCLKCTRL1 (Table 115) register, set the appropriate bit for the related Flexcomm Interface in order to enable the clock to the register interface.
  - Enable or disable the related Flexcomm Interface interrupts in the NVIC (see Table 79).
  - Configure the required Flexcomm Interface pin functions through IOCON. See Section 25.4.
  - Configure the Flexcomm Interface clock and SPI data rate (see Section 25.7.4).

    Remark: The Flexcomm Interface function clock frequency should not be above 48 MHz.
  - Set the RXIGNORE bit to only transmit data and not read the incoming data. Otherwise, the transmit halts when the FIFORD buffer is full.
- Configure the SPI function to wake up the part from low power modes. See Section 25.3.1.

25.3.1 Configure the SPI for wake-up

In sleep mode, any signal that triggers an SPI interrupt can wake up the part, provided that the interrupt is enabled in the INTENSET register and the NVIC. As long as the SPI clock is configured to be active in sleep mode, the SPI can wake up the part independently of whether the SPI block is configured in master or slave mode.

In deep-sleep mode, the SPI clock is turned off. However, if the SPI is configured in slave mode and an external master on the provides the clock signal, the SPI can create an interrupt asynchronously and wake up the device. The appropriate interrupt(s) must be enabled in the SPI and the NVIC.

25.3.1.1 Wake-up from sleep mode

- Configure the SPI in either master or slave mode. See Table 391.
- Enable the SPI interrupt in the NVIC.
- Any enabled SPI interrupt wakes up the part from sleep mode.

25.3.1.2 Wake-up from deep-sleep mode

- Configure the SPI in slave mode. See Table 391. The SCK function must be connected to a pin and the pin connected to the master.
- Enable the SPI interrupt in the STARTER0 register. See Table 155.
- Enable the SPI interrupt in the NVIC.
- Enable desired SPI interrupts. Examples are the following wake-up events:
  – A change in the state of the SSEL pins.
  – Data available to be received.
  – Receive FIFO overflow.

Wake-up for DMA only

The device can optionally be woken up only far enough to perform needed DMA before returning to deep-sleep mode, without the CPU waking up at all. To accomplish this:

- Set up the appropriate SPI function to use DMA, and set the related WAKE bit (WAKETX for the transmit function, and WAKERX for the receive function) in the FIFOCFG register.
- Configure the DMA controller appropriately, including a transfer complete interrupt.
- Disable the related SPI interrupt in the NVIC.
- Enable the DMA interrupt in the NVIC.
- Enable FCWAKE and WAKEDMA in the HWWAKE register in Syscon (see Section 6.5.54).

25.4 Pin description

The SPI signals are movable Flexcomm Interface functions and are assigned to external pins via IOCON.

Table 388: SPI Pin Description

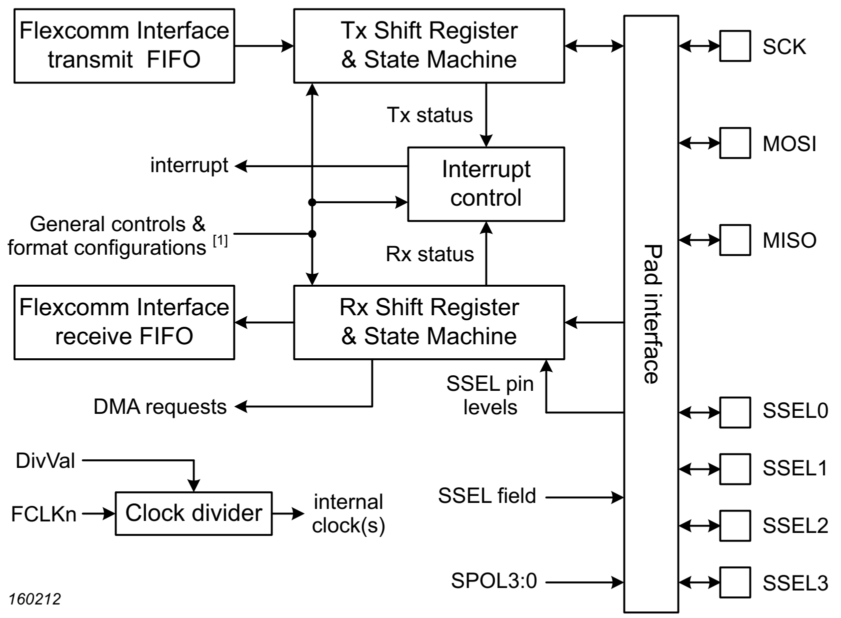
Function	Type	Pin name used in Pin Description chapter	Description
SCK	I/O	FCn_SCK	Serial Clock for SPI on Flexcomm Interface n. SCK is a clock signal used to synchronize the transfer of data. It is driven by the master and received by the slave. When the SPI interface is used, the clock is programmable to be active-high or active-low. SCK only switches during a data transfer. It is driven whenever the Master bit in CFG equals 1, regardless of the state of the Enable bit.
MOSI	I/O	FCn_RXD_SDA_MOSI or FCn_RXD_SDA_MOSI_DATA	Master Out Slave In for SPI on Flexcomm Interface n. The MOSI signal transfers serial data from the master to the slave. When the SPI is a master, it outputs serial data on this signal. When the SPI is a slave, it clocks in serial data from this signal. MOSI is driven whenever the Master bit in CFG equals 1, regardless of the state of the Enable bit.
MISO	I/O	FCn_TXD_SCL_MISO or FCn_TXD_SCL_MISO_WS	Master In Slave Out for SPI on Flexcomm Interface n. The MISO signal transfers serial data from the slave to the master. When the SPI is a master, serial data is input from this signal. When the SPI is a slave, serial data is output to this signal. MISO is driven when the SPI block is enabled, the Master bit in CFG equals 0, and when the slave is selected by one or more SSEL signals.
SSEL0	I/O	FCn_CTS_SDA_SSEL0	Slave Select 0 for SPI on Flexcomm Interface n. When the SPI interface is a master, it will drive the SSEL signals to an active state before the start of serial data and then release them to an inactive state after the serial data has been sent. By default, this signal is active low but can be selected to operate as active high. When the SPI is a slave, any SSEL in an active state indicates that this slave is being addressed. The SSEL pin is driven whenever the Master bit in the CFG register equals 1, regardless of the state of the Enable bit.
SSEL1	I/O	FCn_RTS_SCL_SSEL1	Slave Select 1 for SPI on Flexcomm Interface n.
SSEL2	I/O	FCn_SSEL2	Slave Select 2 for SPI on Flexcomm Interface n.
SSEL3	I/O	FCn_SSEL3	Slave Select 3 for SPI on Flexcomm Interface n.

Recommended IOCON settings are shown in Table 389. See Chapter 9 for definitions of pin types.

Table 389: Suggested SPI pin settings

IOCON bit(s)	Type D pin	Type A pin	Type I pin
10	OD: Set to 0 unless open-drain output is desired.	Same as type D.	I2CFILTER: Set to 1.
9	SLEW: Generally set to 0. Setting to 1 at higher SPI rates may improve performance.	Not used, set to 0.	I2CDRIVE: Set to 0
8	FILTEROFF: Generally set to 1.	Same as type D.	Same as type D.
7	DIGIMODE: Set to 1.	DIGIMODE: Set to 1.	DIGIMODE: Set to 1.
6	INVERT: Set to 0.	Same as type D.	Same as type D.
5	Not used, set to 0.	Same as type D.	I2CSLEW: Set to 1.
4:3	MODE: Set to 0 (pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input).	Same as type D.	Not used, set to 0.
2:0	FUNC: Must select the correct function for this peripheral.	Same as type D.	Same as type D.
General comment	A good choice for SPI input or output.	A reasonable choice for SPI input or output.	Not recommended for SPI functions that can be outputs in the chosen mode.

25.5 General description



(1) Includes CPOL, CPHA, LSBF, LEN, master, enable, transfer_delay, frame_delay, pre_delay, post_delay, SOT, EOT, EOF, RXIGNORE, individual interrupt enables.

Fig 58. SPI block diagram

25.6 Register description

Address offsets are within the address space of the related Flexcomm Interface. The Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

Table 390. SPI register overview

Name	Access	Offset	Description	Reset value	Section
Registers for the SPI function:					
CFG	R/W	0x400	SPI Configuration register.	0x0	25.6.1
DLY	R/W	0x404	SPI Delay register.	0x0	25.6.2
STAT	R/W	0x408	SPI Status. Some status flags can be cleared by writing a 1 to that bit position.	0x100	25.6.3
INTENSET	R/W	0x40C	SPI Interrupt Enable read and Set. A complete value may be read from this register. Writing a 1 to any implemented bit position causes that bit to be set.	0x0	25.6.4
INTENCLR	WO	0x410	SPI Interrupt Enable Clear. Writing a 1 to any implemented bit position causes the corresponding bit in INTENSET to be cleared.	-	25.6.5
DIV	R/W	0x424	SPI clock Divider.	0x0	25.6.6
INTSTAT	RO	0x428	SPI Interrupt Status.	0x0	25.6.7
Registers for FIFO control and data access:					
FIFOCFG	R/W	0xE00	FIFO configuration and enable register.	0x0	25.6.8
FIFOSTAT	R/W	0xE04	FIFO status register.	0x30	25.6.9
FIFOTRIG	R/W	0xE08	FIFO trigger level settings for interrupt and DMA request.	0x0	25.6.10
FIFOINTENSET	R/W1C	0xE10	FIFO interrupt enable set (enable) and read register.	0x0	25.6.11
FIFOINTENCLR	R/W1C	0xE14	FIFO interrupt enable clear (disable) and read register.	0x0	25.6.12
FIFOINTSTAT	RO	0xE18	FIFO interrupt status register.	0x0	25.6.13
FIFOWR	WO	0xE20	FIFO write data.	-	25.6.14
FIFORD	RO	0xE30	FIFO read data.	-	25.6.15
FIFORDNOPOP	RO	0xE40	FIFO data read with no FIFO pop.	-	25.6.16
ID register:					
ID	RO	0xFFC	SPI module Identification. This value appears in the shared Flexcomm Interface peripheral ID register when SPI is selected.	0xE020 0000	25.6.17

25.6.1 SPI Configuration register

The CFG register contains information for the general configuration of the SPI. Typically, this information is not changed during operation. See the description of the master idle status (MSTIDLE in Table 393) for more information.

Remark: A setup sequence is recommended for initial SPI setup (after the SPI function has been selected (see Chapter 23), and when changes need to be made to settings in the CFG register after the interface has been in use. See the list below. In the case of changing existing settings, the interface should first be disabled by clearing the ENABLE bit once the interface is fully idle. See the description of the master idle status (MSTIDLE in Table 393) for more information.

- Disable the FIFO by clearing the ENABLETX and ENABLERX bits in FIFOCFG
- Setup the SPI interface in the CFG register, leaving ENABLE = 0.
- Enable the FIFO by setting the ENABLETX and/or ENABLERX bits in FIFOCFG
- Enable the SPI by setting the ENABLE bit in CFG.

Table 391. SPI Configuration register (CFG, offset 0x400) bit description

Bit	Symbol	Value	Description	Reset value
0	ENABLE		SPI enable.	0x0
		0	Disabled. The SPI is disabled and the internal state machine and counters are reset.	
		1	Enabled. The SPI is enabled for operation.	
1	-	-	Reserved. Read value is undefined, only zero should be written.	-
2	MASTER		Master mode select.	0x0
		0	Slave mode. The SPI will operate in slave mode. SCK, MOSI, and the SSEL signals are inputs, MISO is an output.	
		1	Master mode. The SPI will operate in master mode. SCK, MOSI, and the SSEL signals are outputs, MISO is an input.	
3	LSBF		LSB First mode enable.	0x0
		0	Standard. Data is transmitted and received in standard MSB first order.	
		1	Reverse. Data is transmitted and received in reverse order (LSB first).	
4	CPHA		Clock Phase select.	0x0
		0	Change. The SPI captures serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is changed on the following edge.	
		1	Capture. The SPI changes serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is captured on the following edge.	
5	CPOL		Clock Polarity select.	0x0
		0	Low. The rest state of the clock (between transfers) is low.	
		1	High. The rest state of the clock (between transfers) is high.	
6	-	-	Reserved. Read value is undefined, only zero should be written.	-
7	LOOP		Loopback mode enable. Loopback mode applies only to Master mode, and connects transmit and receive data connected together to allow simple software testing.	0x0
		0	Disabled.	
		1	Enabled.	

Table 391. SPI Configuration register (CFG, offset 0x400) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
8	SPOL0		SSEL0 Polarity select.	0x0
		0	Low. The SSEL0 pin is active low.	
		1	High. The SSEL0 pin is active high.	
9	SPOL1		SSEL1 Polarity select.	0x0
		0	Low. The SSEL1 pin is active low.	
		1	High. The SSEL1 pin is active high.	
10	SPOL2		SSEL2 Polarity select.	0x0
		0	Low. The SSEL2 pin is active low.	
		1	High. The SSEL2 pin is active high.	
11	SPOL3		SSEL3 Polarity select.	0x0
		0	Low. The SSEL3 pin is active low.	
		1	High. The SSEL3 pin is active high.	
31:12	-	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.2 SPI Delay register

The DLY register controls several programmable delays related to SPI signalling. These delays apply only to master mode, and are all stated in SPI clocks.

Timing details are shown in Section 25.7.3.1 “Pre_delay and Post_delay”, Section 25.7.3.2 “Frame_delay”, and Section 25.7.3.3 “Transfer_delay”.

Table 392. SPI Delay register (DLY, offset 0x404) bit description

Bit	Symbol	Description	Reset value
3:0	PRE_DELAY	Controls the amount of time between SSEL assertion and the beginning of a data transfer. There is always one SPI clock time between SSEL assertion and the first clock edge. This is not considered part of the pre-delay. 0x0 = No additional time is inserted. 0x1 = 1 SPI clock time is inserted. 0x2 = 2 SPI clock times are inserted. ... 0xF = 15 SPI clock times are inserted.	0x0
7:4	POST_DELAY	Controls the amount of time between the end of a data transfer and SSEL deassertion. 0x0 = No additional time is inserted. 0x1 = 1 SPI clock time is inserted. 0x2 = 2 SPI clock times are inserted. ... 0xF = 15 SPI clock times are inserted.	0x0

Table 392. SPI Delay register (DLY, offset 0x404) bit description

Bit	Symbol	Description	Reset value
11:8	FRAME_DELAY	If the EOF flag is set, controls the minimum amount of time between the current frame and the next frame (or SSEL deassertion if EOT). 0x0 = No additional time is inserted. 0x1 = 1 SPI clock time is inserted. 0x2 = 2 SPI clock times are inserted. ... 0xF = 15 SPI clock times are inserted.	0x0
15:12	TRANSFE_DELAY	Controls the minimum amount of time that the SSEL is deasserted between transfers. 0x0 = The minimum time that SSEL is deasserted is 1 SPI clock time. (Zero added time.) 0x1 = The minimum time that SSEL is deasserted is 2 SPI clock times. 0x2 = The minimum time that SSEL is deasserted is 3 SPI clock times. ... 0xF = The minimum time that SSEL is deasserted is 16 SPI clock times.	0x0
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.3 SPI Status register

The STAT register provides SPI status flags for software to read, and a control bit for forcing an end of transfer. Flags other than read-only flags may be cleared by writing ones to corresponding bits of STAT.

In this register, the following notation is used: RO = Read-only, W1C = write 1 to clear.

Table 393. SPI Status register (STAT, offset 0x408) bit description

Bit	Symbol	Description	Reset value	Access
3:0	-	Reserved. Read value is undefined, only zero should be written.	-	-
4	SSA	Slave Select Assert. This flag is set whenever any slave select transitions from deasserted to asserted, in both master and slave modes. This allows determining when the SPI transmit/receive functions become busy, and allows waking up the device from reduced power modes when a slave mode access begins. This flag is cleared by software.	0x0	W1C
5	SSD	Slave Select Deassert. This flag is set whenever any asserted slave selects transition to deasserted, in both master and slave modes. This allows determining when the SPI transmit/receive functions become idle. This flag is cleared by software.	0x0	W1C
6	STALLED	Stalled status flag. This indicates whether the SPI is currently in a stall condition.	0x0	RO
7	ENDTRANSFER	End Transfer control bit. Software can set this bit to force an end to the current transfer when the transmitter finishes any activity already in progress, as if the EOT flag had been set prior to the last transmission. This capability is included to support cases where it is not known when transmit data is written that it will be the end of a transfer. The bit is cleared when the transmitter becomes idle as the transfer comes to an end. Forcing an end of transfer in this manner causes any specified FRAME_DELAY and TRANSFER_DELAY to be inserted.	0x0	RO/ W1C
8	MSTIDLE	Master idle status flag. This bit is 1 whenever the SPI master function is fully idle. This means that the transmit holding register is empty and the transmitter is not in the process of sending data.	0x1	RO
31:9	-	Reserved. Read value is undefined, only zero should be written.	-	-

25.6.4 SPI Interrupt Enable read and Set register

The INTENSET register is used to enable various SPI interrupt sources. Enable bits in INTENSET are mapped in locations that correspond to the flags in the STAT register. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in this register causes those bits to be set. The INTENCLR register is used to clear bits in this register. See Table 393 for details of the interrupts.

Table 394. SPI Interrupt Enable read and Set register (INTENSET, offset 0x40C) bit description

Bit	Symbol	Value	Description	Reset value
3:0	-	-	Reserved. Read value is undefined, only zero should be written.	-
4	SSAEN		Slave select assert interrupt enable. Determines whether an interrupt occurs when the Slave Select is asserted.	0x0
		0	Disabled. No interrupt will be generated when any Slave Select transitions from deasserted to asserted.	
		1	Enabled. An interrupt will be generated when any Slave Select transitions from deasserted to asserted.	
5	SSDEN		Slave select deassert interrupt enable. Determines whether an interrupt occurs when the Slave Select is deasserted.	0x0
		0	Disabled. No interrupt will be generated when all asserted Slave Selects transition to deasserted.	
		1	Enabled. An interrupt will be generated when all asserted Slave Selects transition to deasserted.	
7:6	-	-	Reserved. Read value is undefined, only zero should be written.	-
8	MSTIDLEEN		Master idle interrupt enable.	0x0
		0	No interrupt will be generated when the SPI master function is idle.	
		1	An interrupt will be generated when the SPI master function is fully idle.	
31:9	-	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.5 SPI Interrupt Enable Clear register

The INTENCLR register is used to clear interrupt enable bits in the INTENSET register.

Table 395. SPI Interrupt Enable clear register (INTENCLR, offset 0x410) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved. Read value is undefined, only zero should be written.	-
4	SSAEN	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
5	SSDEN	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
7:6	-	Reserved. Read value is undefined, only zero should be written.	-
8	MSTIDLE	Writing 1 clears the corresponding bit in the INTENSET register.	0x0
31:9	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.6 SPI Divider register

The DIV register determines the clock used by the SPI in master mode.

For details on clocking, see Section 25.7.4 “Clocking and data rates”.

Table 396. SPI Divider register (DIV, offset 0x424) bit description

Bit	Symbol	Description	Reset Value
15:0	DIVVAL	Rate divider value. Specifies how the Flexcomm Interface clock (FCLK) is divided to produce the SPI clock rate in master mode. DIVVAL is -1 encoded such that the value 0 results in FCLK/1, the value 1 results in FCLK/2, up to the maximum possible divide value of 0xFFFF, which results in FCLK/65536.	0x0
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.7 SPI Interrupt Status register

The read-only INTSTAT register provides a view of those interrupt flags that are currently enabled. This can simplify software handling of interrupts. See Table 393 for detailed descriptions of the interrupt flags.

Table 397. SPI Interrupt Status register (INTSTAT, offset 0x428) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved. Read value is undefined, only zero should be written.	-
4	SSA	Slave Select Assert.	0x0
5	SSD	Slave Select Deassert.	0x0
7:6	-	Reserved. Read value is undefined, only zero should be written.	-
8	MSTIDLE	Master Idle status flag.	0x0
31:9	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.8 FIFO Configuration register

This register configures FIFO usage. A peripheral function within the Flexcomm Interface must be selected prior to configuring the FIFO.

Table 398. FIFO Configuration register (FIFOCFG - offset 0xE00) bit description

Bit	Symbol	Value	Description	Reset value	Access
0	ENABLETX		Enable the transmit FIFO.	0x0	R/W
		0	The transmit FIFO is not enabled.		
		1	The transmit FIFO is enabled.		
1	ENABLERX		Enable the receive FIFO.	0x0	R/W
		0	The receive FIFO is not enabled.		
		1	The receive FIFO is enabled.		
3:2	-	-	Reserved. Read value is undefined, only zero should be written.	-	-
5:4	SIZE		FIFO size configuration. This is a read-only field. 0x1 = FIFO is configured as 8 entries of 16 bits. 0x0, 0x2, 0x3 = not applicable to SPI.	-	RO
11:6	-	-	Reserved. Read value is undefined, only zero should be written.	-	-
12	DMATX		DMA configuration for transmit.	0x0	R/W
		0	DMA is not used for the transmit function.		
		1	Generate a DMA request for the transmit function if the FIFO is not full. Generally, data interrupts would be disabled if DMA is enabled.		
13	DMARX		DMA configuration for receive.	0x0	R/W
		0	DMA is not used for the receive function.		
		1	Generate a DMA request for the receive function if the FIFO is not empty. Generally, data interrupts would be disabled if DMA is enabled.		
14	WAKETX		Wake-up for transmit FIFO level. This allows the device to be woken from reduced power modes (up to deep-sleep, as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion.See Section 6.5.54 "Hardware Wake-up control register".	0x0	R/W
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the transmit FIFO level reaches the value specified by TXLVL in FIFOTRIG, even when the TXLVL interrupt is not enabled.		
15	WAKERX		Wake-up for receive FIFO level. This allows the device to be woken from reduced power modes (up to deep-sleep, as long as the peripheral function works in that power mode) without enabling the RXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion. See Section 6.5.54 "Hardware Wake-up control register".	0x0	R/W
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the receive FIFO level reaches the value specified by RXLVL in FIFOTRIG, even when the RXLVL interrupt is not enabled.		

Table 398. FIFO Configuration register (FIFOCFG - offset 0xE00) bit description

Bit	Symbol	Value	Description	Reset value	Access
16	EMPTYTX	-	Empty command for the transmit FIFO. When a 1 is written to this bit, the TX FIFO is emptied.	-	WO
17	EMPTYRX	-	Empty command for the receive FIFO. When a 1 is written to this bit, the RX FIFO is emptied.	-	WO
31:18	-	-	Reserved. Read value is undefined, only zero should be written.	-	-

25.6.9 FIFO status register

This register provides status information for the FIFO and also indicates an interrupt from the peripheral function.

Table 399. FIFO status register (FIFOSTAT - offset 0xE04) bit description

Bit	Symbol	Description	Reset value	Access
0	TXERR	TX FIFO error. Will be set if a transmit FIFO error occurs. This could be an overflow caused by pushing data into a full FIFO, or by an underflow if the FIFO is empty when data is needed. Cleared by writing a 1 to this bit.	0x0	R/W1C
1	RXERR	RX FIFO error. Will be set if a receive FIFO overflow occurs, caused by software or DMA not emptying the FIFO fast enough. Cleared by writing a 1 to this bit.	0x0	R/W1C
2	-	Reserved. Read value is undefined, only zero should be written.	-	
3	PERINT	Peripheral interrupt. When 1, this indicates that the peripheral function has asserted an interrupt. The details can be found by reading the peripheral's STAT register.	0x0	RO
4	TXEMPTY	Transmit FIFO empty. When 1, the transmit FIFO is empty. The peripheral may still be processing the last piece of data.	0x1	RO
5	TXNOTFULL	Transmit FIFO not full. When 1, the transmit FIFO is not full, so more data can be written. When 0, the transmit FIFO is full and another write would cause it to overflow.	0x1	RO
6	RXNOTEMPTY	Receive FIFO not empty. When 1, the receive FIFO is not empty, so data can be read. When 0, the receive FIFO is empty.	0x0	RO
7	RXFULL	Receive FIFO full. When 1, the receive FIFO is full. Data needs to be read out to prevent the peripheral from causing an overflow.	0x0	RO
12:8	TXLVL	Transmit FIFO current level. A 0 means the TX FIFO is currently empty, and the TXEMPTY and TXNOTFULL flags will be 1. Other values tell how much data is actually in the TX FIFO at the point where the read occurs. If the TX FIFO is full, the TXEMPTY and TXNOTFULL flags will be 0.	0x0	RO
15:13	-	Reserved. Read value is undefined, only zero should be written.	-	
20:16	RXLVL	Receive FIFO current level. A 0 means the RX FIFO is currently empty, and the RXFULL and RXNOTEMPTY flags will be 0. Other values tell how much data is actually in the RX FIFO at the point where the read occurs. If the RX FIFO is full, the RXFULL and RXNOTEMPTY flags will be 1.	0x0	RO
31:21	-	Reserved. Read value is undefined, only zero should be written.	-	-

25.6.10 FIFO trigger settings register

This register allows selecting when FIFO-level related interrupts occur.

Table 400. FIFO trigger settings register (FIFOTRIG - offset 0xE08) bit description

Bit	Symbol	Value	Description	Reset value
0	TXLVLENA		Transmit FIFO level trigger enable. The FIFO level trigger will cause an interrupt if enabled in FIFOINTENSET. This field is not used for DMA requests (see DMATX in FIFOCFG).	0x0
		0	Transmit FIFO level does not generate a FIFO level trigger.	
		1	An interrupt will be generated if the transmit FIFO level reaches the value specified by the TXLVL field in this register.	
1	RXLVLENA		Receive FIFO level trigger enable. This trigger will become an interrupt if enabled in FIFOINTENSET. This field is not used for DMA requests (see DMARX in FIFOCFG).	0x0
		0	Receive FIFO level does not generate a FIFO level trigger.	
		1	An interrupt will be generated if the receive FIFO level reaches the value specified by the RXLVL field in this register.	
7:2	-	-	Reserved. Read value is undefined, only zero should be written.	-
11:8	TXLVL		Transmit FIFO level trigger point. This field is used only when TXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode See Section 6.5.54 “Hardware Wake-up control register”. 0 = generate an interrupt when the TX FIFO becomes empty. 1 = generate an interrupt when the TX FIFO level decreases to one entry. ... 7 = generate an interrupt when the TX FIFO level decreases to 7 entries (is no longer full).	0x0
15:12	-	-	Reserved. Read value is undefined, only zero should be written.	-
19:16	RXLVL		Receive FIFO level trigger point. The RX FIFO level is checked when a new piece of data is received. This field is used only when RXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode See Section 6.5.54 “Hardware Wake-up control register”. 0 = generate an interrupt when the RX FIFO has one entry (is no longer empty). 1 = generate an interrupt when the RX FIFO has two entries. ... 7 = generate an interrupt when the RX FIFO increases to 8 entries (has become full).	0x0
31:20	-	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.11 FIFO interrupt enable set and read

This register is used to enable various interrupt sources. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in this register causes those bits to be set. The FIFOINTENCLR register is used to clear bits in this register.

Table 401. FIFO interrupt enable set and read register (FIFOINTENSET - offset 0xE10) bit description

Bit	Symbol	Value	Description	Reset value
0	TXERR		Determines whether an interrupt occurs when a transmit error occurs, based on the TXERR flag in the FIFOSTAT register.	0x0
		0	No interrupt will be generated for a transmit error.	
		1	An interrupt will be generated when a transmit error occurs.	
1	RXERR		Determines whether an interrupt occurs when a receive error occurs, based on the RXERR flag in the FIFOSTAT register.	0x0
		0	No interrupt will be generated for a receive error.	
		1	An interrupt will be generated when a receive error occurs.	
2	TXLVL		Determines whether an interrupt occurs when a the transmit FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0x0
		0	No interrupt will be generated based on the TX FIFO level.	
		1	If TXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the TX FIFO level decreases to the level specified by TXLVL in the FIFOTRIG register.	
3	RXLVL		Determines whether an interrupt occurs when a the receive FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0x0
		0	No interrupt will be generated based on the RX FIFO level.	
		1	If RXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the when the RX FIFO level increases to the level specified by RXLVL in the FIFOTRIG register.	
31:4	-	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.12 FIFO interrupt enable clear and read

The FIFOINTENCLR register is used to clear interrupt enable bits in FIFOINTENSET. The complete set of interrupt enables may also be read from this register as well as FIFOINTENSET.

Table 402. FIFO interrupt enable clear and read (FIFOINTENCLR - offset 0xE14) bit description

Bit	Symbol	Description	Reset value
0	TXERR	Writing a one to this bit disables the TXERR interrupt.	0x0
1	RXERR	Writing a one to this bit disables the RXERR interrupt.	0x0
2	TXLVL	Writing a one to this bit disables the interrupt caused by the transmit FIFO reaching the level specified by the TXLVL field in the FIFOTRIG register.	0x0
3	RXLVL	Writing a one to this bit disables the interrupt caused by the receive FIFO reaching the level specified by the RXLVL field in the FIFOTRIG register.	0x0
31:4	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.13 FIFO interrupt status register

The read-only FIFOINTSTAT register provides a view of those interrupt flags that are both pending and currently enabled. This can simplify software handling of interrupts. Refer to the descriptions of interrupts in Section 25.6.9 and Section 25.6.10 for details.

Table 403. FIFO interrupt status register (FIFOINTSTAT - offset 0xE18) bit description

Bit	Symbol	Description	Reset value
0	TXERR	TX FIFO error.	0x0
1	RXERR	RX FIFO error.	0x0
2	TXLVL	Transmit FIFO level interrupt.	0x0
3	RXLVL	Receive FIFO level interrupt.	0x0
4	PERINT	Peripheral interrupt.	0x0
31:5	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.14 FIFO write data register

The FIFOWR register is used to write values to be transmitted to the FIFO.

FIFOWR provides the possibility of altering some SPI controls at the same time as sending new data. For example, this can allow a series of SPI transactions involving multiple slaves to be stored in a DMA buffer and sent automatically.These added fields are described for bits 16 through 27 below.

Each FIFO entry holds data and associated control bits. Before data and control bits are pushed into the FIFO, the control bit settings can be modified. Halfword writes to just the control bits (offset 0xE22) doesn't push anything into the FIFO. A zero written to the upper halfword will not modify the control settings. Non-zero writes to it will modify all the control bits. Note that this is a write only register. Do not read-modify-write the register.

Byte, halfword or word writes to FIFOWR will push the data and control bits into the FIFO. Word writes with the upper halfword of zero, byte writes or halfword writes to FIFOWR will push the data and the current control bits, into the FIFO. Word writes with a non-zero upper halfword will modify the control bits before pushing them onto the stack.

Table 404. FIFO write data register (FIFOWR - offset 0xE20) bit description

Bit	Symbol	Value	Description	Reset value
15:0	TXDATA	-	Transmit data to the FIFO.	-
16	TXSSEL0_N		Transmit Slave Select. This field asserts SSEL0 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL0 pin is configured by bits in the CFG register.	-
		0	SSEL0 asserted.	
		1	SSEL0 not asserted.	
17	TXSSEL1_N		Transmit Slave Select. This field asserts SSEL1 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL1 pin is configured by bits in the CFG register.	-
		0	SSEL1 asserted.	
		1	SSEL1 not asserted.	

Table 404. FIFO write data register (FIFOWR - offset 0xE20) bit description

Bit	Symbol	Value	Description	Reset value
18	TXSSEL2_N		Transmit Slave Select. This field asserts SSEL2 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL2 pin is configured by bits in the CFG register.	-
		0	SSEL2 asserted.	
		1	SSEL2 not asserted.	
19	TXSSEL3_N		Transmit Slave Select. This field asserts SSEL3 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL3 pin is configured by bits in the CFG register.	-
		0	SSEL3 asserted.	
		1	SSEL3 not asserted.	
20	EOT		End of Transfer. The asserted SSEL will be deasserted at the end of a transfer, and remain so for at least the time specified by the Transfer_delay value in the DLY register.	-
		0	SSEL not deasserted. This piece of data is not treated as the end of a transfer. SSEL will not be deasserted at the end of this data.	
		1	SSEL deasserted. This piece of data is treated as the end of a transfer. SSEL will be deasserted at the end of this piece of data.	
21	EOF		End of Frame. Between frames, a delay may be inserted, as defined by the FRAME_DELAY value in the DLY register. The end of a frame may not be particularly meaningful if the FRAME_DELAY value = 0. This control can be used as part of the support for frame lengths greater than 16 bits.	-
		0	Data not EOF. This piece of data transmitted is not treated as the end of a frame.	
		1	Data EOF. This piece of data is treated as the end of a frame, causing the FRAME_DELAY time to be inserted before subsequent data is transmitted.	
22	RXIGNORE		Receive Ignore. This allows data to be transmitted using the SPI without the need to read unneeded data from the receiver. Setting this bit simplifies the transmit process and can be used with the DMA.	-
		0	Read received data. Received data must be read first and then the RxData should be written to allow transmission to progress for non-DMA cases. SPI transmit will halt when the receive data FIFO is full. In slave mode, an overrun error will occur if received data is not read before new data is received.	
		1	Ignore received data. Received data is ignored, allowing transmission without reading unneeded received data. No receiver flags are generated.	
23	-	-	Reserved. Read value is undefined, only zero should be written.	-
27:24	LEN		Data Length. Specifies the data length from 4 to 16 bits. Note that transfer lengths greater than 16 bits are supported by implementing multiple sequential transmits. 0x0-2 = Reserved 0x3 = Data transfer is 4 bits in length. 0x4 = Data transfer is 5 bits in length. ... 0xF = Data transfer is 16 bits in length.	-
31:28	-	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.15 FIFO read data register

The FIFORD register is used to read values that have been received by the FIFO.

Table 405. FIFO read data register (FIFORD - offset 0xE30) bit description

Bit	Symbol	Description	Reset value
15:0	RXDATA	Received data from the FIFO.	-
16	RXSSEL0_N	Slave Select for receive. This field allows the state of the SSEL0 pin to be saved along with received data. The value will reflect the SSEL0 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.	-
17	RXSSEL1_N	Slave Select for receive. This field allows the state of the SSEL1 pin to be saved along with received data. The value will reflect the SSEL1 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.	-
18	RXSSEL2_N	Slave Select for receive. This field allows the state of the SSEL2 pin to be saved along with received data. The value will reflect the SSEL2 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.	-
19	RXSSEL3_N	Slave Select for receive. This field allows the state of the SSEL3 pin to be saved along with received data. The value will reflect the SSEL3 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.	-
20	SOT	Start of Transfer flag. This flag will be 1 if this is the first data after the SSELs went from deasserted to asserted (i.e., any previous transfer has ended). This information can be used to identify the first piece of data in cases where the transfer length is greater than 16 bits.	-
31:21	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.16 FIFO data read with no FIFO pop

This register acts in exactly the same way as FIFORD, except that it supplies data from the top of the FIFO without popping the FIFO (i.e. leaving the FIFO state unchanged). This could be used to allow system software to observe incoming data without interfering with the peripheral driver.

Table 406. FIFO data read with no FIFO pop (FIFORDNOPOP - offset 0xE40) bit description

Bit	Symbol	Description	Reset value
15:0	RXDATA	Received data from the FIFO.	-
16	RXSSEL0_N	Slave Select for receive.	-
17	RXSSEL1_N	Slave Select for receive.	-
18	RXSSEL2_N	Slave Select for receive.	-
19	RXSSEL3_N	Slave Select for receive.	-
20	SOT	Start of Transfer flag.	-
31:21	-	Reserved. Read value is undefined, only zero should be written.	-

25.6.17 Module identification register

The ID register identifies the type and revision of the SPI module. A generic SW driver can make use of this information register to implement module type or revision specific behavior.

Table 407. Module identification register (ID - offset 0xFFC) bit description

Bit	Symbol	Description	Reset Value
7:0	APERTURE	Aperture: encoded as (aperture size/4K) -1, so 0x00 means a 4K aperture.	0x0
11:8	MINOR_REV	Minor revision of module implementation, starting at 0. Software compatibility is expected between minor revisions.	-
15:12	MAJOR_REV	Major revision of module implementation, starting at 0. There may not be software compatibility between major revisions.	-
31:16	ID	Unique module identifier for this IP block.	0xE020

25.7 Functional description

25.7.1 AHB bus access

With the exception of the FIFOWR register, the bus interface to the SPI registers contained in the Flexcomm Interface support only word writes. Byte and halfword writes are not supported in conjunction with the SPI function for those registers.

The FIFOWR register also supports byte and halfword (data only) writes in order to allow writing FIFO data without affecting the SPI control fields above bit 15 (see Section 25.6.14 “FIFO write data register”).

25.7.2 Operating modes: clock and phase selection

SPI interfaces typically allow configuration of clock phase and polarity. These are sometimes referred to as numbered SPI modes, as described in Table 408 and shown in Figure 59. CPOL and CPHA are configured by bits in the CFG register (Section 25.6.1).

Table 408: SPI mode summary

CPOL	CPHA	SPI Mode	Description	SCK rest state	SCK data change edge	SCK data sample edge
0	0	0	The SPI captures serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is changed on the following edge.	low	falling	rising
0	1	1	The SPI changes serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is captured on the following edge.	low	rising	falling
1	0	2	Same as mode 0 with SCK inverted.	high	rising	falling
1	1	3	Same as mode 1 with SCK inverted.	high	falling	rising

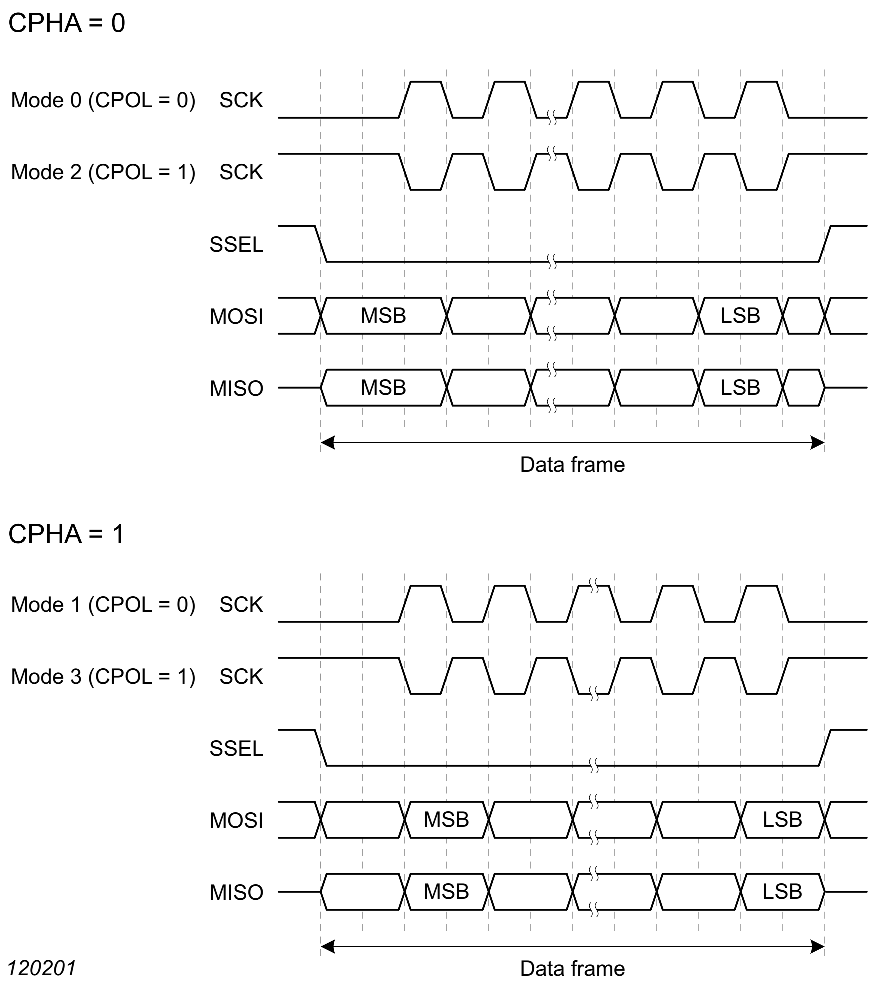


Fig 59. Basic SPI operating modes

25.7.3 Frame delays

Several delays can be specified for SPI frames. These include:

- Pre_delay: delay after SSEL is asserted before data clocking begins
- Post_delay: delay at the end of a data frame before SSEL is deasserted
- Frame_delay: delay between data frames when SSEL is not deasserted
- Transfer_delay: minimum duration of SSEL in the deasserted state between transfers

25.7.3.1 Pre_delay and Post_delay

Pre_delay and Post_delay are illustrated by the examples in Figure 60. The Pre_delay value controls the amount of time between SSEL being asserted and the beginning of the subsequent data frame. The Post_delay value controls the amount of time between the end of a data frame and the deassertion of SSEL.

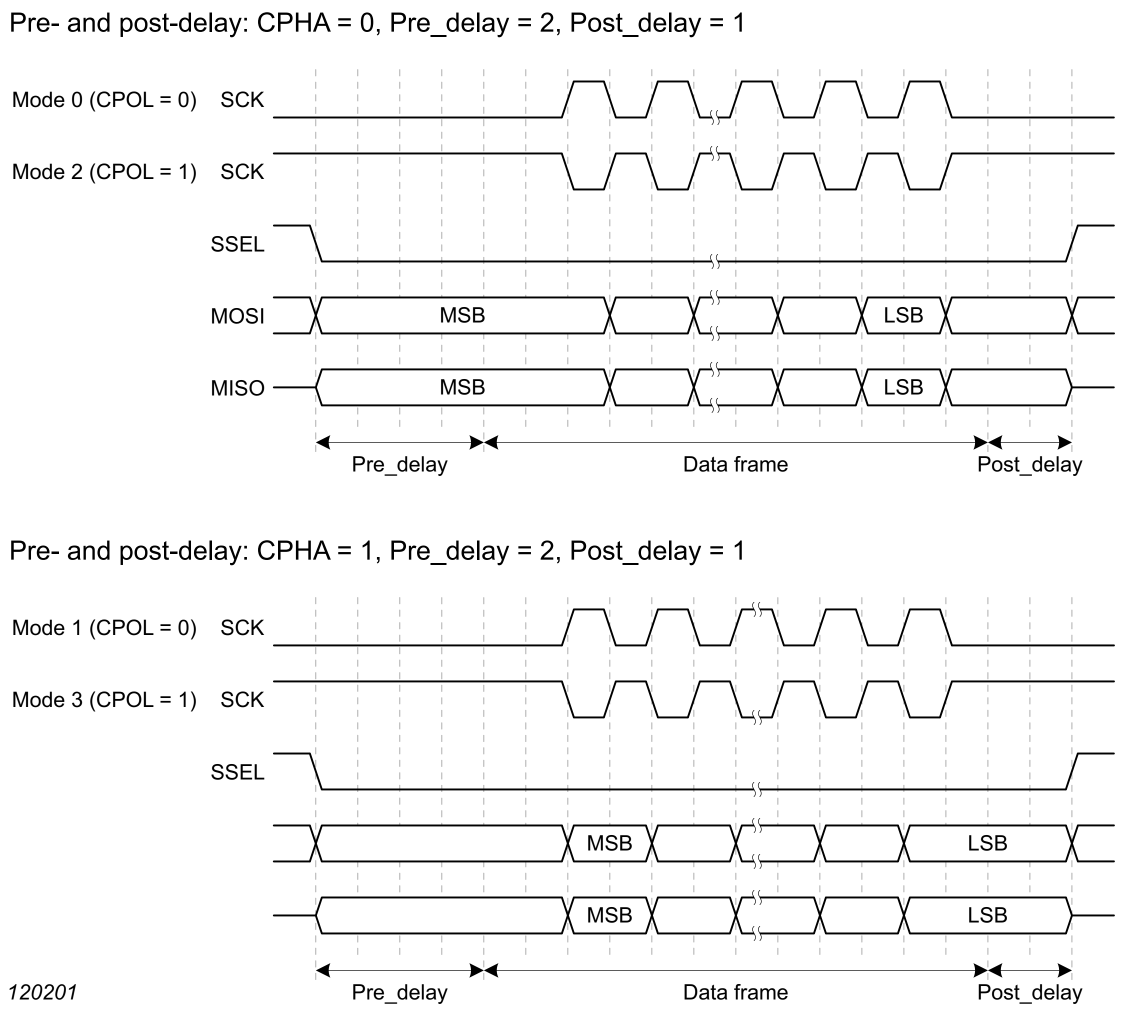


Fig 60. Pre_delay and Post_delay

25.7.3.2 Frame_delay

The Frame_delay value controls the amount of time at the end of each frame. This delay is inserted when the EOF bit = 1. Frame_delay is illustrated by the examples in Figure 61. Note that frame boundaries occur only where specified. This is because frame lengths can be any size, involving multiple data writes. See Section 25.7.7 for more information.

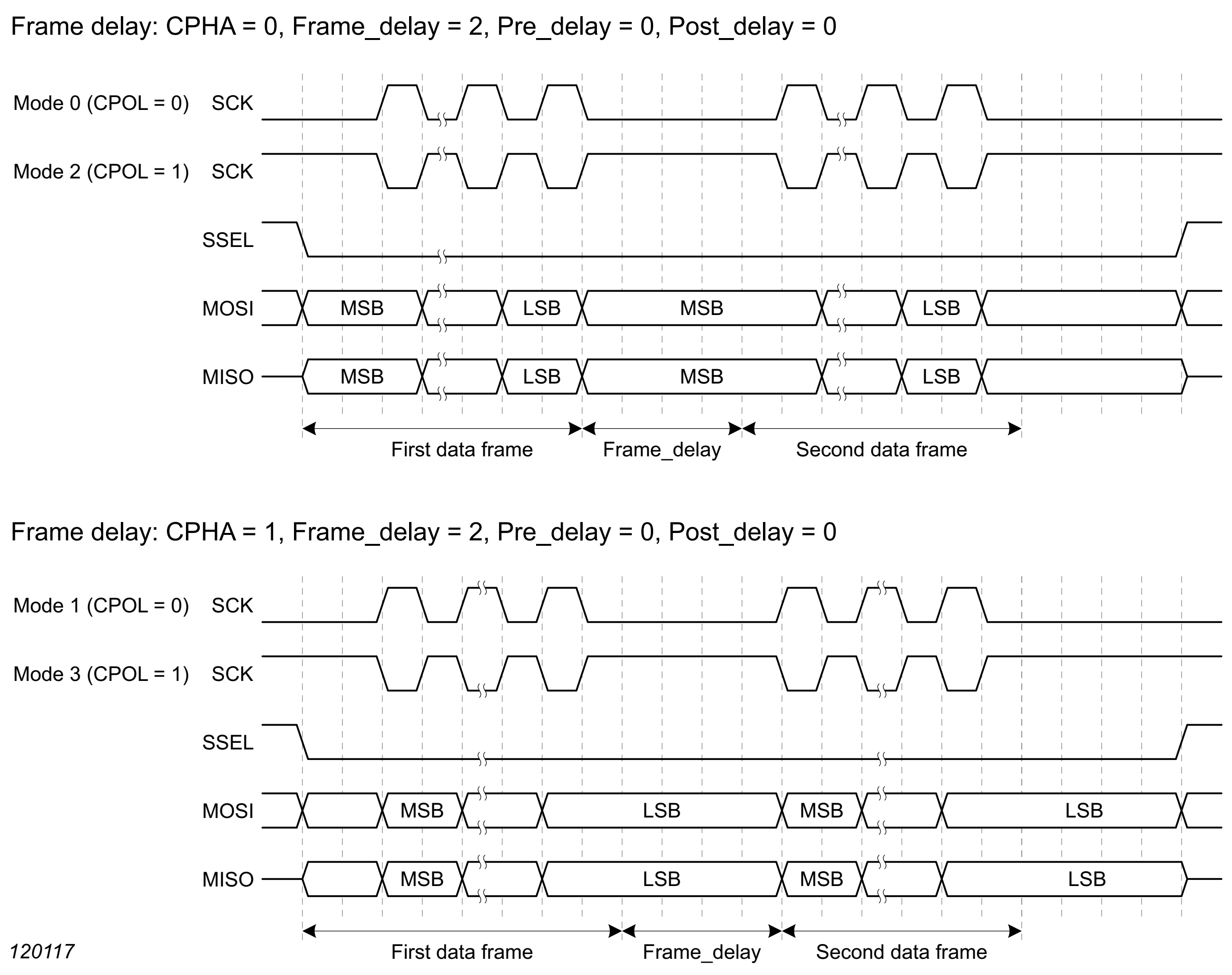


Fig 61. Frame_delay

25.7.3.3 Transfer_delay

The Transfer_delay value controls the minimum amount of time that SSEL is deasserted between transfers, because the EOT bit = 1. When Transfer_delay = 0, SSEL may be deasserted for a minimum of one SPI clock time. Transfer_delay is illustrated by the examples in Figure 62.

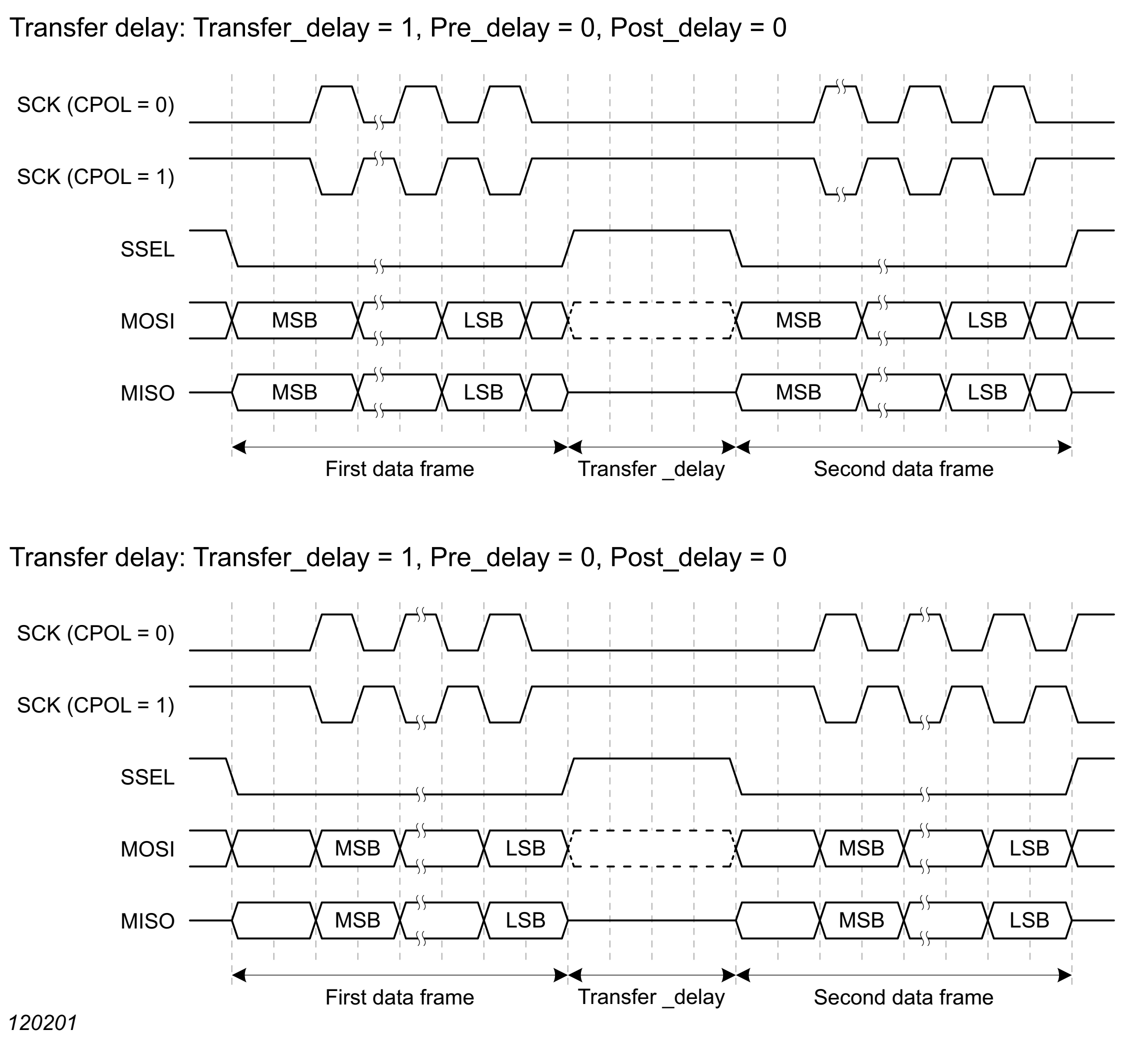


Fig 62. Transfer_delay

25.7.4 Clocking and data rates

In order to use the SPI, clocking details must be defined. This includes configuring the system clock and selection of the clock divider value in DIV. See Figure 9 “Clock generation”.

25.7.4.1 Data rate calculations

The SPI interface is designed to operate asynchronously from any on-chip clocks, and without the need for overclocking.

In slave mode, this means that the SCK from the external master is used directly to run the transmit and receive shift registers and other logic.

In master mode, the SPI rate clock produced by the SPI clock divider is used directly as the outgoing SCK.

The SPI clock divider is an integer divider. The SPI in master mode can be set to run at the same speed as the selected FCLK, or at lower integer divide rates. The SPI rate will be = FCLK of Flexcomm Interface n / DIVVAL.

In slave mode, the clock is taken from the SCK input and the SPI clock divider is not used.

25.7.5 Slave select

The SPI block provides for four Slave Select inputs in slave mode or outputs in master mode. Each SSEL can be set for normal polarity (active low), or can be inverted (active high). Representation of the 4 SSELs in a register is always active low. If an SSEL is inverted, this is done as the signal leaves/enters the SPI block.

In slave mode, **any** asserted SSEL that is connected to a pin will activate the SPI. In master mode, all SSELs that are connected to a pin will be output as defined in the SPI registers. In the latter case, the SSELs could potentially be decoded externally in order to address more than four slave devices. Note that at least one SSEL is asserted when data is transferred in master mode.

In master mode, Slave Selects come from the TXSSEL bits in the FIFOWR register. In slave mode, the state of all four SSELs is saved along with received data in the RXSSEL_N field of the FIFORD register.

25.7.6 DMA operation

A DMA request is provided for each SPI direction, and can be used in lieu of interrupts for transferring data by configuring the DMA controller appropriately. The DMA controller provides an acknowledgement signal that clears the related request when it completes handling that request.

The transmitter DMA request is asserted when Tx DMA is enabled and the transmitter can accept more data.

The receiver DMA request is asserted when Rx DMA is enabled and received data is available to be read.

25.7.6.1 DMA master mode End-Of-Transfer

When using polled or interrupt mode to transfer data in master mode, the transition to end-of-transfer status (drive SSEL inactive) is straightforward. The EOT bit of the FIFOWR control bits would be set just before or along with the writing of the last data to be sent.

When using the DMA in master mode, the end-of-transfer status (drive SSEL inactive) can be generated in a number of ways:

1. Using DMA interrupt and a second DMA transfer:

   To use only 8 or 16 bit wide DMA transfers for all the data, a second DMA transfer can be used to terminate the transfer (drive SSEL inactive).

   The transfer would be started by setting the control bits and then initiating the DMA transfer of all but the last byte/halfword of data. The DMA completion interrupt function must modify the control bits to set EOT and then set-up DMA to send the last data.
2. Using DMA and SPI interrupts (or background SPI status polling):

   To use only one 8 or 16 bit wide DMA transfer for all the data, two interrupts would be required to properly terminate the transfer (drive SSEL inactive).

   The SPI Tx DMA completion interrupt function sets the TXLVL field in the SPI FIFOTRIG register to 0 and sets the TXLVL interrupt enable bit in the FIFOINTENSET register.

   The interrupt function handling the SPI TXLVL would set the SPI STAT register "END TRANSFER" bit, to force termination after all data output is complete.
3. Using DMA linked descriptor:

   The DMA controller provides for a linked list of DMA transfer control descriptors. The initial descriptor(s) can be used to transfer all but the last data byte/halfword. These data transfers can be done as 8 or 16 bit wide DMA operations. A final DMA descriptor, linked to the first DMA descriptor, can be used to send the last data along with control bits to the FIFOWR register. The control bits would include the setting of the EOT bit.

   Note: The DMA interrupt function cannot set the SPI Status register (STAT) END TRANSFER control bit. This may terminate the transfer while the FIFO still has data to send.
4. Using 32 bit wide DMA:

Write both data and control bits to FIFOWR for all data. The control bits for the last entry would include the setting of the EOT bit. This also allows a series of SPI transactions involving multiple slaves with one DMA operation, by changing the TXSSELn_N bits.

25.7.7 Data lengths greater than 16 bits

The SPI interface handles data frame sizes from 4 to 16 bits directly. Larger sizes can be handled by splitting data up into groups of 16 bits or less. For example, 24 bits can be supported as 2 groups of 16 bits and 8 bits or 2 groups of 12 bits, among others. Frames of any size, including greater than 32 bits, can supported in the same way.

Details of how to handle larger data widths depend somewhat on other SPI configuration options. For instance, if it is intended for Slave Selects to be deasserted between frames, then this must be suppressed when a larger frame is split into more than one part. Sending 2 groups of 12 bits with SSEL deasserted between 24-bit increments, for instance, would require changing the value of the EOF bit on alternate 12-bit frames.

25.8 Data stalls

A stall for Master transmit data can happen in modes 0 and 2 when SCK cannot be returned to the rest state until the MSB of the next data frame can be driven on MOSI. In this case, the stall happens just before the final clock edge of data if the next piece of data is not yet available.

A stall for Master receive can happen when a FIFO overflow (see RXERR in the FIFOSTAT register) would otherwise occur if the transmitter was not stalled. In modes 0 and 2, this occurs if the FIFO is full when the next piece of data is received. This stall happens one clock edge earlier than the transmitter stall.

In modes 1 and 3, the same kind of receiver stall can occur, but just before the final clock edge of the received data. Also, a transmitter stall will not happen in modes 1 and 3 because the transmitted data is complete at the point where a stall would otherwise occur, so it is not needed.

Stalls are reflected in the STAT register by the Stalled status flag, which indicates the current SPI status. The transmitter will be stalled until data is read from the receive FIFO. Use the RXIGNORE control bit setting to avoid the need to read the received data.

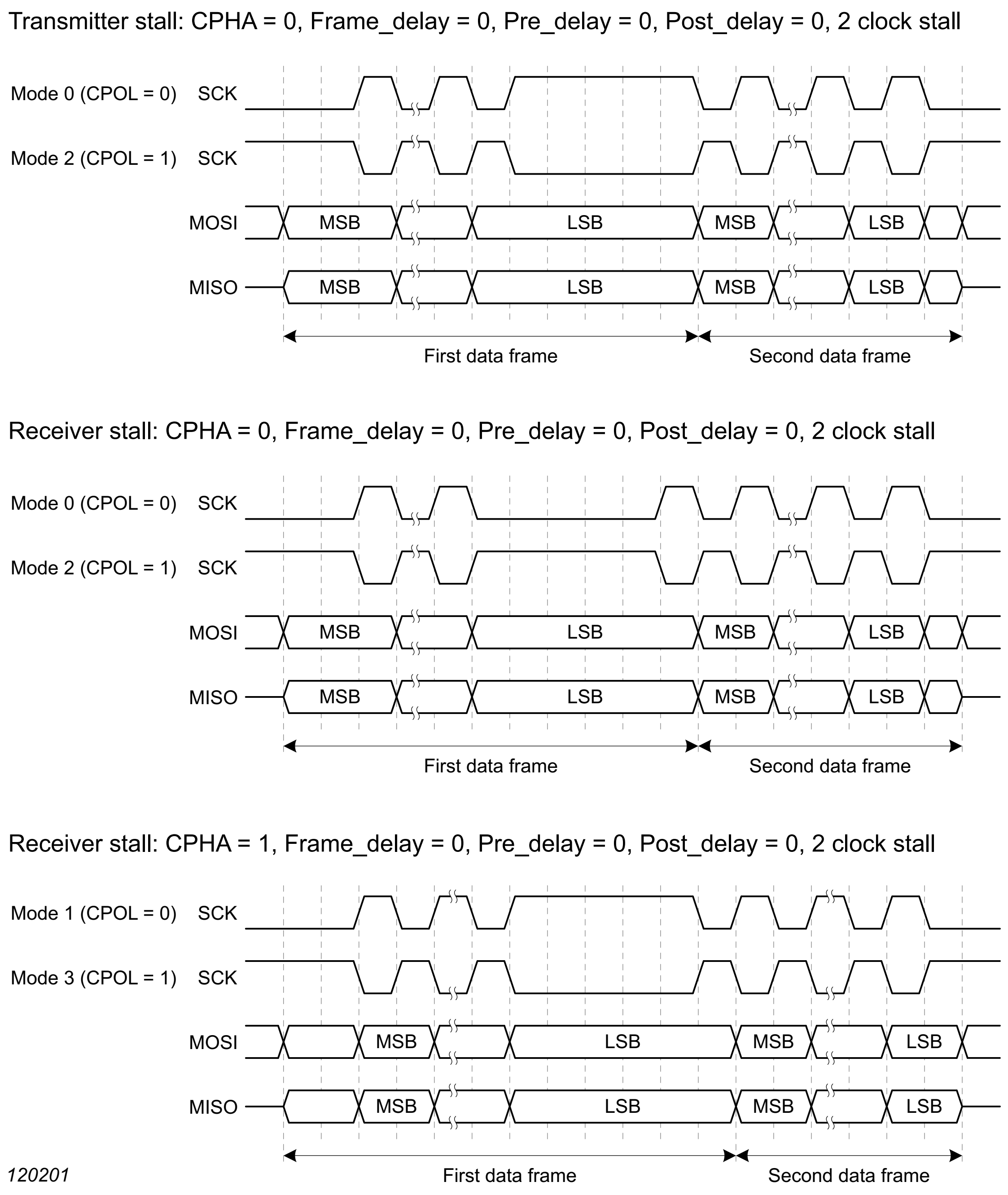


Fig 63. Examples of data stalls

UM11071

Chapter 26: LPC51U68 I^{2}C-bus interfaces

Rev. 1.1 — 17 May 2018 **User manual**

26.1 How to read this chapter

I^{2}C-bus functions are available on all LPC51U68 devices as a selectable function in each Flexcomm Interface peripheral. Up to 8 Flexcomm Interfaces are available.

26.2 Features

- Independent Master, Slave, and Monitor functions.
- Bus speeds supported:
  - Standard mode, up to 100 kbits/s.
  - Fast-mode, up to 400 kbits/s.
  - Fast-mode Plus, up to 1 Mbits/s (on pins PIO0_23 and 24 or PIO0_25 and 26 that include specific I^{2}C support).
  - High speed mode, 3.4 Mbits/s as a Slave only (on pins PIO0_23 and 24 or PIO0_25 and 26 that include specific I^{2}C support).
- Supports both Multi-master and Multi-master with Slave functions.
- Multiple I^{2}C slave addresses supported in hardware.
- One slave address can be selectively qualified with a bit mask or an address range in order to respond to multiple I^{2}C bus addresses.
- 10-bit addressing supported with software assist.
- Supports System Management Bus (SMBus).
- Separate DMA requests for Master, Slave, and Monitor functions.
- No chip clocks are required in order to receive and compare an address as a Slave, so this event can wake up the device from deep-sleep mode.
- Automatic modes optionally allow less software overhead for some use cases.

26.3 Pin description

The I^{2}C pins are fixed-pin functions and enabled through IOCON. Refer to the IOCON settings in Table 410 and in Section 9.5.2.

Table 409. I^{2}C-bus pin description

Function	Type	Pin name used in data sheet Pin Description	Description
SCL	I/O	FCn_TXD_SCL_MISO, FCn_TXD_SCL_MISO_WS, or FCn_RTS_SCL_SSEL1	I^{2}C serial clock.
SDA	I/O	FCn_RXD_SDA_MOSI, FCn_RXD_SDA_MOSI_DATA, or FCn_CTS_SDA_SSEL0	I^{2}C serial data.

Table 410: Suggested I^{2}C pin settings

IOCON bit(s)	Type D pin	Type A pin	Type I pin
10	OD: Set to 1 for simulated open-drain output.	Same as type D.	I2CFILTER: 0 for Fast / Standard mode I^{2}C. 1 for Fast Mode Plus or High Speed slave.
9	SLEW: Set to 0.	Not used, set to 0.	I2CDRIVE: 0 for Fast / Standard mode I^{2}C. 1 for Fast Mode Plus or High Speed slave.
8	FILTEROFF: Generally set to 1.	Same as type D.	Same as type D.
7	DIGIMODE: Set to 1.	Same as type D.	Same as type D.
6	INVERT: Set to 0.	Same as type D.	Same as type D.
5	Not used, set to 0.	Same as type D.	I2CSLEW: Set to 0.
4:3	MODE: Set to 0 (pull-down/pull-up resistor disabled).	Same as type D.	Not used, set to 0.
2:0	FUNC: The function will be “SCL” or “SDA”.	Same as type D.	FUNC: The function will be “SCL” or “SDA”.
General comment	A reasonable choice for I^{2}C at or below 400 kHz.	Same as type D.	Recommended for I^{2}C operation above 400 kHz.

26.4 Basic configuration

Configure the I^{2}C and related clocks as follows:

- If needed, use the PRESETCTRL1 register (Table 109) to reset the Flexcomm Interface that is about to have a specific peripheral function selected.
- Select the desired Flexcomm Interface function by writing to the PSELID register of the related Flexcomm Interface (Section 23.7.1). Note that any selection that has been made will be cleared if the Flexcomm Interface itself is reset via the PRESETCTRL1 register.
- Configure the I^{2}C for the desired functions:
  - In the AHBCLKCTRL1 register (Table 116), set the appropriate bit for the related Flexcomm Interface in order to enable the clock to the register interface.
  - Enable or disable the related Flexcomm Interface interrupt in the NVIC (see Table 79).
  - Configure the related Flexcomm Interface pin functions via IOCON, see Chapter 9.
  - Configure the I^{2}C clock and data rate. This includes the CLKDIV register for both master and slave modes, and MSTTIME for master mode. Also see Section 26.6.6 and Section 26.7.2.

    Remark: The Flexcomm Interface function clock frequency should not be above 48 MHz.

Remark: While the I^{2}C function is incorporated into the Flexcomm Interface, it does not make use of the Flexcomm Interface FIFO.

26.4.1 I^{2}C transmit/receive in master mode

In this example, Flexcomm Interface 1 is configured as an I^{2}C master. The master sends 8 bits to the slave and then receives 8 bits from the slave.

If specialized I^{2}C pins are used (PIO0_23 through PIO0_26), the pins should be configured as required for the I^{2}C-bus mode that will be used (SM, FM, FM+, HS) via the IOCON block. If these or standard pins are used, they should be configured as described in Section 9.5.2.

The transmission of the address and data bits is controlled by the state of the MSTPENDING status bit. Whenever the status is Master pending, the master can read or write to the MSTDAT register and go to the next step of the transmission protocol by writing to the MSTCTL register.

Configure the I^{2}C bit rate:

- Select a source for the Flexcomm Interface 1 clock that will allow for the desired I^{2}C-bus rate. Divide the clock as needed, see Table 423.
- Further divide the source clock if needed using the CLKDIV register (Section 26.6.6).
- Set the SCL high and low times to complete the bus rate setup. See Section 26.6.9.

26.4.1.1 Master write to slave

Configure Flexcomm Interface 1 as I^{2}C interface, see Chapter 23.

Configure the I^{2}C as a master: set the MSTEN bit to 1 in the CFG register. See Table 416.

Write data to the slave:

1. Write the slave address with the $\overline{\text{RW}}$ bit set to 0 to the Master data register MSTDAT. See Table 427.
2. Start the transmission by setting the MSTSTART bit to 1 in the Master control register. See Table 425. The following happens:
   – The pending status is cleared and the I^{2}C-bus is busy.
   – The I^{2}C master sends the start bit and address with the $\overline{\text{RW}}$ bit to the slave.
3. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
4. Write 8 bits of data to the MSTDAT register.
5. Continue with the transmission of data by setting the MSTCONT bit to 1 in the Master control register. See Table 425. The following happens:
   – The pending status is cleared and the I^{2}C-bus is busy.
   – The I^{2}C master sends the data bits to the slave address.
6. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
7. Stop the transmission by setting the MSTSTOP bit to 1 in the Master control register. See Table 425.

Table 411. Code example

Master write to slave

```
//Master write 1 byte to slave. Address 0x23, Data 0xdd. Polling mode.
I2C->CFG = I2C_CFG_MSTEN;
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
I2C->MSTDAT = (0x23 << 1) | 0; // address and 0 for RWn bit
I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_TX) abort();
I2C->MSTDAT = 0xdd; // send data
I2C->MSTCTL = I2C_MSTCTL_MSTCONTINUE; // continue transaction
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_TX) abort();
I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // send stop
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
```

26.4.1.2 Master read from slave

Configure Flexcomm Interface 1 as I^{2}C interface, see Chapter 23.

Configure the I^{2}C as a master: set the MSTEN bit to 1 in the CFG register. See Table 416.

Read data from the slave:

1. Write the slave address with the $\overline{\text{RW}}$ bit set to 1 to the Master data register MSTDAT. See Table 427.
2. Start the transmission by setting the MSTSTART bit to 1 in the Master control register. See Table 425. The following happens:
   – The pending status is cleared and the I^{2}C-bus is busy.
   – The I^{2}C master sends the start bit and address with the $\overline{\text{RW}}$ bit to the slave.
   – The slave sends 8 bit of data.
3. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
4. Read 8 bits of data from the MSTDAT register.
5. Stop the transmission by setting the MSTSTOP bit to 1 in the Master control register. See Table 425.

Table 412. Code example

Master read from slave

```
// Master read 1 byte from slave. Address 0x23. Polling mode. No error checking.
uint8_t data;
I2C->CFG = I2C_CFG_MSTEN;
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
I2C->MSTDAT = (0x23 << 1) | 1; // address and 1 for RWn bit
I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_RX) abort();
data = I2C->MSTDAT; // read data
if(data != 0xdd) abort();
I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // send stop
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
```

26.4.2 I^{2}C receive/transmit in slave mode

In this example, Flexcomm Interface 1 is configured as an I^{2}C slave. The slave receives 8 bits from the master and then sends 8 bits to the master. The SCL and SDA functions must be enabled on pins PIO0_22 and PIO0_23 through IOCON. See Section 9.5.2.

The pins should be configured as required for the I^{2}C-bus mode that will be used (SM, FM, FM+, HS) via the IOCON block. See Section 9.5.2.

The transmission of the address and data bits is controlled by the state of the SLVPENDING status bit. Whenever the status is Slave pending, the slave can acknowledge (“ack”) or send or receive an address and data. The received data or the data to be sent to the master are available in the SLVDAT register. After sending and receiving data, continue to the next step of the transmission protocol by writing to the SLVCTL register.

26.4.2.1 Slave read from master

Configure Flexcomm Interface 1 as I^{2}C interface, see Chapter 23.

Configure the I^{2}C as a slave with address x:

- Set the SLVEN bit to 1 in the CFG register. See Table 416.
- Write the slave address x to the address 0 match register. See Table 430.

Read data from the master:

1. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
2. Acknowledge (“ack”) the address by setting SLVCONTINUE = 1 in the slave control register. See Table 428.
3. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
4. Read 8 bits of data from the SLVDAT register. See Table 429.
5. Acknowledge (“ack”) the data by setting SLVCONTINUE = 1 in the slave control register. See Table 428.

Table 413. Code example

Slave read from master

`//Slave read 1 byte from master. Address 0x23. Polling mode.`
`uint8_t data;`
`I2C->SLVADR0 = 0x23 << 1; // put address in address 0 register`
`I2C->CFG = I2C_CFG_SLVEN;`
`while(!(I2C->STAT & I2C_STAT_SLVPENDING));`
`if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort();`
`I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack address`
`while(!(I2C->STAT & I2C_STAT_SLVPENDING));`
`if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_RX) abort();`
`data = I2C->SLVDAT; // read data`
`if(data != 0xdd) abort();`
`I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack data`

26.4.2.2 Slave write to master

Configure Flexcomm Interface 1 as I^{2}C interface, see Chapter 23.

Configure the I^{2}C as a slave with address x:

- Set the SLVEN bit to 1 in the CFG register. See Table 416.
- Write the slave address x to the address 0 match register. See Table 430.

Write data to the master:

1. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
2. ACK the address by setting SLVCONTINUE = 1 in the slave control register. See Table 428.
3. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
4. Write 8 bits of data to SLVDAT register. See Table 429.
5. Continue the transaction by setting SLVCONTINUE = 1 in the slave control register. See Table 428.

Table 414. Code example

Slave write to master

`//Slave write 1 byte to master. Address 0x23, Data 0xdd. Polling mode.`
`I2C->SLVADR0 = 0x23 << 1; // put address in address 0 register`
`I2C->CFG = I2C_CFG_SLVEN;`
`while(!(I2C->STAT & I2C_STAT_SLVPENDING));`
`if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort();`
`I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack address`
`while(!(I2C->STAT & I2C_STAT_SLVPENDING));`
`if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_TX) abort();`
`I2C->SLVDAT = 0xdd; // write data`
`I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // continue transaction`

26.4.3 Configure the I²C for wake-up

In sleep mode, any activity on the I²C-bus that triggers an I²C interrupt can wake up the part, provided that the interrupt is enabled in the INTENSET register and the NVIC. As long as the Flexcomm Interface clock remains active in sleep mode, the I²C can wake up the part independently of whether the I²C interface is configured in master or slave mode.

In deep-sleep mode, the I²C clock is turned off as are all peripheral clocks. However, if the I²C is configured in slave mode and an external master on the I²C-bus provides the clock signal, the I²C interface can create an interrupt asynchronously. This interrupt, if enabled in the NVIC and in the I²C interface INTENCLR register, can then wake up the core.

26.4.3.1 Wake-up from sleep mode

- Enable the I²C interrupt in the NVIC.
- Enable the I²C wake-up event in the INTENSET register. Wake-up on any enabled interrupts is supported (see the INTENSET register). Examples are the following events:
  – Master pending
  – Change to idle state
  – Start/stop error
  – Slave pending
  – Address match (in slave mode)
  – Data available/ready

26.4.3.2 Wake-up from deep-sleep mode

- Enable the I²C interrupt in the NVIC.
- Enable the I²C interrupt in the STARTER0 register in the SYSCON block to create the interrupt signal asynchronously while the core and the peripheral are not clocked.
- Configure the I²C in slave mode.
- Enable the I²C the interrupt in the INTENCLR register which configures the interrupt as wake-up event. Examples are the following events:
  – Slave deselect
  – Slave pending (wait for read, write, or ACK)
  – Address match
  – Data available/ready for the Monitor function

26.5 General description

The architecture of the I2C-bus interface is shown in Figure 64.

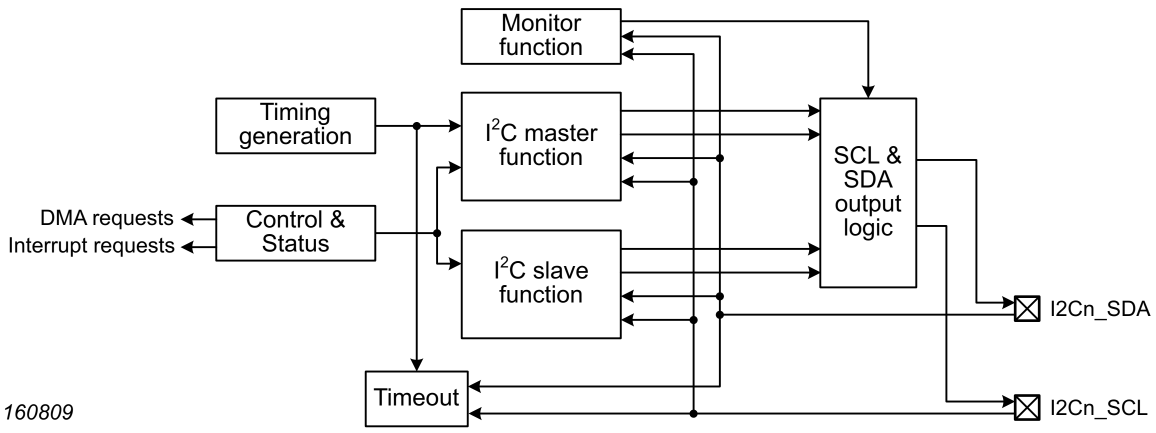


Fig 64. I2C block diagram

26.6 Register description

Address offsets are within the address space of the related Flexcomm Interface. The Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

Table 415: I^{2}C register overview

Name	Access	Offset	Description	Reset value	Section
Shared I^{2}C registers:					
CFG	R/W	0x800	Configuration for shared functions.	0x0	26.6.1
STAT	R/W	0x804	Status register for Master, Slave, and Monitor functions.	0x0801	26.6.2
INTENSET	R/W	0x808	Interrupt Enable Set and read register.	0x0	26.6.3
INTENCLR	WO	0x80C	Interrupt Enable Clear register.	-	26.6.4
TIMEOUT	R/W	0x810	Time-out value register.	0xFFFF	26.6.5
CLKDIV	R/W	0x814	Clock pre-divider for the entire I^{2}C interface. This determines what time increments are used for the MSTTIME register, and controls some timing of the Slave function.	0x0	26.6.6
INTSTAT	RO	0x818	Interrupt Status register for Master, Slave, and Monitor functions.	0x0	26.6.7
Master function registers:					
MSTCTL	R/W	0x820	Master control register.	0x0	26.6.8
MSTTIME	R/W	0x824	Master timing configuration.	0x77	26.6.9
MSTDAT	R/W	0x828	Combined Master receiver and transmitter data register.	-	26.6.10
Slave function registers:					
SLVCTL	R/W	0x840	Slave control register.	0x0	26.6.11
SLVDAT	R/W	0x844	Combined Slave receiver and transmitter data register.	-	26.6.12
SLVADR0	R/W	0x848	Slave address 0.	0x01	26.6.13
SLVADR1	R/W	0x84C	Slave address 1.	0x01	26.6.14
SLVADR2	R/W	0x850	Slave address 2.	0x01	26.6.14
SLVADR3	R/W	0x854	Slave address 3.	0x01	26.6.14
SLVQUAL0	R/W	0x858	Slave Qualification for address 0.	0x0	26.6.15
Monitor function registers:					
MONRXDAT	RO	0x880	Monitor receiver data register.	0x0	26.6.16
ID register:					
ID	RO	0xFFC	I2C module Identification. This value appears in the shared Flexcomm Interface peripheral ID register when I2C is selected.	0xE030 0000	26.6.17

26.6.1 I2C Configuration register

The CFG register contains mode settings that apply to Master, Slave, and Monitor functions.

Table 416. I^{2}C Configuration register (CFG, offset 0x800) bit description

Bit	Symbol	Value	Description	Reset Value
0	MSTEN		Master Enable. When disabled, configurations settings for the Master function are not changed, but the Master function is internally reset.	0x0
		0	Disabled. The I^{2}C Master function is disabled.	
		1	Enabled. The I^{2}C Master function is enabled.	
1	SLVEN		Slave Enable. When disabled, configurations settings for the Slave function are not changed, but the Slave function is internally reset.	0x0
		0	Disabled. The I^{2}C slave function is disabled.	
		1	Enabled. The I^{2}C slave function is enabled.	
2	MONEN		Monitor Enable. When disabled, configurations settings for the Monitor function are not changed, but the Monitor function is internally reset.	0x0
		0	Disabled. The I^{2}C Monitor function is disabled.	
		1	Enabled. The I^{2}C Monitor function is enabled.	
3	TIMEOUTEN		I^{2}C bus Time-out Enable. When disabled, the time-out function is internally reset.	0x0
		0	Disabled. Time-out function is disabled.	
		1	Enabled. Time-out function is enabled. Both types of time-out flags will be generated and will cause interrupts if they are enabled. Typically, only one time-out will be used in a system.	
4	MONCLKSTR		Monitor function Clock Stretching.	0x0
		0	Disabled. The Monitor function will not perform clock stretching. Software or DMA may not always be able to read data provided by the Monitor function before it is overwritten. This mode may be used when non-invasive monitoring is critical.	
		1	Enabled. The Monitor function will perform clock stretching in order to ensure that software or DMA can read all incoming data supplied by the Monitor function.	
5	HSCAPABLE		High-speed mode Capable enable. Since High Speed mode alters the way I^{2}C pins drive and filter, as well as the timing for certain I^{2}C signalling, enabling High-speed mode applies to all functions: Master, Slave, and Monitor.	0x0
		0	Standard or Fast modes. The I^{2}C interface will support Standard-mode, Fast-mode, and Fast-mode Plus, to the extent that the pin electronics support these modes. Any changes that need to be made to the pin controls, such as changing the drive strength or filtering, must be made by software via the IOCON register associated with each I^{2}C pin,	
		1	High-speed. In addition to Standard-mode, Fast-mode, and Fast-mode Plus, the I^{2}C interface will support High-speed mode to the extent that the pin electronics support these modes. See Section 26.7.2.2 for more information.	
31:6	-	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.2 I2C Status register

The STAT register provides status flags and state information about all of the functions of the I^{2}C interface. Access to bits in this register varies. RO = Read-only, W1C = write 1 to clear.

Details of the master and slave states described in the MSTSTATE and SLVSTATE bits in this register are listed in Table 418 and Table 419.

Table 417. I^{2}C Status register (STAT, offset 0x804) bit description

Bit	Symbol	Value	Description	Reset value	Access
0	MSTPENDING		Master Pending. Indicates that the Master is waiting to continue communication on the I^{2}C-bus (pending) or is idle. When the master is pending, the MSTSTATE bits indicate what type of software service if any the master expects. This flag will cause an interrupt when set if, enabled via the INTENSET register. The MSTPENDING flag is not set when the DMA is handling an event (if the MSTDMA bit in the MSTCTL register is set). If the master is in the idle state, and no communication is needed, mask this interrupt.	0x1	RO
		0	In progress. Communication is in progress and the Master function is busy and cannot currently accept a command.		
		1	Pending. The Master function needs software service or is in the idle state. If the master is not in the idle state, it is waiting to receive or transmit data or the NACK bit.		
3:1	MSTSTATE		Master State code. The master state code reflects the master state when the MSTPENDING bit is set, that is the master is pending or in the idle state. Each value of this field indicates a specific required service for the Master function. All other values are reserved. See Table 418 for details of state values and appropriate responses.	0x0	RO
		0x0	Idle. The Master function is available to be used for a new transaction.		
		0x1	Receive ready. Received data available (Master Receiver mode). Address plus Read was previously sent and Acknowledged by slave.		
		0x2	Transmit ready. Data can be transmitted (Master Transmitter mode). Address plus Write was previously sent and Acknowledged by slave.		
		0x3	NACK Address. Slave NACKed address.		
		0x4	NACK Data. Slave NACKed transmitted data.		
4	MSTARBLOSS		Master Arbitration Loss flag. This flag can be cleared by software writing a 1 to this bit. It is also cleared automatically when a 1 is written to MSTCONTINUE.	0x0	W1C
		0	No Arbitration Loss has occurred.		
		1	Arbitration loss. The Master function has experienced an Arbitration Loss. At this point, the Master function has already stopped driving the bus and gone to an idle state. Software can respond by doing nothing, or by sending a Start in order to attempt to gain control of the bus when it next becomes idle.		
5	-	-	Reserved. Read value is undefined, only zero should be written.	-	-

Table 417. I^{2}C Status register (STAT, offset 0x804) bit description *…continued*

Bit	Symbol	Value	Description	Reset value	Access
6	MSTSTSTPERR		Master Start/Stop Error flag. This flag can be cleared by software writing a 1 to this bit. It is also cleared automatically when a 1 is written to MSTCONTINUE.	0x0	W1C
		0	No Start/Stop Error has occurred.		
		1	The Master function has experienced a Start/Stop Error. A Start or Stop was detected at a time when it is not allowed by the I^{2}C specification. The Master interface has stopped driving the bus and gone to an idle state, no action is required. A request for a Start could be made, or software could attempt to insure that the bus has not stalled.		
7	-	-	Reserved. Read value is undefined, only zero should be written.	-	-
8	SLVPENDING		Slave Pending. Indicates that the Slave function is waiting to continue communication on the I^{2}C-bus and needs software service. This flag will cause an interrupt when set if enabled via INTENSET. The SLVPENDING flag is not set when the DMA is handling an event (if the SLVDMA bit in the SLVCTL register is set). The SLVPENDING flag is read-only and is automatically cleared when a 1 is written to the SLVCONTINUE bit in the SLVCTL register. The point in time when SlvPending is set depends on whether the I^{2}C interface is in HSCAPABLE mode. See Section 26.7.2.2.2. When the I^{2}C interface is configured to be HSCAPABLE, HS master codes are detected automatically. Due to the requirements of the HS I^{2}C specification, slave addresses must also be detected automatically, since the address must be acknowledged before the clock can be stretched.	0x0	RO
		0	In progress. The Slave function does not currently need service.		
		1	Pending. The Slave function needs service. Information on what is needed can be found in the adjacent SLVSTATE field.		
10:9	SLVSTATE		Slave State code. Each value of this field indicates a specific required service for the Slave function. All other values are reserved. See Table 419 for state values and actions. **Remark:** The occurrence of some states and how they are handled are affected by DMA mode and Automatic Operation modes.	0x0	RO
		0x0	Slave address. Address plus R/W received. At least one of the four slave addresses has been matched by hardware.		
		0x1	Slave receive. Received data is available (Slave Receiver mode).		
		0x2	Slave transmit. Data can be transmitted (Slave Transmitter mode).		
11	SLVNOTSTR		Slave Not Stretching. Indicates when the slave function is stretching the I^{2}C clock. This is needed in order to gracefully invoke deep-sleep mode during slave operation. This read-only flag reflects the slave function status in real time.	0x1	RO
		0	Stretching. The slave function is currently stretching the I^{2}C bus clock. deep-sleep mode cannot be entered at this time.		
		1	Not stretching. The slave function is not currently stretching the I^{2}C bus clock. deep-sleep mode could be entered at this time.		

Table 417. I²C Status register (STAT, offset 0x804) bit description *…continued*

Bit	Symbol	Value	Description	Reset value	Access
13:12	SLVIDX		Slave address match Index. This field is valid when the I²C slave function has been selected by receiving an address that matches one of the slave addresses defined by any enabled slave address registers, and provides an identification of the address that was matched. It is possible that more than one address could be matched, but only one match can be reported here.	0x0	RO
		0x0	Address 0. Slave address 0 was matched.		
		0x1	Address 1. Slave address 1 was matched.		
		0x2	Address 2. Slave address 2 was matched.		
		0x3	Address 3. Slave address 3 was matched.		
14	SLVSEL		Slave selected flag. SLVSEL is set after an address match when software tells the Slave function to acknowledge the address, or when the address has been automatically acknowledged. It is cleared when another address cycle presents an address that does not match an enabled address on the Slave function, when slave software decides to NACK a matched address, when there is a Stop detected on the bus, when the master NACKs slave data, and in some combinations of Automatic Operation. SLVSEL is not cleared if software NACKs data.	0x0	RO
		0	Not selected. The Slave function is not currently selected.		
		1	Selected. The Slave function is currently selected.		
15	SLVDESEL		Slave Deselected flag. This flag will cause an interrupt when set if enabled via INTENSET. This flag can be cleared by writing a 1 to this bit.	0x0	W1C
		0	Not deselected. The Slave function has not become deselected. This does not mean that it is currently selected. That information can be found in the SLVSEL flag.		
		1	Deselected. The Slave function has become deselected. This is specifically caused by the SLVSEL flag changing from 1 to 0. See the description of SLVSEL for details on when that event occurs.		
16	MONRDY		Monitor Ready. This flag is cleared when the MONRXDAT register is read.	0x0	RO
		0	No data. The Monitor function does not currently have data available.		
		1	Data waiting. The Monitor function has data waiting to be read.		
17	MONOV		Monitor Overflow flag.	0x0	W1C
		0	No overrun. Monitor data has not overrun.		
		1	Overrun. A Monitor data overrun has occurred. This can only happen when Monitor clock stretching not enabled via the MONCLKSTR bit in the CFG register. Writing 1 to this bit clears the flag.		
18	MONACTIVE		Monitor Active flag. Indicates when the Monitor function considers the I²C bus to be active. Active is defined here as when some Master is on the bus: a bus Start has occurred more recently than a bus Stop.	0x0	RO
		0	Inactive. The Monitor function considers the I²C bus to be inactive.		
		1	Active. The Monitor function considers the I²C bus to be active.		

Table 417. I^{2}C Status register (STAT, offset 0x804) bit description *…continued*

Bit	Symbol	Value	Description	Reset value	Access
19	MONIDLE		Monitor Idle flag. This flag is set when the Monitor function sees the I^{2}C bus change from active to inactive. This can be used by software to decide when to process data accumulated by the Monitor function. This flag will cause an interrupt when set if enabled via the INTENSET register. The flag can be cleared by writing a 1 to this bit.	0x0	W1C
		0	Not idle. The I^{2}C bus is not idle, or this flag has been cleared by software.		
		1	Idle. The I^{2}C bus has gone idle at least once since the last time this flag was cleared by software.		
23:20	-	-	Reserved. Read value is undefined, only zero should be written.	-	-
24	EVENTTIMEOUT		Event Time-out Interrupt flag. Indicates when the time between events has been longer than the time specified by the TIMEOUT register. Events include Start, Stop, and clock edges. The flag is cleared by writing a 1 to this bit. No time-out is created when the I^{2}C-bus is idle.	0x0	W1C
		0	No time-out. I^{2}C bus events have not caused a time-out.		
		1	Event time-out. The time between I^{2}C bus events has been longer than the time specified by the TIMEOUT register.		
25	SCLTIMEOUT		SCL Time-out Interrupt flag. Indicates when SCL has remained low longer than the time specific by the TIMEOUT register. The flag is cleared by writing a 1 to this bit.	0x0	W1C
		0	No time-out. SCL low time has not caused a time-out.		
		1	Time-out. SCL low time has caused a time-out.		
31:26	-	-	Reserved. Read value is undefined, only zero should be written.	-	-

Table 418. Master function state codes (MSTSTATE)

MST STATE	Description	Actions	DMA allowed
0x0	**Idle.** The Master function is available to be used for a new transaction.	Send a Start or disable MSTPENDING interrupt if the Master function is not needed currently.	No
0x1	**Received data is available (Master Receiver mode).** Address	Read data and either continue, send a Stop, or send a Repeated Start.	Yes
0x2	**Data can be transmitted (Master Transmitter mode).** Address plus Write was previously sent and Acknowledged by slave.	Send data and continue, or send a Stop or Repeated Start.	Yes
0x3	**Slave NACKed address.**	Send a Stop or Repeated Start.	No
0x4	**Slave NACKed transmitted data.**	Send a Stop or Repeated Start.	No

Table 419. Slave function state codes (SLVSTATE)

SLVSTATE		Description	Actions	DMA allowed
0	SLVST_ADDR	**Address plus R/W received.** At least one of the 4 slave addresses has been matched by hardware.	Software can further check the address if needed, for instance if a subset of addresses qualified by SLVQUAL0 is to be used. Software can ACK or NACK the address by writing 1 to either SLVCONTINUE or SLVNACK. Also see Section 26.7.4 regarding 10-bit addressing.	No
1	SLVST_RX	**Received data is available (Slave Receiver mode).**	Read data, reply with an ACK or a NACK.	Yes
2	SLVST_TX	**Data can be transmitted (Slave Transmitter mode).**	Send data. Note that when the Master NACKs dat transmitted by the slave, the slave becomes de-selected.	Yes

26.6.3 Interrupt Enable Set and read register

The INTENSET register controls which I²C status flags generate interrupts. Writing a 1 to a bit position in this register enables an interrupt in the corresponding position in the STAT register (Table 417), if an interrupt is supported there. Reading INTENSET indicates which interrupts are currently enabled.

Table 420. Interrupt Enable Set and read register (INTENSET, offset 0x808) bit description

Bit	Symbol	Value	Description	Reset value
0	MSTPENDINGEN		Master Pending interrupt Enable.	0x0
		0	Disabled. The MstPending interrupt is disabled.	
		1	Enabled. The MstPending interrupt is enabled.	
3:1	-	-	Reserved. Read value is undefined, only zero should be written.	-
4	MSTARBLOSSEN		Master Arbitration Loss interrupt Enable.	0x0
		0	Disabled. The MstArbLoss interrupt is disabled.	
		1	Enabled. The MstArbLoss interrupt is enabled.	
5	-	-	Reserved. Read value is undefined, only zero should be written.	-
6	MSTSTSTPERREN		Master Start/Stop Error interrupt Enable.	0x0
		0	Disabled. The MstStStpErr interrupt is disabled.	
		1	Enabled. The MstStStpErr interrupt is enabled.	
7	-	-	Reserved. Read value is undefined, only zero should be written.	-
8	SLVPENDINGEN		Slave Pending interrupt Enable.	0x0
		0	Disabled. The SlvPending interrupt is disabled.	
		1	Enabled. The SlvPending interrupt is enabled.	
10:9	-	-	Reserved. Read value is undefined, only zero should be written.	-
11	SLVNOTSTREN		Slave Not Stretching interrupt Enable.	0x0
		0	Disabled. The SlvNotStr interrupt is disabled.	
		1	Enabled. The SlvNotStr interrupt is enabled.	
14:12	-	-	Reserved. Read value is undefined, only zero should be written.	-
15	SLVDESELEN		Slave Deselect interrupt Enable.	0x0
		0	Disabled. The SlvDeSel interrupt is disabled.	
		1	Enabled. The SlvDeSel interrupt is enabled.	

Table 420. Interrupt Enable Set and read register (INTENSET, offset 0x808) bit description

Bit	Symbol	Value	Description	Reset value
16	MONRDYEN		Monitor data Ready interrupt Enable.	0x0
		0	Disabled. The MonRdy interrupt is disabled.	
		1	Enabled. The MonRdy interrupt is enabled.	
17	MONOVEN		Monitor Overrun interrupt Enable.	0x0
		0	Disabled. The MonOv interrupt is disabled.	
		1	Enabled. The MonOv interrupt is enabled.	
18	-	-	Reserved. Read value is undefined, only zero should be written.	-
19	MONIDLEEN		Monitor Idle interrupt Enable.	0x0
		0	Disabled. The MonIdle interrupt is disabled.	
		1	Enabled. The MonIdle interrupt is enabled.	
23:20	-	-	Reserved. Read value is undefined, only zero should be written.	-
24	EVENTTIMEOUTEN		Event time-out interrupt Enable.	0x0
		0	Disabled. The Event time-out interrupt is disabled.	
		1	Enabled. The Event time-out interrupt is enabled.	
25	SCLTIMEOUTEN		SCL time-out interrupt Enable.	0x0
		0	Disabled. The SCL time-out interrupt is disabled.	
		1	Enabled. The SCL time-out interrupt is enabled.	
31:26	-	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.4 Interrupt Enable Clear register

Writing a 1 to a bit position in INTENCLR clears the corresponding position in the INTENSET register, disabling that interrupt. INTENCLR is a write-only register.

Bits that do not correspond to defined bits in INTENSET are reserved and only zeroes should be written to them.

Table 421. Interrupt Enable Clear register (INTENCLR, offset 0x80C) bit description

Bit	Symbol	Description	Reset value
0	MSTPENDINGCLR	Master Pending interrupt clear. Writing 1 to this bit clears the corresponding bit in the INTENSET register if implemented.	0x0
3:1	-	Reserved. Read value is undefined, only zero should be written.	-
4	MSTARBLOSSCLR	Master Arbitration Loss interrupt clear.	0x0
5	-	Reserved. Read value is undefined, only zero should be written.	-
6	MSTSTSTPERRCLR	Master Start/Stop Error interrupt clear.	0x0
7	-	Reserved. Read value is undefined, only zero should be written.	-
8	SLVPENDINGCLR	Slave Pending interrupt clear.	0x0
10:9	-	Reserved. Read value is undefined, only zero should be written.	-
11	SLVNOTSTRCLR	Slave Not Stretching interrupt clear.	0x0
14:12	-	Reserved. Read value is undefined, only zero should be written.	-
15	SLVDESELCLR	Slave Deselect interrupt clear.	0x0
16	MONRDYCLR	Monitor data Ready interrupt clear.	0x0
17	MONOVCLR	Monitor Overrun interrupt clear.	0x0
18	-	Reserved. Read value is undefined, only zero should be written.	-

Table 421. Interrupt Enable Clear register (INTENCLR, offset 0x80C) bit description *…continued*

Bit	Symbol	Description	Reset value
19	MONIDLECLR	Monitor Idle interrupt clear.	0x0
23:20	-	Reserved. Read value is undefined, only zero should be written.	-
24	EVENTTIMEOUTCLR	Event time-out interrupt clear.	0x0
25	SCLTIMEOUTCLR	SCL time-out interrupt clear.	0x0
31:26	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.5 Time-out value register

The TIMEOUT register allows setting an upper limit to certain I²C bus times, informing by status flag and/or interrupt when those times are exceeded.

Two time-outs are generated, and software can elect to use either of them.

1. EVENTTIMEOUT checks the time between bus events while the bus is not idle: Start, SCL rising, SCL falling, and Stop. The EVENTTIMEOUT status flag in the STAT register is set if the time between any two events becomes longer than the time configured in the TIMEOUT register. The EVENTTIMEOUT status flag can cause an interrupt if enabled to do so by the EVENTTIMEOUTEN bit in the INTENSET register.
2. SCLTIMEOUT checks only the time that the SCL signal remains low while the bus is not idle. The SCLTIMEOUT status flag in the STAT register is set if SCL remains low longer than the time configured in the TIMEOUT register. The SCLTIMEOUT status flag can cause an interrupt if enabled to do so by the SCLTIMEOUTEN bit in the INTENSET register. The SCLTIMEOUT can be used with the SMBus.

Also see Section 26.7.3 “Time-out”.

Table 422. Time-out value register (TIMEOUT, offset 0x810) bit description

Bit	Symbol	Description	Reset value
3:0	TOMIN	Time-out time value, bottom four bits. These are hard-wired to 0xF. This gives a minimum time-out of 16 I²C function clocks and also a time-out resolution of 16 I²C function clocks.	0xF
15:4	TO	Time-out time value. Specifies the time-out interval value in increments of 16 I²C function clocks, as defined by the CLKDIV register. To change this value while I²C is in operation, disable all time-outs, write a new value to TIMEOUT, then re-enable time-outs. 0x000 = A time-out will occur after 16 counts of the I²C function clock. 0x001 = A time-out will occur after 32 counts of the I²C function clock. ... 0xFFF = A time-out will occur after 65,536 counts of the I²C function clock.	0xFFF
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.6 Clock Divider register

The CLKDIV register divides down the Flexcomm Interface clock (FCLK) to produce the I²C function clock that is used to time various aspects of the I²C interface. The I²C function clock is used for some internal operations in the I²C interface and to generate the timing required by the I²C bus specification, some of which are user configured in the MSTTIME register for Master operation. Slave operation uses CLKDIV for some timing functions.

Table 423. I2C Clock Divider register (CLKDIV, offset 0x814) bit description

Bit	Symbol	Description	Reset value
15:0	DIVVAL	This field controls how the Flexcomm Interface clock (FCLK) is used by the I2C functions that need an internal clock in order to operate. See Section 26.7.2.1 “Rate calculations” 0x0000 = I2C clock divider provides FCLK divided by 1. 0x0001 = I2C clock divider provides FCLK divided by 2. 0x0002 = I2C clock divider provides FCLK divided by 3. ... 0xFFFF = I2C clock divider provides FCLK divided by 65,536.	0x0
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.7 Interrupt Status register

The INTSTAT register provides register provides a view of those interrupt flags that are currently enabled. This can simplify software handling of interrupts. See Table 417 for detailed descriptions of the interrupt flags.

Table 424. I2C Interrupt Status register (INTSTAT, offset 0x818) bit description

Bit	Symbol	Description	Reset value
0	MSTPENDING	Master Pending.	0x1
3:1	-	Reserved.	
4	MSTARBLOSS	Master Arbitration Loss flag.	0x0
5	-	Reserved. Read value is undefined, only zero should be written.	-
6	MSTSTSTPERR	Master Start/Stop Error flag.	0x0
7	-	Reserved. Read value is undefined, only zero should be written.	-
8	SLVPENDING	Slave Pending.	0x0
10:9	-	Reserved. Read value is undefined, only zero should be written.	-
11	SLVNOTSTR	Slave Not Stretching status.	0x1
14:12	-	Reserved. Read value is undefined, only zero should be written.	-
15	SLVDESEL	Slave Deselected flag.	0x0
16	MONRDY	Monitor Ready.	0x0
17	MONOV	Monitor Overflow flag.	0x0
18	-	Reserved. Read value is undefined, only zero should be written.	-
19	MONIDLE	Monitor Idle flag.	0x0
23:20	-	Reserved. Read value is undefined, only zero should be written.	-
24	EVENTTIMEOUT	Event time-out Interrupt flag.	0x0
25	SCLTIMEOUT	SCL time-out Interrupt flag.	0x0
31:26	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.8 Master Control register

The MSTCTL register contains bits that control various functions of the I²C Master interface. Only write to this register when the master is pending (MSTPENDING = 1 in the STAT register, Table 417).

Software should always write a complete value to MSTCTL, and not OR new control bits into the register as is possible in other registers such as CFG. This is due to the fact that MSTSTART and MSTSTOP are not self-clearing flags. ORing in new data following a Start or Stop may cause undesirable side effects.

After an initial I²C Start, MSTCTL should generally only be written when the MSTPENDING flag in the STAT register is set, after the last bus operation has completed. An exception is when DMA is being used and a transfer completes. In this case there is no MSTPENDING flag, and the MSTDMA control bit would be cleared by software potentially at the same time as setting either the MSTSTOP or MSTSTART control bit.

Remark: When in the idle or slave NACKed states (see Table 418), set the MSTDMA bit either with or after the MSTCONTINUE bit. MSTDMA can be cleared at any time.

Table 425. Master Control register (MSTCTL, offset 0x820) bit description

Bit	Symbol	Value	Description	Reset value
0	MSTCONTINUE		Master Continue. This bit is write-only.	0x0
		0	No effect.	
		1	Continue. Informs the Master function to continue to the next operation. This must done after writing transmit data, reading received data, or any other housekeeping related to the next bus operation.	
1	MSTSTART		Master Start control. This bit is write-only.	0x0
		0	No effect.	
		1	Start. A Start will be generated on the I²C bus at the next allowed time.	
2	MSTSTOP		Master Stop control. This bit is write-only.	0x0
		0	No effect.	
		1	Stop. A Stop will be generated on the I²C bus at the next allowed time, preceded by a NACK to the slave if the master is receiving data from the slave (Master Receiver mode).	
3	MSTDMA		Master DMA enable. Data operations of the I²C can be performed with DMA. Protocol type operations such as Start, address, Stop, and address match must always be done with software, typically via an interrupt. Address acknowledgement must also be done by software except when the I²C is configured to be HSCAPABLE (and address acknowledgement is handled entirely by hardware) or when Automatic Operation is enabled. When a DMA data transfer is complete, MSTDMA must be cleared prior to beginning the next operation, typically a Start or Stop.This bit is read/write.	0x0
		0	Disable. No DMA requests are generated for master operation.	
		1	Enable. A DMA request is generated for I²C master data operations. When this I²C master is generating Acknowledge bits in Master Receiver mode, the acknowledge is generated automatically.	
31:4	-	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.9 Master Time register

The MSTTIME register allows programming of certain times that may be controlled by the Master function. These include the clock (SCL) high and low times, repeated Start setup time, and transmitted data setup time.

The I^{2}C clock pre-divider is described in Table 423.

Table 426. Master Time register (MSTTIME, offset 0x824) bit description

Bit	Symbol	Value	Description	Reset value
2:0	MSTSCLLOW		Master SCL Low time. Specifies the minimum low time that will be asserted by this master on SCL. Other devices on the bus (masters or slaves) could lengthen this time. This corresponds to the parameter t_{LOW} in the I^{2}C bus specification. I^{2}C bus specification parameters t_{BUF} and $t_{SU;STA}$ have the same values and are also controlled by MSTSCLLOW.	0x7
		0x0	SCL low multiplier = 2. See Section 26.7.2.1 “Rate calculations”.	
		0x1	SCL low multiplier = 3. See Section 26.7.2.1 “Rate calculations”.	
		0x2	SCL low multiplier = 4. See Section 26.7.2.1 “Rate calculations”.	
		0x3	SCL low multiplier = 5. See Section 26.7.2.1 “Rate calculations”.	
		0x4	SCL low multiplier = 6. See Section 26.7.2.1 “Rate calculations”.	
		0x5	SCL low multiplier = 7. See Section 26.7.2.1 “Rate calculations”.	
		0x6	SCL low multiplier = 8. See Section 26.7.2.1 “Rate calculations”.	
		0x7	SCL low multiplier = 9. See Section 26.7.2.1 “Rate calculations”.	
3	-	-	Reserved.	0x0
6:4	MSTSCLHIGH		Master SCL High time. Specifies the minimum high time that will be asserted by this master on SCL. Other masters in a multi-master system could shorten this time. This corresponds to the parameter t_{HIGH} in the I^{2}C bus specification. I^{2}C bus specification parameters $t_{SU;STO}$ and $t_{HD;STA}$ have the same values and are also controlled by MSTSCLHIGH.	0x7
		0x0	SCL high multiplier = 2. See Section 26.7.2.1 “Rate calculations”.	
		0x1	SCL high multiplier = 3. See Section 26.7.2.1 “Rate calculations”.	
		0x2	SCL high multiplier = 4. See Section 26.7.2.1 “Rate calculations”.	
		0x3	SCL high multiplier = 5. See Section 26.7.2.1 “Rate calculations”.	
		0x4	SCL high multiplier = 6. See Section 26.7.2.1 “Rate calculations”.	
		0x5	SCL high multiplier = 7. See Section 26.7.2.1 “Rate calculations”.	
		0x6	SCL high multiplier = 8. See Section 26.7.2.1 “Rate calculations”.	
		0x7	SCL high multiplier = 9. See Section 26.7.2.1 “Rate calculations”.	
31:7	-	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.10 Master Data register

The MSTDAT register provides the means to read the most recently received data for the Master function, and to transmit data using the Master function.

Table 427. Master Data register (MSTDAT, offset 0x828) bit description

Bit	Symbol	Description	Reset value
7:0	DATA	Master function data register. Read: read the most recently received data for the Master function. Write: transmit data using the Master function.	0x0
31:8	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.11 Slave Control register

The SLVCTL register contains bits that control various functions of the I²C Slave interface. Only write to this register when the slave is pending (SLVPENDING = 1 in the STAT register, Table 417).

Refer to Section 26.7.8 “Automatic operation” for details of the AUTOACK, AUTOMATCHREAD, and related settings.

Remark: When in the slave address state (slave state 0, see Table 419), set the SLVDMA bit either with or after the SLVCONTINUE bit. SLVDMA can be cleared at any time.

Table 428. Slave Control register (SLVCTL, offset 0x840) bit description

Bit	Symbol	Value	Description	Reset Value
0	SLVCONTINUE		Slave Continue.	0x0
		0	No effect.	
		1	Continue. Informs the Slave function to continue to the next operation, by clearing the SLVPENDING flag in the STAT register. This must be done after writing transmit data, reading received data, or any other housekeeping related to the next bus operation. Automatic Operation has different requirements. SLVCONTINUE should not be set unless SLVPENDING = 1.	
1	SLVNACK		Slave NACK.	0x0
		0	No effect.	
		1	NACK. Causes the Slave function to NACK the master when the slave is receiving data from the master (Slave Receiver mode).	
2	-	-	Reserved. Read value is undefined, only zero should be written.	-
3	SLVDMA		Slave DMA enable.	0x0
		0	Disabled. No DMA requests are issued for Slave mode operation.	
		1	Enabled. DMA requests are issued for I²C slave data transmission and reception.	
7:4	-	-	Reserved. Read value is undefined, only zero should be written.	-

Table 428. Slave Control register (SLVCTL, offset 0x840) bit description

Bit	Symbol	Value	Description	Reset Value
8	AUTOACK		Automatic Acknowledge.When this bit is set, it will cause an I^{2}C header which matches SLVADR0 and the direction set by AUTOMATCHREAD to be ACKed immediately; this is used with DMA to allow processing of the data without intervention. If this bit is cleared and a header matches SLVADR0, the behavior is controlled by AUTONACK in the SLVADR0 register: allowing NACK or interrupt.	0x0
		0	Normal, non-automatic operation. If AUTONACK = 0, an SlvPending interrupt is generated when a matching address is received. If AUTONACK = 1, received addresses are NACKed (ignored).	
		1	A header with matching SLVADR0 and matching direction as set by AUTOMATCHREAD will be ACKed immediately, allowing the master to move on to the data bytes. The ACK will clear this bit. If the address matches SLVADR0, but the direction does not match AUTOMATCHREAD, the behavior will depend on the AUTONACK bit in the SLVADR0 register: if AUTONACK is set, then it will be Nacked; else if AUTONACK is clear, then a SlvPending interrupt is generated.	
9	AUTOMATCHREAD		When AUTOACK is set, this bit controls whether it matches a read or write request on the next header with an address matching SLVADR0. Since DMA needs to be configured to match the transfer direction, the direction needs to be specified. This bit allows a direction to be chosen for the next operation.	0x0
		0	The expected next operation in Automatic Mode is an I^{2}C write.	
		1	The expected next operation in Automatic Mode is an I^{2}C read.	
31:10	-	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.12 Slave Data register

The SLVDAT register provides the means to read the most recently received data for the Slave function and to transmit data using the Slave function.

Table 429. Slave Data register (SLVDAT, offset 0x844) bit description

Bit	Symbol	Description	Reset value
7:0	DATA	Slave function data register. Read: read the most recently received data for the Slave function. Write: transmit data using the Slave function.	0x0
31:8	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.13 Slave Address 0 register

The SLVADR0 register allows enabling and defining one of the addresses that can be automatically recognized by the I^{2}C slave hardware.

The I^{2}C slave function has a total of 4 address comparators. The value in SLVADR0 can be qualified by the setting of the SLVQUAL0 register. The additional 3 address comparators do not include the address qualifier feature. For handling of the general call address, one of the 4 address registers can be programmed to respond to address 0.

Refer to Section 26.7.8 “Automatic operation” for details of AUTONACK and related settings.

Table 430. Slave Address 0 register (SLVADR[0], offset 0x848) bit description

Bit	Symbol	Value	Description	Reset value
0	SADISABLE0		Slave Address 0 Disable.	0x1
		0	Enabled. Slave Address 0 is enabled.	
		1	Ignored Slave Address 0 is ignored.	
7:1	SLVADR0	-	Slave Address. Seven bit slave address that is compared to received addresses if enabled. The compare can be affected by the setting of the SLVQUAL0 register.	0x0
14:8	-	-	Reserved. Read value is undefined, only zero should be written.	-
15	AUTONACK		Automatic NACK operation. Used in conjunction with AUTOACK and AUTOMATCHREAD, allows software to ignore I²C traffic while handling previous I²C data or other operations.	0x0
		0	Normal operation, matching I²C addresses are not ignored.	
		1	Automatic-only mode. All incoming addresses are ignored (NACKed), unless AUTOACK is set, it matches SLVADR0, and AUTOMATCHREAD matches the direction.	
31:16	-	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.14 Slave Address 1, 2, and 3 registers

These slave address registers provide for three additional addresses that can be automatically recognized by the I2C slave hardware.

Table 431. Slave Address registers (SLVADR[1:3], offset [0x84C:0x854]) bit description

Bit	Symbol	Value	Description	Reset value
0	SADISABLE		Slave Address n Disable.	0x1
		0	Enabled. Slave Address n is enabled.	
		1	Ignored Slave Address n is ignored.	
7:1	SLVADR		Slave Address. Seven bit slave address that is compared to received addresses if enabled.	0x0
31:8	-	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.15 Slave address Qualifier 0 register

The SLVQUAL0 register can alter how Slave Address 0 (specified by the SLVADR0 register) is interpreted.

Table 432. Slave address Qualifier 0 register (SLVQUAL0, offset 0x858) bit description

Bit	Symbol	Value	Description	Reset Value
0	QUALMODE0		Qualify mode for slave address 0.	0x0
		0	Mask. The SLVQUAL0 field is used as a logical mask for matching address 0.	
		1	Extend. The SLVQUAL0 field is used to extend address 0 matching in a range of addresses.	
7:1	SLVQUAL0	-	Slave address Qualifier for address 0. A value of 0 causes the address in SLVADR0 to be used as-is, assuming that it is enabled. If QUALMODE0 = 0, any bit in this field which is set to 1 will cause an automatic match of the corresponding bit of the received address when it is compared to the SLVADR0 register. If QUALMODE0 = 1, an address range is matched for address 0. This range extends from the value defined by SLVADR0 to the address defined by SLVQUAL0 (address matches when SLVADR0[7:1] ≤ received address ≤ SLVQUAL0[7:1]).	0x0
31:8	-	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.16 Monitor data register

The read-only MONRXDAT register provides information about events on the I²C bus, primarily to facilitate debugging of the I²C during application development. All data addresses and data passing on the bus and whether these were acknowledged, as well as Start and Stop events, are reported.

The Monitor function must be enabled by the MONEN bit in the CFG register. Monitor mode can be configured to stretch the I²C clock if data is not read from the MONRXDAT register in time to prevent it, via the MONCLKSTR bit in the CFG register**.** This can help ensure that nothing is missed but can cause the Monitor function to be somewhat intrusive (by potentially adding clock delays, depending on software or DMA response time). In order to improve the chance of collecting all Monitor information if clock stretching is not enabled, Monitor data is buffered such that it is available until the end of the next piece of information from the I²C bus.

Details of clock stretching are different in HS mode, see Section 26.7.2.2.2.

Table 433. Monitor data register (MONRXDAT, offset 0x880) bit description

Bit	Symbol	Value	Description	Reset value
7:0	MONRXDAT	-	Monitor function Receiver Data. This reflects every data byte that passes on the I²C pins.	0x0
8	MONSTART		Monitor Received Start.	0x0
		0	No start detected. The Monitor function has not detected a Start event on the I²C bus.	
		1	Start detected. The Monitor function has detected a Start event on the I²C bus.	
9	MONRESTART		Monitor Received Repeated Start.	0x0
		0	No repeated start detected. The Monitor function has not detected a Repeated Start event on the I²C bus.	
		1	Repeated start detected. The Monitor function has detected a Repeated Start event on the I²C bus.	

Table 433. Monitor data register (MONRXDAT, offset 0x880) bit description

Bit	Symbol	Value	Description	Reset value
10	MONNACK		Monitor Received NACK.	0x0
		0	Acknowledged. The data currently being provided by the Monitor function was acknowledged by at least one master or slave receiver.	
		1	Not acknowledged. The data currently being provided by the Monitor function was not acknowledged by any receiver.	
31:11	-	-	Reserved. Read value is undefined, only zero should be written.	-

26.6.17 Module identification register

The ID register identifies the type and revision of the I²C module. A generic SW driver can make use of this information register to implement module type or revision specific behavior.

Table 434. Module identification register (ID - offset 0xFFC) bit description

Bit	Symbol	Description	Reset Value
7:0	APERTURE	Aperture: encoded as (aperture size/4K) -1, so 0x00 means a 4K aperture.	0x00
11:8	MINOR_REV	Minor revision of module implementation, starting at 0. Software compatibility is expected between minor revisions.	-
15:12	MAJOR_REV	Major revision of module implementation, starting at 0. There may not be software compatibility between major revisions.	-
31:16	ID	Unique module identifier for this IP block.	0xE030

26.7 Functional description

26.7.1 AHB bus access

The bus interface to the I^{2}C registers contained in the Flexcomm Interface support only word writes. Byte and halfword writes are not supported in conjunction with the I^{2}C function.

26.7.2 Bus rates and timing considerations

Due to the nature of the I^{2}C bus, it is generally not possible to guarantee a specific clock rate on the SCL pin. On the I^{2}C-bus, the clock can be stretched by any slave device, extended by software overhead time, etc.

In a multi-master system, the master that provides the shortest SCL high time will cause that time to appear on SCL as long as that master is participating in I^{2}C traffic (i.e. when it is the only master on the bus, or during arbitration between masters).

In addition, I^{2}C implementations generally base subsequent actions on what actually happens on the bus lines. For instance, a bus master allows SCL to go high. It then monitors the line to make sure it actually did go high (this would be required in a multi-master system). This results in a small delay before the next action on the bus, caused by the rise time of the open drain bus line.

Rate calculations give a base frequency that represents the fastest that the I^{2}C bus could operate if nothing slows it down.

26.7.2.1 Rate calculations

Master timing

SCL high time (in Flexcomm Interface function clocks) =
I2C clock divider * SCL high multiplier (see Table 423 and Table 426)

SCL low time (in Flexcomm Interface function clocks) =
I2C clock divider * SCL low multiplier (see Table 423 and Table 426)

Nominal SCL rate =
Flexcomm Interface function clock rate / (SCL high time + SCL low time)

Remark: DIVVAL must be ≥ 1.

Remark: For 400 KHz clock rate, the clock frequency after the I^{2}C divider (divval) must be ≤ 2 MHz. Table 435 shows the recommended settings for 400 KHz clock rate.

Table 435. Settings for 400 KHz clock rate

Input clock to I2C	DIVVAL for CLKDIV register	MSTSCLHIGH for MSTTIME register	MSTSCLLOW for MSTTIME register
96 MHz	59	0	0
48 MHz	29	0	0
48 MHz	23	0	1
30 MHz	14	0	1
24 MHz	14	0	0
24 MHz	11	0	1
12 MHz	5	0	1

Slave timing

Most aspects of slave operation are controlled by SCL received from the I^{2}C bus master. However, if the slave function stretches SCL to allow for software response, it must provide sufficient data setup time to the master before releasing the stretched clock. This is accomplished by inserting one clock time of CLKDIV at that point.

If CLKDIV is already configured for master operation, that is sufficient. If only the slave function is used, CLKDIV should be configured such that one clock time is greater than the tSU;DAT value noted in the I^{2}C bus specification for the I^{2}C mode that is being used.

26.7.2.2 Bus rate support

The I^{2}C interface can support 4 modes from the I^{2}C bus specification:

- Standard-mode (SM, rate up to 100 kbits/s)
- Fast-mode (FM, rate up to 400 kbits/s)
- Fast-mode Plus (FM+, rate up to 1 Mbits/s)
- High-speed mode (HS, rate up to 3.4 Mbits/s)

Refer to Ref. 3 "UM10204" for details of I^{2}C modes and other details.

The I^{2}C interface supports Standard-mode, Fast-mode, and Fast-mode Plus with the same software sequence, which also supports SMBus. High-speed mode is intrinsically incompatible with SMBus due to conflicting requirements and limitations for clock stretching, and therefore requires a slightly different software sequence.

26.7.2.2.1 High-speed mode support

High-speed mode requires different pin filtering, somewhat different timing, and a different drive strength on SCL for the master function. The changes needed for the handling of the acknowledge bit mean that SMBus cannot be supported when the I^{2}C is configured to be HS capable. This limitation is intrinsic to the SMBus and High-speed I^{2}C specifications.

Because of the timing of changes to pin drive strength and filtering, the I^{2}C interface is designed to directly control those pin characteristics when configured to be HS capable. The I^{2}C also recognizes HS master codes and responds to programmed addresses when HS capable.

For software consistency, the changes required for handling of acknowledge and address recognition, and which affect when interrupts occur, are always in effect when the I2C is configured to be HS capable. This means that software does not need to know if a particular transfer is actually in HS mode or not.

26.7.2.2.2 Clock stretching

The I2C interface automatically stretches the clock when it does not have sufficient information on how to proceed, i.e. software has not supplied data and/or instructions to generate a start or stop. In principle, at least, I2C can allow the clock to be stretched by any bus participant at any time that SCL is low, in SM, FM, and MF+ modes.

In practice, the I2C interface described here may stretch SCL at the following times, in SM, FM, and MF+ modes:

- As a Slave:
  - after an address is received that complies with at least one slave address (before the address is acknowledged)
  - as a slave receiver, after each data byte received (software then acknowledges the data)
  - as a slave transmitter, after each data byte is sent and the matching acknowledge is received from the master
- As a master:

  after each
  - address is sent and the acknowledge bit has been received
  - as a master receiver, after each after each data byte is received (software then acknowledges the data)
  - as a master transmitter, after each data byte is sent and the matching acknowledge bit has been received from the slave

In HS mode:

- As a Slave (only slave functions in HS mode are supported on this device)
  - as a slave receiver, after each data byte is received and automatically acknowledged
  - as a slave transmitter, after each after each data byte is sent and the matching acknowledge is received from the master

In each case, the relevant pending flag (MSTPENDING or SLVPENDING) is set at the point where clock stretching occurs.

26.7.3 Time-out

A time-out feature on an I2C interface can be used to detect a “stuck” bus and potentially do something to alleviate the condition. Two different types of time-out are supported. Both types apply whenever the I2C interface and the time-out function are both enabled. Master, Slave, or Monitor functions do not need to be enabled.

In the first type of time-out, reflected by the EVENTTIMEOUT flag in the STAT register, the time between bus events governs the time-out check. These events include Start, Stop, and all changes on the I2C clock (SCL). This time-out is asserted when the time between any of these events is longer than the time configured in the TIMEOUT register. This time-out could be useful in monitoring an I^{2}C bus within a system as part of a method to keep the bus running of problems occur.

The second type of I^{2}C time-out is reflected by the SCLTIMEOUT flag in the STAT register. This time-out is asserted when the SCL signal remains low longer than the time configured in the TIMEOUT register. This corresponds to SMBus time-out parameter $T_{TIMEOUT}$. In this situation, a slave could reset its own I^{2}C interface in case it is the offending device. If all listening slaves (including masters that can be addressed as slaves) do this, then the bus will be released unless it is a current master causing the problem. Refer to the SMBus specification for more details.

Both types of time-out are generated only when the I^{2}C bus is considered busy, i.e. when there has been a Start condition more recently than a Stop condition.

26.7.4 Ten-bit addressing

Ten-bit addressing is accomplished by the I^{2}C master sending a second address byte to extend a particular range of standard 7-bit addresses. In the case of the master writing to the slave, the I^{2}C frame simply continues with data after the 2 address bytes. For the master to read from a slave, it needs to reverse the data direction after the second address byte. This is done by sending a Repeated Start, followed by a repeat of the same standard 7-bit address, with a Read bit. The slave must remember that it had been addressed by the previous write operation and stay selected for the subsequent read with the correct partial I^{2}C address.

For the Master function, the I^{2}C is simply instructed to perform the 2-byte addressing as a normal write operation, followed either by more write data, or by a Repeated Start with a repeat of the first part of the 10-bit slave address and then reading in the normal fashion.

For the Slave function, the first part of the address is automatically matched in the same fashion as 7-bit addressing. The slave address qualifier feature (see Section 26.6.15) can be used to intercept all potential 10-bit addresses (first address byte values F0 through F6), or just one. In the case of Slave Receiver mode, data is received in the normal fashion after software matches the first data byte to the remaining portion of the 10-bit address. The Slave function should record the fact that it has been addressed, in case there is a follow-up read operation.

For Slave Transmitter mode, the slave function responds to the initial address in the same fashion as for Slave Receiver mode, and checks that it has previously been addressed with a full 10-bit address. If the address matched is address 0, and address qualification is enabled, software must check that the first part of the 10-bit address is a complete match to the previous address before acknowledging the address.

26.7.5 Clocking and power considerations

The Master function of the I^{2}C always requires a peripheral clock to be running in order to operate. The Slave function can operate without any internal clocking when the slave is not currently addressed. This means that reduced power modes up to deep-sleep mode can be entered, and the device will wake up when the I^{2}C Slave function recognizes an address. Monitor mode can similarly wake up the device from a reduced power mode when information becomes available.

26.7.6 Interrupt handling

The I^{2}C provides a single interrupt output that handles all interrupts for Master, Slave, and Monitor functions.

26.7.7 DMA

DMA with the I^{2}C is done only for data transfer, DMA cannot handle control of the I^{2}C. Once DMA is transferring data, I^{2}C acknowledges are handled implicitly. No CPU intervention is required while DMA is transferring data.

Generally, data transfers can be handled by DMA for Master mode after an address is sent and acknowledged by a slave, and for Slave mode after software has acknowledged an address. In either mode, software is always involved in the address portion of a message. In master and slave modes, data receive and transmit data can be transferred by the DMA. The DMA supports three DMA requests: data transfer in master mode, slave mode, and Monitor mode.

DMA may be used in connection with Automatic Operation in order to minimize software overhead time for I^{2}C handling.

A received NACK (from a slave in Master mode, or from a master in Slave mode) will cause DMA to stop and an interrupt to be generated. A Repeated Start sensed on the bus will similarly cause DMA to stop and an interrupt to be generated.

The Monitor function may be used with DMA if a channel is available See Section 14.5.1.1.1 “DMA with I2C monitor mode” for how DMA channels are used with the Monitor function.

26.7.7.1 DMA as a Master transmitter

A basic sequence for a Master transmitter:

- Software sets up DMA to transmit a message.
- Software causes a slave address with write command to be sent and checks that the address was acknowledged.
- Software turns on DMA mode in the I^{2}C.
- DMA transfers data and eventually completes the transfer.
- Software causes a stop (or repeated start) to be sent.

Software will be invoked to handle any exceptions to the standard transfer, such as the slave sending a NACK before the end of the transfer.

26.7.7.2 DMA as a Master receiver

A basic sequence for a Master receiver:

- Software sets up DMA to receive a message.
- Software causes a slave address with read command to be sent and checks that the address was acknowledged.
- Software starts DMA.
- DMA completes.
- Software causes a stop or repeated start to be sent.

Software will be invoked to handle any exceptions to the standard transfer.

26.7.7.3 DMA as a Slave transmitter

A basic sequence for a Slave transmitter:

- Software acknowledges an I2C address.
- Software sets up DMA to transmit a message.
- Software starts DMA.
- DMA completes.

26.7.7.4 DMA as a Slave receiver

A basic sequence for a Slave receiver:

- Software receives an interrupt for a slave address received, and acknowledges the address.
- Software sets up DMA to receive a message, less the final data byte.
- Software starts DMA.
- DMA completes.
- Software sets SLVNACK prior to receiving the final data byte.
- Software receives the final data byte.

26.7.8 Automatic operation

Automatic operation modes provide a way to reduce software overhead for I2C slave functions with some limitations. They are intended to be used primarily in conjunction with slave DMA. Related control bits are SLVDMA, AUTOACK, and AUTOMATCHREAD in the SLCCTL register, and the AUTONACK bit in the SLVADR0 register. Table 27 shows how these controls may be used. These cases apply when an address matching SLVADR0, qualified by SLVQUAL0, is received.

Table 436: Automatic operation cases

Conditions:			**Response:**		
AUTONACK bit	**AUTOACK bit**	**Received R/W bit matches AUTOMATCHREAD**	**SLVPENDING interrupt generated?**	**ACK/NACK on I2C bus**	**Description**
0	0	x	Yes	None	Normal, non-automatic operation.
0	1	No	Yes	None	Automatic slave DMA: unexpected read/write case. Same as normal non-automatic operation.
x	1	Yes	No	ACK	Automatic slave DMA: expected read/write case. When the automatic Ack is sent, the SLVDMA bit is set and the AUTOACK bit is cleared.
1	0	x	No	NACK	Bus is ignored until software changes the setup.
1	1	No	No	NACK	Bus is ignored until software changes the setup.

UM11071

Chapter 27: LPC51U68 I^{2}S interface

Rev. 1.1 — 17 May 2018 User manual

27.1 How to read this chapter

I2S functionality is available on all LPC51U68 devices. I^{2}S is a function that is implemented in selected Flexcomm Interface peripherals. In the LPC51U68, the I^{2}S function is included in Flexcomm Interface 6 and Flexcomm Interface 7. Each of these Flexcomm Interfaces implements 1 I^{2}S channel pair, for a total of 2 channel pairs on this device.

27.2 Basic configuration

Initial configuration of the I^{2}S peripheral is accomplished as follows:

1. Peripheral clock: Make sure that the related Flexcomm Interface is enabled in the AHBCLKCTRL1 register (Section 6.5.17).
2. Flexcomm Interface clock: Select a clock source for the related Flexcomm Interface. Options are shown in Figure 9. Also see Section 6.5.28.

   Remark: The Flexcomm Interface function clock frequency should not be above 48 MHz.
3. If needed, use the PRESETCTRL1 register (Table 109) to reset the Flexcomm Interface that is about to have a specific peripheral function selected.
4. Select the desired Flexcomm Interface function by writing to the PSELID register of the related Flexcomm Interface (Section 23.7.1).
5. Pins: Make sure that the IOCON block is enabled in the AHBCLKCTRL0 register (Section 6.5.16). Select I^{2}S pins and pin modes through the relevant IOCON registers (Chapter 9).
6. I^{2}S rate: For master operation, the I^{2}S rate is determined by the clock selected in step 2 above, optionally modified using the DIV register (Table 443). Slave functions typically use the incoming I^{2}S clock directly.
7. Interrupts: To enable I^{2}S channel pair interrupts, see (FIFOINENSET in Section 27.7.8, FIFOINTENCLR in Section 27.7.9, and FIFOINTSTAT in Section 27.7.10). The related Flexcomm Interface interrupt must be enabled in the NVIC using the appropriate Interrupt Set Enable register (see Chapter 5).
8. DMA: I^{2}S channel pair master and slave functions can operated with the system DMA controller (see Chapter 14), and must be enabled in the FIFOCFG register (Section 27.7.5).

27.2.1 Configure the I2S for wake-up

In sleep mode, any I2S interrupt can wake up the part, provided that the interrupt is enabled in the INTENSET register and the NVIC. As long as the I2S clock is configured to be active in sleep mode, the I2S can wake up the part independently of whether the I2S block is configured in master or slave mode.

In deep-sleep mode, I2S clocks are normally turned off. However, if the I2S is configured to use the MCLK input or as a slave receiving an external SCK, the I2S can generate an interrupt and wake up the device. The appropriate interrupt(s) must be enabled in the I2S and the NVIC.

27.2.1.1 Wake-up from sleep mode

- Configure the I2S for the desired mode and other operational details, see Table 440 and Table 441.
- Enable the I2S interrupt in the NVIC.
- Any enabled I2S interrupt wakes up the part from sleep mode.

27.2.1.2 Wake-up from deep-sleep mode

- Configure the I2S for the desired mode and other operational details, see Table 440 and Table 441. The selected function clock of the I2S (whether internal or externally sourced) must be running.
- Enable the I2S interrupt in the STARTER0 register. See Table 155.
- Enable the I2S interrupt in the NVIC.
- Any enabled I2S interrupt wakes up the part from sleep mode.

Wake-up for DMA only

The device can optionally be woken up only far enough to perform needed DMA before returning to deep-sleep mode, without the CPU waking up at all. To accomplish this:

- Configure the I2S for the desired mode and other operational details, see Table 440 and Table 441. The selected function clock of the I2S (whether internal or externally sourced) must be running.
- Configure the DMA controller appropriately, including a transfer complete interrupt.
- Disable the related I2S interrupt in the NVIC.
- Enable the DMA interrupt in the NVIC.
- Enable FCWAKE and WAKEDMA in the HWWAKE register in Syscon (see Section 6.5.54).

27.3 Features

The I^{2}S bus provides a standard communication interface for streaming data transfer applications such as digital audio or data collection. The I^{2}S bus specification defines a 3-wire serial bus, having one data, one clock, and one word select/frame trigger signal, providing single or dual (mono or stereo) audio data transfer as well as other configurations.

The I^{2}S interface within one Flexcomm Interface provides at least one channel pair that can be configured as a master or a slave. Other channel pairs, if present, always operate as slaves. All of the channel pairs within one Flexcomm Interface share one set of I^{2}S signals, and are configured together for either transmit or receive operation, using the same mode, same data configuration and frame configuration. All such channel pairs can participate in a time division multiplexing (TDM) arrangement. For cases requiring an MCLK input and/or output, this is handled outside of the I^{2}S block in the system level clocking scheme.

- A Flexcomm Interface may implement one or more I^{2}S channel pairs, the first of which could be a master or a slave, and the rest of which would be slaves. All channel pairs are configured together for either transmit or receive and other shared attributes. The number of channel pairs is defined for each Flexcomm Interface, and may be from 0 to 4.
- Configurable data size for all channels within one Flexcomm Interface, from 4 bits to 32 bits. Each channel pair can also be configured independently to act as a single channel (mono as opposed to stereo operation).
- All channel pairs within one Flexcomm Interface share a single bit clock (SCK) and word select/frame trigger (WS), and data line (SDA).
- Data for all I^{2}S traffic within one Flexcomm Interface uses the Flexcomm Interface FIFO. The FIFO depth is 8 entries.
- Left justified and right justified data modes.
- DMA support using FIFO level triggering.
- TDM (Time Division Multiplexing) with a several stereo slots and/or mono slots is supported. Each channel pair can act as any data slot. Multiple channel pairs can participate as different slots on one TDM data line.
- The bit clock and WS can be selectively inverted.
- Sampling frequencies supported depends on the specific device configuration and applications constraints (e.g. system clock frequency, PLL availability, etc.) but generally supports standard audio data rates.

27.4 Architecture

The overall architecture of an example I²S subsystem is shown in Figure 65.

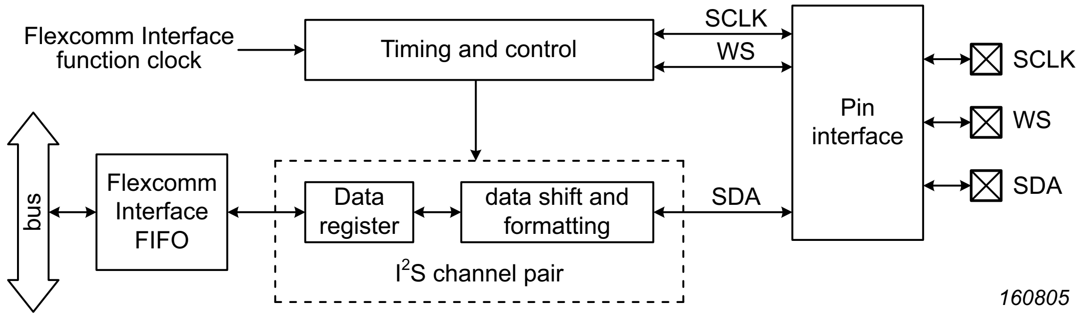


Fig 65. I²S block diagram

27.5 Terminology

Table 437: List of some terminology used in this document

Term	Description
Channel	Essentially, one piece of information on a single SDA line. In classic I²S, there is a single set of stereo data, which is 2 channels (left and right). In TDM modes, there may be many channels on a single SDA line.
Channel Pair	Two channels of data can be carried on one wire in classic I²S: left and right. On a microcontroller, this is typically what is implemented in a single instance of an I²S interface.
Classic I²S	This term is used in this document in reference to the original I²S bus specification from Philips Semiconductors. That specification defines 2 channel stereo data on SDA, where the WS state identifies the left (low) and right (high) channel, and data is delayed by 1 clock after WS transitions. The many variations of I²S that may be found have descended from this original specification.
DSP mode	DSP mode packs channel data together in the bit stream (left data followed by right data for each slot) and does not use WS to identify left and right data. WS may be a single SCK pulse, or a single data slot long pulse, in addition to a 50% duty cycle pulse. May be used in conjunction with TDM mode.
MCLK	Master Clock. In some I²S systems, this is provided as a multiple of the sample rate (fs), higher than the bit rate, such as 256 fs. Devices could potentially use this clock to construct a bit clock, or for internal operations such as data filtering.
SCK	Serial Clock. Sometimes referred to as BCK. This is a bit clock for data on the SDA line.
SDA	Serial Data. A single SDA provides one data stream, which may have many formats.
Slot	One data position in an I²S stream, typically each with the same slot length. For classic I²S, there is only one slot for stereo data. In a TDM mode, there can be several slots. In MONO mode, each slot is defined as one piece of data, rather than both left and right data.
TDM mode	TDM mode uses multiple data slots in order to put more channels of data into a single stream. May be used in conjunction with DSP mode or I²S mode.
WS	Word Select. Sometimes called LRCLK. Distinguishes left versus right data in most single stereo formats. Used as a frame delimiter in DSP and TDM modes.

27.6 Pin description

Remark: When the I^{2}S function is outputting SCK and/or WS, it uses a return signal from the related pin to adjust internal timing. For that reason, those signals must in fact be connected to a device pin, via IOCON selection, in order for the I^{2}S to operate.

Table 438: I^{2}S Pin Description

Pin	Type	Name used in Pin Configuration chapter	Description
SCK	I/O	FCn_SCK	Serial Clock for I2Sn. Clock signal used to synchronize the transfer of data on the SDA pin. It is normally driven by the master and received by one or more slaves. **Remark:** When the primary I^{2}S channel pair of a Flexcomm Interface is configured as a master, such that SCK is an output, it must actually be connected to a pin in order for the I^{2}S to work properly.
WS	I/O	FCn_TXD_SCL_MISO_WS	Word Select for I2Sn. Synchronizing signal for the beginning of each data frame and, in some modes, left vs. right channel data. It is normally driven by the master and received by one or more slaves. **Remark:** When the primary I^{2}S channel pair of a Flexcomm Interface is configured as a master, such that WS is an output, it must actually be connected to a pin in order for the I^{2}S to work properly.
SDA	I/O	FCn_RXD_SDA_MOSI_DATA	Serial Data for a single data stream used by one or more I^{2}S channel pairs of I2Sn. The format of data is configurable. It is driven by one or more transmitters and read by one or more receivers.
MCLK	I/O	MCLK	Master Clock. A multiple of the sample clock can optionally be provided by a master to other devices in the system, or can be received and divided down within a Flexcomm Interface in order to locally generate SCK and/or WS. This clock is not created inside the I^{2}S block. If MCLK is supported as an input to the device, it can be routed to the I^{2}S block and used to operate its functions. If MCLK is an output from the device, the clock that is used to create that MCLK can also be routed to the I^{2}S block and used to operate its functions.

27.7 Register description

The registers shown in Table 439 apply if the I2S function is selected in a Flexcomm Interface that supports I2S.

The reset value reflects the value of defined bits only, and does not include reserved bits.

Table 439: Register overview for the I2S function of one Flexcomm Interface

Name	Access	Offset [1]	Description	Reset Value	Section
Registers for the primary channel pair and shared registers:					
CFG1	R/W	0xC00	Configuration register 1 for the primary channel pair.	0x0	27.7.1
CFG2	R/W	0xC04	Configuration register 2 for the primary channel pair.	0x0	27.7.2
STAT	RO/W1C	0xC08	Status register for the primary channel pair.	0x0	27.7.3
DIV	R/W	0xC1C	Clock divider, used by all channel pairs.	0x0	27.7.4
Registers for FIFO control and data access:					
FIFOCFG	R/W	0xE00	FIFO configuration and enable register.	0x0	27.7.5
FIFOSTAT	R/W	0xE04	FIFO status register.	0x30	27.7.6
FIFOTRIG	R/W	0xE08	FIFO trigger level settings for interrupt and DMA request.	0x0	27.7.7
FIFOINTENSET	R/W1C	0xE10	FIFO interrupt enable set (enable) and read register.	0x0	27.7.8
FIFOINTENCLR	R/W1C	0xE14	FIFO interrupt enable clear (disable) and read register.	0x0	27.7.9
FIFOINTSTAT	RO	0xE18	FIFO interrupt status register.	0x0	27.7.10
FIFOWR	WO	0xE20	FIFO write data.	-	27.7.11
FIFOWR48H	WO	0xE24	FIFO write data for upper data bits. May only be used if the I2S is configured for 2x 24-bit data and not using DMA.	-	27.7.12
FIFORD	RO	0xE30	FIFO read data.	-	27.7.13
FIFORD48H	RO	0xE34	FIFO read data for upper data bits. May only be used if the I2S is configured for 2x 24-bit data and not using DMA.	-	27.7.14
FIFORDNOPOP	RO	0xE40	FIFO data read with no FIFO pop.	-	27.7.15
FIFORD48HNOPOP	RO	0xE44	FIFO data read for upper data bits with no FIFO pop. May only be used if the I2S is configured for 2x 24-bit data and not using DMA.	-	27.7.16
ID register:					
ID	RO	0xFFC	I2S Module identification. This value appears in the shared Flexcomm Interface peripheral ID register when I2S is the selected function.	0xE090 0000	27.7.17

[1] Offset is within the related Flexcomm Interface address space.

27.7.1 Configuration register 1

The CFG1 register contains mode settings, most of which apply to all I^{2}S channel pairs within one Flexcomm Interface. A few settings apply only to the primary channel pair, as noted.

Table 440. Configuration register 1 (CFG1 - offset 0xC00) bit description

Bit	Symbol	Value	Description	Reset Value
0	MAINENABLE		Main enable for I^{2}S function in this Flexcomm Interface	0x0
		0	All I^{2}S channel pairs in this Flexcomm Interface are disabled and the internal state machines, counters, and flags are reset. No other channel pairs can be enabled.	
		1	This I^{2}S channel pair is enabled. Other channel pairs in this Flexcomm Interface may be enabled in their individual PAIRENABLE bits.	
1	DATAPAUSE		Data flow Pause. Allows pausing data flow between the I^{2}S serializer/deserializer and the FIFO. This could be done in order to change streams, or while restarting after a data underflow or overflow. When paused, FIFO operations can be done without corrupting data that is in the process of being sent or received. Once a data pause has been requested, the interface may need to complete sending data that was in progress before interrupting the flow of data. Software must check that the pause is actually in effect before taking action. This is done by monitoring the DATAPAUSED flag in the STAT register. When DATAPAUSE is cleared, data transfer will resume at the beginning of the next frame.	0x0
		0	Normal operation, or resuming normal operation at the next frame if the I^{2}S has already been paused.	
		1	A pause in the data flow is being requested. It is in effect when DATAPAUSED in STAT = 1.	
3:2	PAIRCOUNT		Provides the number of I^{2}S channel pairs in this Flexcomm Interface This is a read-only field whose value may be different in other Flexcomm Interfaces. 00 = there is one I^{2}S channel pair in this Flexcomm Interface. 01 = there are two I^{2}S channel pairs in this Flexcomm Interface. 10 = there are three I^{2}S channel pairs in this Flexcomm Interface. 11 = there are four I^{2}S channel pairs in this Flexcomm Interface.	(see text)
5:4	MSTSLVCFG		Master / slave configuration selection, determining how SCK and WS are used by all channel pairs in this Flexcomm Interface.	0x0
		0x0	Normal slave mode, the default mode. SCK and WS are received from a master and used to transmit or receive data.	
		0x1	WS synchronized master. WS is received from another master and used to synchronize the generation of SCK, when divided from the Flexcomm Interface function clock.	
		0x2	Master using an existing SCK. SCK is received and used directly to generate WS, as well as transmitting or receiving data.	
		0x3	Normal master mode. SCK and WS are generated so they can be sent to one or more slave devices.	

Table 440. Configuration register 1 (CFG1 - offset 0xC00) bit description

Bit	Symbol	Value	Description	Reset Value
7:6	MODE		Selects the basic I^{2}S operating mode. Other configurations modify this to obtain all supported cases. See Section 27.8.2 “Formats and modes” for examples.	0x0
		0x0	I^{2}S mode a.k.a. “classic” mode. WS has a 50% duty cycle, with (for each enabled channel pair) one piece of left channel data occurring during the first phase, and one pieces of right channel data occurring during the second phase. In this mode, the data region begins one clock after the leading WS edge for the frame. **Remark:** For a 50% WS duty cycle, FRAMELEN must define an even number of I2S clocks for the frame. If FRAMELEN defines an odd number of clocks per frame, the extra clock will occur on the right.	
		0x1	DSP mode where WS has a 50% duty cycle. See remark for mode 0.	
		0x2	DSP mode where WS has a one clock long pulse at the beginning of each data frame.	
		0x3	DSP mode where WS has a one data slot long pulse at the beginning of each data frame.	
8	RIGHTLOW		Right channel data is in the Low portion of FIFO data. Essentially, this swaps left and right channel data as it is transferred to or from the FIFO. This bit is not used if the data width is greater than 24 bits. Note that if the ONECHANNEL field (bit 10 of this register) = 1, the one channel to be used is the nominally the left channel. POSITION can still place that data in the frame where right channel data is normally located. **Remark:** If all enabled channel pairs have ONECHANNEL = 1, then RIGHTLOW = 1 is not allowed.	0x0
		0	The right channel is taken from the high part of the FIFO data. For example, when data is 16 bits, FIFO bits 31:16 are used for the right channel.	
		1	The right channel is taken from the low part of the FIFO data. For example, when data is 16 bits, FIFO bits 15:0 are used for the right channel.	
9	LEFTJUST		Left Justify data.	0x0
		0	Data is transferred between the FIFO and the I^{2}S serializer/deserializer right justified, i.e. starting from bit 0 and continuing to the position defined by DATALEN. This would correspond to right justified data in the stream on the data bus.	
		1	Data is transferred between the FIFO and the I^{2}S serializer/deserializer left justified, i.e. starting from the MSB of the FIFO entry and continuing for the number of bits defined by DATALEN. This would correspond to left justified data in the stream on the data bus.	
10	ONECHANNEL		Single channel mode. Applies to both transmit and receive. This configuration bit applies only to the first I^{2}S channel pair. Other channel pairs may select this mode independently in their separate CFG1 registers.	0x0
		0	I^{2}S data for this channel pair is treated as left and right channels.	
		1	I^{2}S data for this channel pair is treated as a single channel, functionally the left channel for this pair. **Remark:** In mode 0 only, the right side of the frame begins at POSITION = 0x100. This is because mode 0 makes a clear distinction between the left and right sides of the frame. When ONECHANNEL = 1, the single channel of data may be placed on the right by setting POSITION to 0x100 + the data position within the right side (e.g. 0x108 would place data starting at the 8th clock after the middle of the frame). In other modes, data for the single channel of data is placed at the clock defined by POSITION.	
11	-		Reserved. Read value is undefined, only zero should be written.	-

Table 440. Configuration register 1 (CFG1 - offset 0xC00) bit description

Bit	Symbol	Value	Description	Reset Value
12	SCK_POL		SCK polarity.	0x0
		0	Data is launched on SCK falling edges and sampled on SCK rising edges (standard for I2S).	
		1	Data is launched on SCK rising edges and sampled on SCK falling edges.	
13	WS_POL		WS polarity.	0x0
		0	Data frames begin at a falling edge of WS (standard for classic I2S).	
		1	WS is inverted, resulting in a data frame beginning at a rising edge of WS (standard for most “non-classic” variations of I2S).	
15:14	-		Reserved. Read value is undefined, only zero should be written.	-
20:16	DATALEN		Data Length, minus 1 encoded, defines the number of data bits to be transmitted or received for all I2S channel pairs in this Flexcomm Interface. Note that data is only driven to or received from SDA for the number of bits defined by DATALEN. DATALEN is also used in these ways by the I2S: 1. Determines the size of data transfers between the FIFO and the I2S serializer/deserializer. See Section 27.8.4 “FIFO buffer configurations and usage” 2. In mode 1, 2, and 3, determines the location of right data following left data in the frame. 3. In mode 3 (where WS has a one data slot long pulse at the beginning of each data frame) determines the duration of the WS pulse. Values: 0x00 to 0x02 = not supported 0x03 = data is 4 bits in length 0x04 = data is 5 bits in length ... 0x1F = data is 32 bits in length	0x0
31:21	-		Reserved. Read value is undefined, only zero should be written.	-

27.7.2 Configuration register 2

The CFG2 register contains bits that control various aspects of data configuration.

Table 441. Configuration register 2 (CFG2 - offset 0xC04) bit description

Bit	Symbol	Description	Reset Value
8:0	FRAMELEN	Frame Length, minus 1 encoded, defines the number of clocks and data bits in the frames that this channel pair participates in. See Section 27.8.2.1 “Frame format”. 0x000 to 0x002 = not supported 0x003 = frame is 4 bits in total length 0x004 = frame is 5 bits in total length ... 0x1FF = frame is 512 bits in total length **Remark:** If FRAMELEN is an defines an odd length frame (e.g. 33 clocks) in mode 0 or 1, the extra clock appears in the right half. **Remark:** When MODE = 3, FRAMELEN must be larger than DATALEN in order for the WS pulse to be generated correctly.	0x0
15:9	-	Reserved. Read value is undefined, only zero should be written.	-
24:16	POSITION	Data Position. Defines the location within the frame of the data for this channel pair. POSITION + DATALEN must be less than FRAMELEN. See Section 27.8.2.1 “Frame format”. **Remark:** When MODE = 0, POSITION defines the location of data in both the left phase and right phase, starting one clock after the WS edge. In other modes, POSITION defines the location of data within the entire frame. ONECHANNEL = 1 while MODE = 0 is a special case, see the description of ONECHANNEL. **Remark:** The combination of DATALEN and the POSITION fields of all channel pairs must be made such that the channels do not overlap within the frame. 0x000 = data begins at bit position 0 (the first bit position) within the frame or WS phase. 0x001 = data begins at bit position 1 within the frame or WS phase. 0x002 = data begins at bit position 2 within the frame or WS phase. ...	0x0
31:25	-	Reserved. Read value is undefined, only zero should be written.	-

27.7.3 Status register

The STAT register provides status flags for the I^{2}S function, and does not include FIFO status. Note that the FIFO status register supplies peripheral interrupt notification and would be the status register normally observed first for an interrupt service. Some information in this register is read-only, some flags can be cleared by writing a 1 to them, details can be found in Table 442.

Table 442. Status register (STAT - offset 0xC08) bit description

Bit	Symbol	Value	Description	Reset value	Type
0	BUSY		Busy status for the primary channel pair. Other BUSY flags may be found in the STAT register for each channel pair.	0x0	RO
		0	The transmitter/receiver for channel pair is currently idle.		
		1	The transmitter/receiver for channel pair is currently processing data.		
1	SLVFRMERR		Slave Frame Error flag. This applies when at least one channel pair is operating as a slave. An error indicates that the incoming WS signal did not transition as expected due to a mismatch between FRAMELEN and the actual incoming I^{2}S stream.	0x0	W1C
		0	No error has been recorded.		
		1	An error has been recorded for some channel pair that is operating in slave mode. ERROR is cleared by writing a 1 to this bit position.		
2	LR		Left/Right indication. This flag is considered to be a debugging aid and is not expected to be used by an I^{2}S driver. Valid when one channel pair is busy. Indicates left or right data being processed for the currently busy channel pair.	-	RO
		0	Left channel.		
		1	Right channel.		
3	DATAPAUSED		Data Paused status flag. Applies to all I^{2}S channels	0x0	RO
		0	Data is not currently paused. A data pause may have been requested but is not yet in force, waiting for an allowed pause point. Refer to the description of the DATAPAUSE control bit in the CFG1 register.		
		1	A data pause has been requested and is now in force.		
31:4	-		Reserved. Read value is undefined, only zero should be written.	-	

27.7.4 Clock Divider register

The DIV register controls how the Flexcomm Interface function clock is used. See Section 27.8.3 “Data rates” for more details.

Remark: DIV must be set to 0 if SCK is used as an input clock for the I2S function, which is the case when the MSTSLVCFG field in the CFG1 register = 0 or 2.

Table 443. Clock Divider register (DIV - offset 0xC1C) bit description

Bit	Symbol	Description	Reset Value
11:0	DIV	This field controls how this I^{2}S block uses the Flexcomm Interface function clock. 0x000 = The Flexcomm Interface function clock is used directly. 0x001 = The Flexcomm Interface function clock is divided by 2. 0x002 = The Flexcomm Interface function clock is divided by 3. ... 0xFFF = The Flexcomm Interface function clock is divided by 4,096.	0x0
31:12	-	Reserved. Read value is undefined, only zero should be written.	-

27.7.5 FIFO Configuration register

This register configures FIFO usage. A peripheral must be selected within the Flexcomm Interface prior to configuring the FIFO.

Remark: Since all I^{2}S channels in a single Flexcomm Interface move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time. Also note that the FIFO for the selected I^{2}S data direction **must** be enabled because the FIFO is the only means for accessing I^{2}S data.

Table 444. FIFO Configuration register (FIFOCFG - offset 0xE00) bit description

Bit	Symbol	Value	Description	Reset Value	Access
0	ENABLETX		Enable the transmit FIFO.	0x0	R/W
		0	The transmit FIFO is not enabled.		
		1	The transmit FIFO is enabled.		
1	ENABLERX		Enable the receive FIFO.	0x0	R/W
		0	The receive FIFO is not enabled.		
		1	The receive FIFO is enabled.		
2	TXI2SE0		Transmit I^{2}S empty 0.Determines the value sent by the I^{2}S in transmit mode if the TX FIFO becomes empty. This value is sent repeatedly until the I^{2}S is paused, the error is cleared, new data is provided, and the I^{2}S is un-paused.	0x0	R/W
		0	If the TX FIFO becomes empty, the last value is sent. This setting may be used when the data length is 24 bits or less, or when MONO = 1 for this channel pair.		
		1	If the TX FIFO becomes empty, 0 is sent. Use if the data length is greater than 24 bits or if zero fill is preferred.		
3	PACK48		Packing format for 48-bit data. This relates to how data is entered into or taken from the FIFO by software or DMA.	0x0	R/W
		0	48-bit I^{2}S FIFO entries are handled as all 24-bit values.		
		1	48-bit I^{2}S FIFO entries are handled as alternating 32-bit and 16-bit values.		
5:4	SIZE		FIFO size configuration. This is a read-only field. 0x0, 0x1 = not applicable to I^{2}S. 0x2 = FIFO is configured as 8 entries of 32 bits, each corresponding to 2 16-bit data values for left and right channels. This setting occurs when the I^{2}S DATALEN is less than 16 bits, or from 25 to 32 bits. 0x3 = FIFO is configured as 8 entries of 48 bits, each corresponding to either 2 16-bit data values for left and right channels. This setting occurs when the I^{2}S DATALEN is from 17 to 24 bits.	-	RO
11:6	-		Reserved. Read value is undefined, only zero should be written.	-	-

Table 444. FIFO Configuration register (FIFOCFG - offset 0xE00) bit description

Bit	Symbol	Value	Description	Reset Value	Access
12	DMATX		DMA configuration for transmit.	0x0	R/W
		0	DMA is not used for the transmit function.		
		1	Generate a DMA request for the transmit function if the FIFO is not full. Generally, data interrupts would be disabled if DMA is enabled.		
13	DMARX		DMA configuration for receive.	0x0	R/W
		0	DMA is not used for the receive function.		
		1	Generate a DMA request for the receive function if the FIFO is not empty. Generally, data interrupts would be disabled if DMA is enabled.		
14	WAKETX		Wake-up for transmit FIFO level. This allows the device to be woken from reduced power modes (up to deep-sleep, as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion. See Section 6.5.54 “Hardware Wake-up control register”.	0x0	R/W
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the transmit FIFO level reaches the value specified by TXLVL in FIFOTRIG, even when the TXLVL interrupt is not enabled.		
15	WAKERX		Wake-up for receive FIFO level. This allows the device to be woken from reduced power modes (up to deep-sleep, as long as the peripheral function works in that power mode) without enabling the RXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion. See Section 6.5.54 “Hardware Wake-up control register”.	0x0	R/W
		0	Only enabled interrupts will wake up the device form reduced power modes.		
		1	A device wake-up for DMA will occur if the receive FIFO level reaches the value specified by RXLVL in FIFOTRIG, even when the RXLVL interrupt is not enabled.		
16	EMPTYTX		Empty command for the transmit FIFO. When a 1 is written to this bit, the TX FIFO is emptied.	-	WO
17	EMPTYRX		Empty command for the receive FIFO. When a 1 is written to this bit, the RX FIFO is emptied.	-	WO
31:18	-		Reserved. Read value is undefined, only zero should be written.	-	-

27.7.6 FIFO status register

This register provides status information for the FIFO and also indicates an interrupt from the peripheral function.

Remark: Since all I^{2}S channels in a single Flexcomm Interface move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time.

Table 445. FIFO status register (FIFOSTAT - offset 0xE04) bit description

Bit	Symbol	Description	Reset Value	Access
0	TXERR	TX FIFO error. Will be set if a transmit FIFO error occurs. This could be an overflow caused by pushing data into a full FIFO, or by an underflow if the FIFO is empty when data is needed. Cleared by writing a 1 to this bit.	0x0	R/W1C
1	RXERR	RX FIFO error. Will be set if a receive FIFO overflow occurs, caused by software or DMA not emptying the FIFO fast enough. Cleared by writing a 1 to this bit.	0x0	R/W1C
2	-	Reserved. Read value is undefined, only zero should be written.	-	
3	PERINT	Peripheral interrupt. When 1, this indicates that the peripheral function has asserted an interrupt. The details can be found by reading the peripheral's STAT register.	0x0	RO
4	TXEMPTY	Transmit FIFO empty. When 1, the transmit FIFO is empty. The peripheral may still be processing the last piece of data.	0x1	RO
5	TXNOTFULL	Transmit FIFO not full. When 1, the transmit FIFO is not full, so more data can be written. When 0, the transmit FIFO is full and another write would cause it to overflow.	0x1	RO
6	RXNOTEMPTY	Receive FIFO not empty. When 1, the receive FIFO is not empty, so data can be read. When 0, the receive FIFO is empty.	0x0	RO
7	RXFULL	Receive FIFO full. When 1, the receive FIFO is full. Data needs to be read out to prevent the peripheral from causing an overflow.	0x0	RO
12:8	TXLVL	Transmit FIFO current level. A 0 means the TX FIFO is currently empty, and the TXEMPTY and TXNOTFULL flags will be 1. Other values tell how much data is actually in the TX FIFO at the point where the read occurs. If the TX FIFO is full, the TXEMPTY and TXNOTFULL flags will be 0.	0x0	RO
15:13	-	Reserved. Read value is undefined, only zero should be written.	-	
20:16	RXLVL	Receive FIFO current level. A 0 means the RX FIFO is currently empty, and the RXFULL and RXNOTEMPTY flags will be 0. Other values tell how much data is actually in the RX FIFO at the point where the read occurs. If the RX FIFO is full, the RXFULL and RXNOTEMPTY flags will be 1.	0x0	RO
31:21	-	Reserved. Read value is undefined, only zero should be written.	-	-

27.7.7 FIFO trigger settings register

This register allows selecting when FIFO-level related interrupts occur.

Remark: Since all I2S channels in a single Flexcomm Interface move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time.

Table 446. FIFO trigger settings register (FIFOTRIG - offset 0xE08) bit description

Bit	Symbol	Value	Description	Reset Value
0	TXLVLENA		Transmit FIFO level trigger enable. The FIFO level trigger will cause an interrupt if enabled in FIFOINTENSET. This field is not used for DMA requests (see DMATX in FIFOCFG).	0x0
		0	Transmit FIFO level does not generate a FIFO level trigger.	
		1	An interrupt will be generated if the transmit FIFO level reaches the value specified by the TXLVL field in this register.	
1	RXLVLENA		Receive FIFO level trigger enable. This trigger will become an interrupt if enabled in FIFOINTENSET. This field is not used for DMA requests (see DMARX in FIFOCFG).	0x0
		0	Receive FIFO level does not generate a FIFO level trigger.	
		1	An interrupt will be generated if the receive FIFO level reaches the value specified by the RXLVL field in this register.	
7:2	-		Reserved. Read value is undefined, only zero should be written.	-
11:8	TXLVL		Transmit FIFO level trigger point. This field is used only when TXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode See Section 6.5.54 “Hardware Wake-up control register”. 0 = generate an interrupt when the TX FIFO becomes empty. 1 = generate an interrupt when the TX FIFO level decreases to one entry. ... 7 = generate an interrupt when the TX FIFO level decreases to 7 entries (is no longer full).	0x0
15:12	-		Reserved. Read value is undefined, only zero should be written.	-
19:16	RXLVL		Receive FIFO level trigger point. The RX FIFO level is checked when a new piece of data is received. This field is used only when RXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode See Section 6.5.54 “Hardware Wake-up control register”. 0 = generate an interrupt when the RX FIFO has one entry (is no longer empty). 1 = generate an interrupt when the RX FIFO has two entries. ... 7 = generate an interrupt when the RX FIFO increases to 8 entries (has become full).	0x0
31:20	-		Reserved. Read value is undefined, only zero should be written.	-

27.7.8 FIFO interrupt enable set and read

This register is used to enable various interrupt sources. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in this register causes those bits to be set. The FIFOINTENCLR register is used to clear bits in this register.

Remark: Since all I^{2}S channels in a single Flexcomm Interface move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time.

Table 447. FIFO interrupt enable set and read register (FIFOINTENSET - offset 0xE10) bit description

Bit	Symbol	Value	Description	Reset Value
0	TXERR		Determines whether an interrupt occurs when a transmit error occurs, based on the TXERR flag in the FIFOSTAT register.	0x0
		0	No interrupt will be generated for a transmit error.	
		1	An interrupt will be generated when a transmit error occurs.	
1	RXERR		Determines whether an interrupt occurs when a receive error occurs, based on the RXERR flag in the FIFOSTAT register.	0x0
		0	No interrupt will be generated for a receive error.	
		1	An interrupt will be generated when a receive error occurs.	
2	TXLVL		Determines whether an interrupt occurs when a the transmit FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0x0
		0	No interrupt will be generated based on the TX FIFO level.	
		1	If TXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the TX FIFO level decreases to the level specified by TXLVL in the FIFOTRIG register.	
3	RXLVL		Determines whether an interrupt occurs when a the receive FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register.	0x0
		0	No interrupt will be generated based on the RX FIFO level.	
		1	If RXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the when the RX FIFO level increases to the level specified by RXLVL in the FIFOTRIG register.	
31:4	-		Reserved. Read value is undefined, only zero should be written.	-

27.7.9 FIFO interrupt enable clear and read

The FIFOINTENCLR register is used to clear interrupt enable bits in FIFOINTENSET. The complete set of interrupt enables may also be read from this register as well as FIFOINTENSET.

Table 448. FIFO interrupt enable clear and read (FIFOINTENCLR - offset 0xE14) bit description

Bit	Symbol	Description	Reset value
0	TXERR	Writing a one to this bit disables the TXERR interrupt.	0x0
1	RXERR	Writing a one to this bit disables the RXERR interrupt.	0x0
2	TXLVL	Writing a one to this bit disables the interrupt caused by the transmit FIFO reaching the level specified by the TXLVL field in the FIFOTRIG register.	0x0
3	RXLVL	Writing a one to this bit disables the interrupt caused by the receive FIFO reaching the level specified by the RXLVL field in the FIFOTRIG register.	0x0
31:4	-	Reserved. Read value is undefined, only zero should be written.	-

27.7.10 FIFO interrupt status register

The read-only FIFOINTSTAT register provides a view of those interrupt flags that are both pending and currently enabled. This can simplify software handling of interrupts. Refer to the descriptions of interrupts in Section 27.7.6 and Section 27.7.7 for details.

Remark: Since all I^{2}S channels in a single Flexcomm Interface move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time.

Table 449. FIFO interrupt status register (FIFOINTSTAT - offset 0xE18) bit description

Bit	Symbol	Description	Reset Value
0	TXERR	TX FIFO error.	0x0
1	RXERR	RX FIFO error.	0x0
2	TXLVL	Transmit FIFO level interrupt.	0x0
3	RXLVL	Receive FIFO level interrupt.	0x0
4	PERINT	Peripheral interrupt.	0x0
31:5	-	Reserved. Read value is undefined, only zero should be written.	-

27.7.11 FIFO write data register

The FIFOWR register is used to write values to be transmitted to the FIFO. Details of how FIFOWR and FIFOWR48H are used can be found in Section 27.8.4 “FIFO buffer configurations and usage”.

Remark: Since all I^{2}S channels in a single Flexcomm Interface move data in the same direction, only FIFO read or write is meaningful at any particular time.

Table 450. FIFO write data register (FIFOWR - offset 0xE20) bit description

Bit	Symbol	Description	Reset Value
31:0	TXDATA	Transmit data to the FIFO. The number of bits used depends on configuration details.	-

27.7.12 FIFO write data for upper data bits

The FIFOWR48H register is used under certain conditions to write values to the FIFO. Details of how FIFOWR and FIFOWR48H are used can be found in Section 27.8.4 “FIFO buffer configurations and usage”.

Remark: Since all I^{2}S channels in a single Flexcomm Interface move data in the same direction, only FIFO read or write is meaningful at any particular time.

Table 451. FIFO write data for upper data bits (FIFOWR48H - offset 0xE24) bit description

Bit	Symbol	Description	Reset Value
23:0	TXDATA	Transmit data to the FIFO. Whether this register is used and the number of bits used depends on configuration details.	-
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

27.7.13 FIFO read data register

The FIFORD register is used to read values that have been received by the FIFO. Details of how FIFORD and FIFORD48H are used can be found in Section 27.8.4 “FIFO buffer configurations and usage”.

Remark: Since all I^{2}S channels in a single Flexcomm Interface move data in the same direction, only FIFO read or write is meaningful at any particular time.

Table 452. FIFO read data register (FIFORD - offset 0xE30) bit description

Bit	Symbol	Description	Reset Value
31:0	RXDATA	Received data from the FIFO. The number of bits used depends on configuration details.	-

27.7.14 FIFO read data for upper data bits

The FIFORD48H register is used under certain conditions to read values from the FIFO. Details of how FIFORD and FIFORD48H are used can be found in Section 27.8.4 “FIFO buffer configurations and usage”.

Remark: Since all I^{2}S channels in a single Flexcomm Interface move data in the same direction, only FIFO read or write is meaningful at any particular time.

Table 453. FIFO read data for upper data bits (FIFORD48H - offset 0xE34) bit description

Bit	Symbol	Description	Reset Value
23:0	RXDATA	Received data from the FIFO. Whether this register is used and the number of bits used depends on configuration details.	-
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

27.7.15 FIFO data read with no FIFO pop

This register acts in exactly the same way as FIFORD, except that it supplies data from the top of the FIFO without popping the FIFO (i.e. leaving the FIFO state unchanged). This could be used to allow system software to observe incoming data without interfering with the peripheral driver.

Table 454. FIFO data read with no FIFO pop (FIFORDNOPOP - offset 0xE40) bit description

Bit	Symbol	Description	Reset Value
31:0	RXDATA	Received data from the FIFO.	-

27.7.16 FIFO data read for upper data bits with no FIFO pop

This register acts in exactly the same way as FIFORD48H, except that it supplies data from the top of the FIFO without popping the FIFO (i.e. leaving the FIFO state unchanged). This could be used to allow system software to observe incoming data without interfering with the peripheral driver.

Table 455. FIFO data read for upper data bits with no FIFO pop (FIFORD48HNOPOP - offset 0xE44) bit description

Bit	Symbol	Description	Reset Value
23:0	RXDATA	Received data from the FIFO.	-
31:24	-	Reserved. Read value is undefined, only zero should be written.	-

27.7.17 Module identification register

The ID register identifies the type and revision of the module. A generic SW driver can make use of this information register to implement module type or revision specific behavior.

Table 456. Module identification register (ID - offset 0xFFC) bit description

Bit	Symbol	Description	Reset value
7:0	APERTURE	Aperture: encoded as (aperture size/4K) -1, so 0x00 means a 4K aperture.	0x0
11:8	MINOR_REV	Minor revision of module implementation, starting at 0. Software compatibility is expected between minor revisions.	-
15:12	MAJOR_REV	Major revision of module implementation, starting at 0. There may not be software compatibility between major revisions.	-
31:16	ID	Unique module identifier for this IP block.	0xE090

27.8 Functional description

27.8.1 AHB bus access

The bus interface to the I^{2}S registers contained in the Flexcomm Interface support only word writes. Byte and halfword writes are not supported in conjunction with the I^{2}S function.

27.8.2 Formats and modes

The format of data frames and WS is determined by several fields in the CFG1 and CFG2 registers, described in Sections 27.7.1 and 27.7.2 respectively. CFG1 and CFG2 together control the formatting of the data and the format of the frame in which the data is contained.

27.8.2.1 Frame format

The overall frame format is defined by fields in the CFG1 and CFG2 registers. The frame includes data related to the primary channel pair and any other channel pairs implemented by this I^{2}S. These fields plus the position of data for each channel pair, as determined by the POSITION field in CFG2, define the main features of the frame.

- MODE: 2-bit field in CFG1 that defines the overall character of the frame.
- FRAMELEN: 9-bit field in CFG2, defines the length of the data frame this I^{2}S participates in. This field is Minus 1 encoded: the value 63 means 64 clocks and bit positions in each frame.
- DATALEN: 5-bit field in CFG1, defines the number of data bits that are used by the transmitter or receiver. This field is minus 1 encoded: the value 15 means 16 data bits. For each channel pair, data is only driven to or received from SDA for the number of bits defined by DATALEN.

  DATALEN is also used in these ways:

  1) Determines the size of data transfers between the FIFO and the I^{2}S serializer/deserializer.

  2) When MODE = 0x1, 0x2, or 0x3 (i.e. not 0x0), determines the position of Right data following Left data within the frame.

  3) When MODE = 0x3, determines the duration of the WS pulse.

27.8.2.2 Example frame configurations

A sampling of frame slot formats are shown in the following figures. This is not an exhaustive set of possibilities, but shows the various frame formatting concepts. Note that slot identifications are illustrative only, data positions are flexible and there are no predefined slots for the hardware.

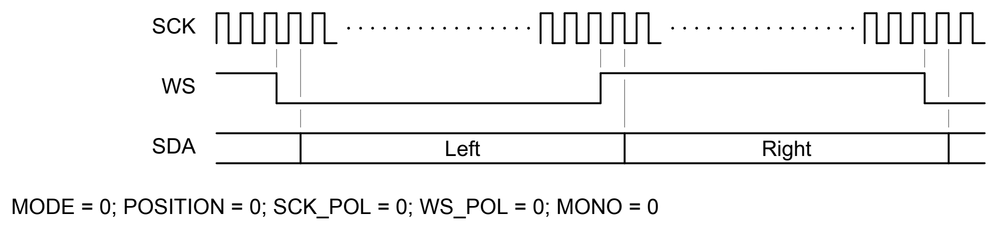


Fig 66. Classic I^{2}S mode

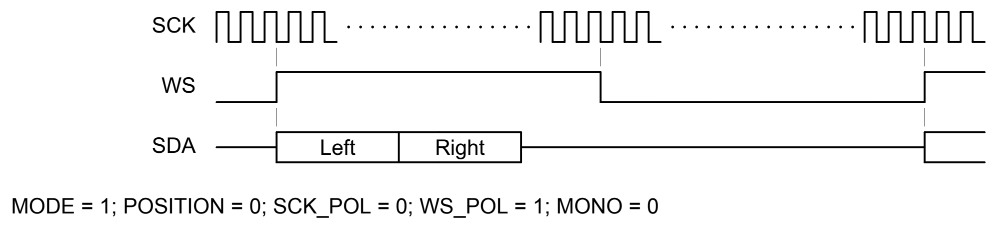


Fig 67. DSP mode with 50% WS

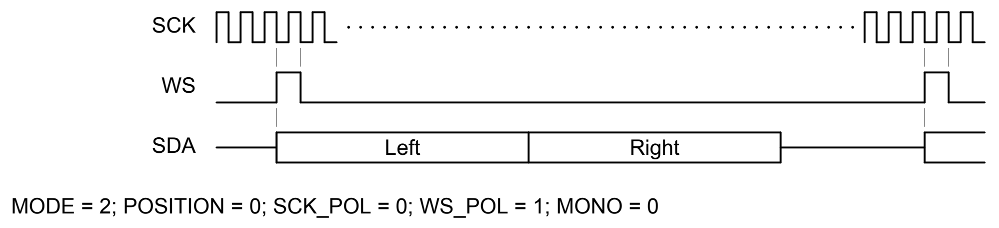


Fig 68. DSP mode with 1 SCK pulsed WS

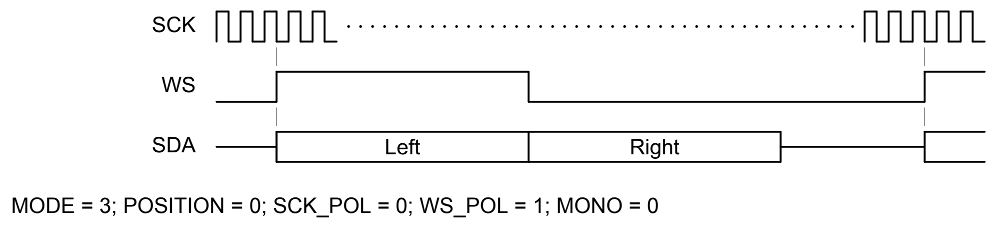
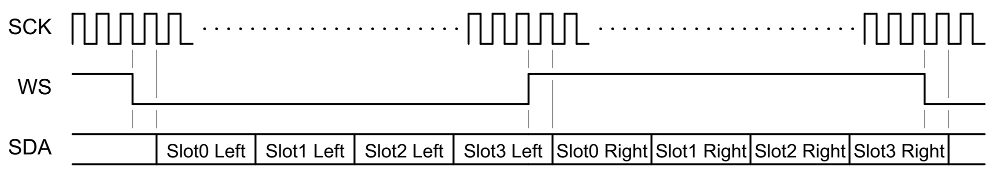


Fig 69. DSP mode with 1 slot pulsed WS

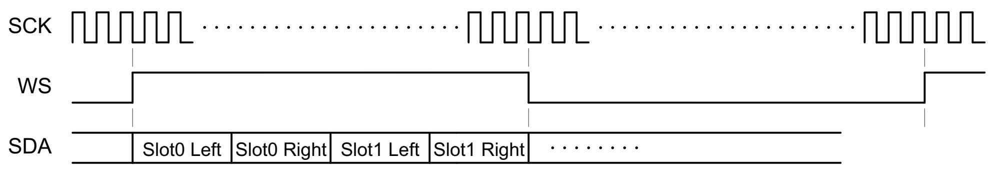


MODE = 0; SCK_POL = 0; WS_POL = 0; MONO = 0

POSITION = bit position of the first used data bit for a slot (within the data for each WS phase).

One Left /Right slot is used by one I²S channel pair. This example shows 4 data slots.

Fig 70. TDM in classic I²S mode

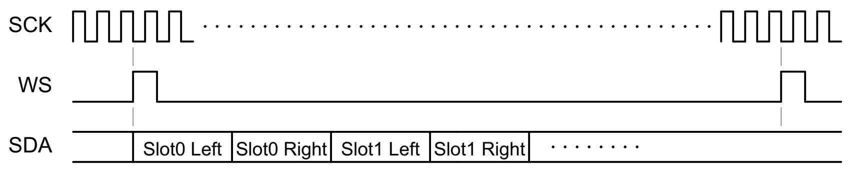


MODE = 1; SCK_POL = 0; WS_POL = 1; MONO = 0

POSITION = bit position of the first used data bit for each slot.

One Left/Right slot would be used by one channel pair.

Fig 71. TDM and DSP modes with 50% WS

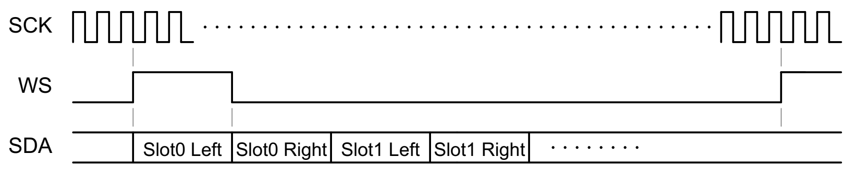


MODE = 2; SCK_POL = 0; WS_POL = 1; MONO = 0

POSITION = bit position of the first used data bit for each slot.

One Left/Right slot would be used by one channel pair.

Fig 72. TDM and DSP modes with 1 SCK pulsed WS

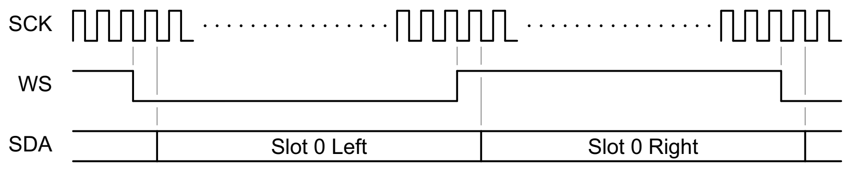


MODE = 3; SCK_POL = 0; WS_POL = 1; MONO = 0

POSITION = bit position of the first used data bit for each slot.

One Left/Right slot would be used by one channel pair.

Fig 73. TDM and DSP modes with 1 slot pulsed WS

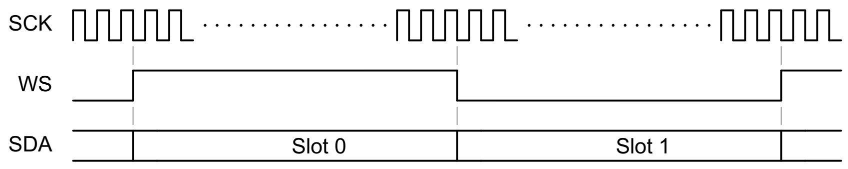


MODE = 0; SCK_POL = 0; WS_POL = 0; MONO = 1.

POSITiON = bit position of the first used data bit for slot 0 Left, bit position within the second half + 0x100 for Slot 0 Right.

One slot would be used by one I²S.

Fig 74. I²S mode, mono

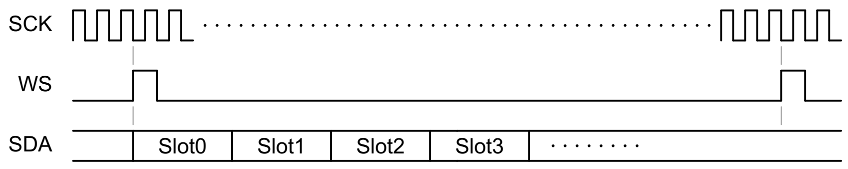


MODE = 1; SCK_POL = 0; WS_POL = 1; MONO = 1

POSITiON = bit position of the first used data bit for each slot.

One slot would be used by one I²S.

Fig 75. DSP mode, mono



MODE = 2; SCK_POL = 0; WS_POL = 1; MONO = 1.

POSITION = bit position of the first used data bit for each slot.

One slot would be used by one I²S.

Fig 76. TDM and DSP modes, mono, with WS pulsed for one SCK time

27.8.2.3 I^{2}S signal polarities

Figure 77 shows examples of SCK and WS polarities and how they relate to data positions.

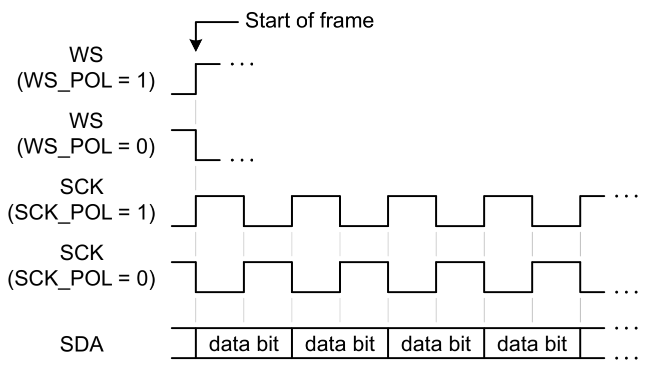


Fig 77. Data at the start of a frame, shown with both SCK and WS polarities

27.8.3 Data rates

27.8.3.1 Rate support

The actual I^{2}S clock rates, sample rates, etc. that can be supported depend on the clocking that is available to run the interface. As a slave, the interface will be receiving SCK from a master. In that case, there is an upper limit to how fast the interface can operate (this will be specified in the interface AC characteristics in a specific device data sheet) and a limit to how much data and be transferred across clock domains and handled by the CPU.

In general, the I^{2}S can support:

- Standard sample rates such as 16, 22.05, 32, 44.1, 48, and 96 kHz, and others.
- External MCLK inputs up to approximately 25 MHz (256 fs of a 96 kHz sample rate) and more. Refer to a specific device data sheet for details.

27.8.3.2 Rate calculations

For operation as a master, the frequency need as the clock input of the I^{2}S is generally an integer multiple of:

- Frame/sample rate * number of bits/clocks in a data frame

If this is a multiple of the desired frequency, the I^{2}S function divider can be used to produce the desired frequency.

Example 1

This I^{2}S channel pair is being used to transfer stereo audio data with 32 bit data slots and a 96 kHz sample rate.

Setup: the sample rate is 96 kHz, the frame is configured for two 32-bit data slots (32-bit stereo). The function clock divider output rate would be 96,000 * (2 * 32) = 6.144 MHz.

The value of DIV would be (function clock divider input frequency / the required divider output frequency) - 1. If the divider input is 24.576 MHz (256 fs of the 96 kHz sample rate), the divider needs to divide by 4 (DIV = 3) to obtain the target divider output rate of 6.144 MHz.

Example 2

This I^{2}S channel pair is being used to supply one 16-bit data slot in a 4 slot frame with a frame rate of 50 kHz.

Setup: the sample rate is 50 kHz, the frame is configured four 16-bit data slots, The function clock divider output rate would be 50,000 * (4 * 16) = 3.2 MHz.

The value of DIV would be (function clock divider input frequency / the required divider output frequency) - 1. If the divider input is 16 MHz, the divider needs to divide by 5 (DIV = 4) to obtain the target divider output rate of 3.2 MHz.

27.8.4 FIFO buffer configurations and usage

The Flexcomm Interface supports several possibilities of data packing/unpacking depending on the size of data being handled.

Some details of FIFO usage are determined by the value of the I^{2}S DATALEN field in the CFG1 register, and some other configuration bits as follows:

- If DATALEN specifies a number of data bits from 4 to 16:
  - The FIFO will be configured as 32 bits wide and 8 entries deep.
  - Each data transfer between the bus and the FIFO will be a pair of left and right values, which fit into a 32-bit word. The order of left and right data is selectable via the RIGHTLOW configuration bit.
  - If a channel pair is configured with ONECHANNEL = 1, then only one value is transferred, nominally the left.
- If DATALEN specifies a number of data bits from 17 to 24:
  - The FIFO will be configured as 48 bits wide and 8 entries deep.
  - Data transfer between the bus and the FIFO depends the PACK48 configuration bit and whether or not DMA is enabled. When DMA is enabled, all transfers are done with FIFOWR or FIFORD. When DMA is not enabled, transfers will alternate between FIFOWR or FIFORD and FIFOWR48H or FIFORD48H, depending on the data direction selected for the I^{2}S function. In all cases, the 2 transfers will constitute a pair of left and right values. The order of left and right data is selectable via the RIGHTLOW configuration bit.
  - If PACK48 = 0, each of the two transfers both define 17 to 24 bits of data. If PACK48 = 1, the first transfer provides 32 bits of data, the second provides the remainder need to complete the paired data as defined.
  - If a channel pair is configured with ONECHANNEL = 1, then only the left value is transferred using the FIFOWR or FIFORD register.
- If DATALEN specifies a number of data bits from 25 to 32:
  - The FIFO will be configured as 32 bits wide and 8 entries deep.
  - Each data transfer between the bus and the FIFO will be a single value, starting with left, then right.

– If a channel pair is configured with ONECHANNEL = 1, then only one value is transferred.

27.8.5 DMA

The Flexcomm Interface can generate DMA requests based on FIFO levels. Data transfers for any channel can be handled by DMA once the I^{2}S clocking and has been configured, that channel has been configured, DMA has been configured, and the I^{2}S bus is running. DMA operation is similar to any other serial peripheral.

DMA related configurations in the Flexcomm Interface I^{2}S may be found in the FIFOCFG register (Section 27.7.5) bits DMATX, DMARX, WAKETX, WAKERX, and PACK48, and in the FIFOTRIG register (Section 27.7.7) bits TXLVLENA, RXLVLENA, and fields TXLVL and RXLVL.

27.8.6 Clocking and power considerations

The master function of the I^{2}S requires the Flexcomm Interface function clock to be running in order to operate. The slave function can operate using external clocks, and can wake up the CPU when data is needed or available.

UM11071

Chapter 28: LPC51U68 12-bit ADC controller (ADC)

Rev. 1.1 — 17 May 2018 **User manual**

28.1 How to read this chapter

The ADC controller is available on all LPC51U68 devices. The number of ADC channels available is dependent on the package size.

28.2 Features

- 12-bit successive approximation analog to digital converter.
- Input multiplexing among up to 12 pins.
- Two configurable conversion sequences with independent triggers.
- Optional automatic high/low threshold comparison and “zero crossing” detection.
- Measurement range V_{REFN} to V_{REFP} (typically 3 V; not to exceed V_{DDA} voltage level).
- Typical 12-bit conversion rate of 5.0 Msps. Options for reduced resolution at higher conversion rates.
- Burst conversion mode for single or multiple inputs.
- Synchronous or asynchronous operation. Asynchronous operation maximizes flexibility in choosing the ADC clock frequency, Synchronous mode minimizes trigger latency and can eliminate uncertainty and jitter in response to a trigger.
- A temperature sensor is connected as an alternative input for ADC channel 0.

28.3 Basic configuration

Configure the ADC as follows:

- Set up the ADCTRL register.
- Use the ADC API to start up the ADC.
- The ADC block creates three interrupts which are connected to the NVIC: ADC0_SEQA, ADC0_SEQB, and ADC0_THCMP. The ADC0_THCMP interrupt at the NVIC combines ADC0_THCMP and ADC0_OVR conditions from the ADC as described in this chapter. See Table 79 for interrupt numbers. The sequence interrupts can also be configured as DMA triggers through the INPUT MUX (see Table 198) for each DMA channel and as inputs to the SCT.
- Use IOCON (Chapter 9) to connect and enable analog function on the ADC input pins.
- Calibration is required after every reset or power cycle of the ADC. The ADC API function may be used for this. Note that the ADC may be power cycled in deep-sleep mode if it is not requested to stay on when these modes are invoked by the Chip_POWER_EnterPowerMode API.
- There are two options in the ADC CTRL register to clock ADC conversions:
  - Use the system clock to clock the ADC in synchronous mode. This option allows exact timing of triggers but requires a system clock of 80 MHz to obtain the full ADC conversion speed.

– Use the ADC clock, determined by the ADCCLKSEL register (Table 125) and the ADCCLKDIV register (Table 133). Some clock sources are independent of the system clock, and may require extra time to synchronize ADC trigger inputs.

Configure the temperature sensor as follows:

- Clear the PDEN_TS bit in the PDRUNCFG0 register (Table 152) in order to enable the temperature sensor.
- Select the temperature sensor as source for channel 0 of the ADC by writing the value 3 to the INSEL register (see Section 28.6.2). In order to return ADC channel 0 to measuring its related device pin, write a 0 to the INSEL register.
- The digital temperature reading is available after an analog-to-digital conversion of ADC channel 0.

Remark: To convert the ADC conversion result into a temperature reading, see the specific device data sheet for calibration information.

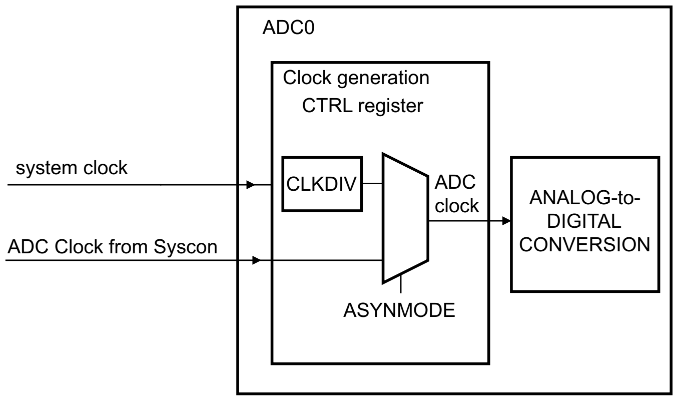


Fig 78. ADC clocking

28.4 Pin description

The ADC can measure the voltage on any of the input signals on the analog input channel. Digital signals must be disconnected from the ADC input pins when the ADC function is to be used by setting DIGIMODE = 0 on those pins in the related IOCON registers.

Warning: If the ADC is used, signal levels on analog input pins must not be above the level of V_{DDA} at any time. Otherwise, ADC readings will be invalid. If the ADC is not used in an application, then the pins associated with ADC inputs can be used as digital I/O pins.

The ADC pin triggers are movable (digital) functions. To use external pin triggers for ADC conversions, assign the pin triggers to a pin via IOCON. In addition to assigning the pin triggers to a pin, they must also be selected in the conversion sequence registers for each ADC conversion sequence defined.

The V_{REFP} and V_{REFN} pins provide a positive and negative reference voltage input. The result of the conversion is 4095 x (V_{IN} - V_{REFN}) / (V_{REFP} - V_{REFN}). The result of an input voltage below V_{REFN} is 0, and the result of an input voltage above V_{REFP} is 4095 (0xFFF).

Analog Power and Ground should typically be the same voltages as V_{DD} and V_{SS}, but should be isolated to minimize noise and error. If the ADC is not used, V_{DDA} and V_{REFP} should be tied to V_{DD}, and V_{SSA} and V_{REFN} should be tied to V_{SS}.

Table 457. ADC common supply and reference pins

Pin	Description
VDDA	Analog supply voltage. V_{REFP} must not exceed the voltage level on V_{DDA}. This pin should be tied to V_{DD} (not left floating) if the ADC is not used. **Remark:** The supply voltage V_{DD} must be equal to V_{DDA}.
VSSA	Analog ground. This pin should be tied to V_{SS} (not left floating) if the ADC is not used.
VREFP	Positive reference voltage. To operate the ADC within specifications at the maximum sampling rate, ensure that V_{REFP} = V_{DDA}. This pin should be tied to V_{DD} (not left floating) if the ADC is not used. **Remark:** V_{REFP} is internally connected (not separately pinned) with V_{DDA} for some packages/part numbers.
VREFN	Negative reference voltage. The voltage level should typically be equal Vss and Vssa.This pin should be tied to V_{SS} (not left floating) if the ADC is not used. **Remark:** V_{REFN} is internally connected (not separately pinned) with V_{SSA} for some packages/part numbers.

Table 458. ADC0 pin description

Function	Connect to	Description
ADC0_0	PIO0_29	Analog input channel 0.
ADC0_1	PIO0_30	Analog input channel 1.
ADC0_2	PIO0_31	Analog input channel 2.
ADC0_3	PIO1_0	Analog input channel 3.
ADC0_4	PIO1_1	Analog input channel 4.
ADC0_5	PIO1_2	Analog input channel 5.
ADC0_6	PIO1_3	Analog input channel 6.
ADC0_7	PIO1_4	Analog input channel 7.
ADC0_8	PIO1_5	Analog input channel 8.
ADC0_9	PIO1_6	Analog input channel 9.

Table 458. ADC0 pin description

Function	Connect to	Description
ADC0_10	PIO1_7	Analog input channel 10.
ADC0_11	PIO1_8	Analog input channel 11.
ADC0_PINTRIG0	PINT0	ADC0 pin trigger input 0, from Pin Interrupt 0. Select in SEQA_CTRL or SEQB_CTRL.
ADC0_PINTRIG1	PINT1	ADC0 pin trigger input 1, from Pin Interrupt 1. Select in SEQA_CTRL or SEQB_CTRL.

Recommended IOCON settings are shown in Table 459. See Chapter 9 for definitions of pin types.

Table 459: Suggested ADC input pin settings

IOCON bit(s)	Type D pin	Type A pin	Type I pin
10	-	OD: Set to 0.	-
9	-	Not used, set to 0.	-
8	-	FILTEROFF: Set to 1.	-
7	-	DIGIMODE: Set to 0.	-
6	-	INVERT: Set to 0.	-
5	-	Not used, set to 0.	-
4:3	-	MODE: Set to 0.	-
2:0	-	FUNC: Select GPIO as the pin function.	-
General comment	Not applicable for ADC.	Only potential choice for ADC inputs.	Not applicable for ADC.

28.5 General description

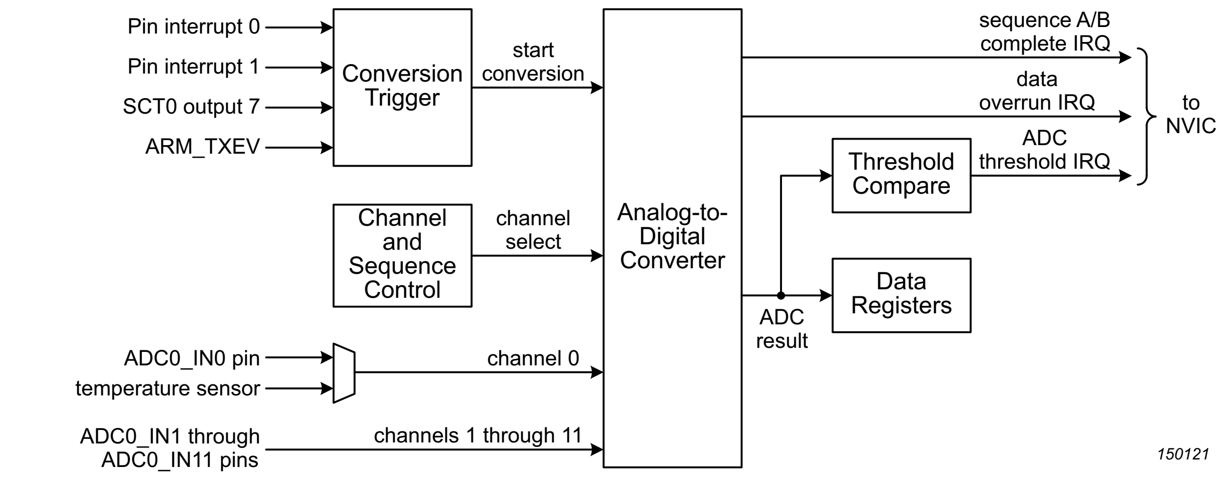


Fig 79. ADC block diagram

The ADC controller provides a great deal of flexibility in launching and controlling sequences of ADC conversions using the associated 12-bit, successive approximation ADC converter. ADC conversion sequences can be initiated under software control or in response to a selected hardware trigger.

Once the triggers are set up (software and hardware triggers can be mixed), the ADC runs through the pre-defined conversion sequences converting a sample whenever a trigger signal arrives until the sequence is disabled.

The ADC controller uses the system clock as a bus clock. The system clock or the asynchronous ADC clock (see Figure 78) can be used to create the ADC clock which drives the successive approximation process:

- In the synchronous operating mode, this ADC clock is derived from the system clock. In this mode, a programmable divider is included to scale the system clock to the maximum ADC clock rate of 80 MHz.
- In the asynchronous mode, an independent clock source is used as the ADC clock source without any further divider in the ADC. The maximum ADC clock rate is 80 MHz as well. **In this mode, the ADC clock frequency must not exceed five times the system clock.**

A fully accurate conversion requires 15 ADC clocks.

28.6 Register description

The reset value reflects the data stored in used bits only. It does not include reserved bits content.

Table 460. Register overview: ADC (base address 0x400A 0000)

Name	Access	Offset	Description	Reset value	Section
CTRL	R/W	0x000	ADC Control register. Contains the clock divide value, resolution selection, sampling time selection, and mode controls.	0x600	28.6.1
INSEL	R/W	0x004	Input Select. Allows selection of the temperature sensor as an alternate input to ADC channel 0.	0x0	28.6.2
SEQA_CTRL	R/W	0x008	ADC Conversion Sequence-A control register: Controls triggering and channel selection for conversion sequence-A. Also specifies interrupt mode for sequence-A.	0x0	28.6.3
SEQB_CTRL	R/W	0x00C	ADC Conversion Sequence-B Control register: Controls triggering and channel selection for conversion sequence-B. Also specifies interrupt mode for sequence-B.	0x0	28.6.4
SEQA_GDAT	RO	0x010	ADC Sequence-A Global Data register. This register contains the result of the most recent ADC conversion performed under sequence-A.	-	28.6.5
SEQB_GDAT	RO	0x014	ADC Sequence-B Global Data register. This register contains the result of the most recent ADC conversion performed under sequence-B.	-	28.6.5
DAT0	RO	0x020	ADC Channel 0 Data register. This register contains the result of the most recent conversion completed on channel 0.	-	28.6.7
DAT1	RO	0x024	ADC Channel 1 Data register. This register contains the result of the most recent conversion completed on channel 1.	-	28.6.7
DAT2	RO	0x028	ADC Channel 2 Data register. This register contains the result of the most recent conversion completed on channel 2.	-	28.6.7
DAT3	RO	0x02C	ADC Channel 3 Data register. This register contains the result of the most recent conversion completed on channel 3.	-	28.6.7
DAT4	RO	0x030	ADC Channel 4 Data register. This register contains the result of the most recent conversion completed on channel 4.	-	28.6.7
DAT5	RO	0x034	ADC Channel 5 Data register. This register contains the result of the most recent conversion completed on channel 5.	-	28.6.7
DAT6	RO	0x038	ADC Channel 6 Data register. This register contains the result of the most recent conversion completed on channel 6.	-	28.6.7
DAT7	RO	0x03C	ADC Channel 7 Data register. This register contains the result of the most recent conversion completed on channel 7.	-	28.6.7
DAT8	RO	0x040	ADC Channel 8 Data register. This register contains the result of the most recent conversion completed on channel 7.	-	28.6.7
DAT9	RO	0x044	ADC Channel 9 Data register. This register contains the result of the most recent conversion completed on channel 7.	-	28.6.7
DAT10	RO	0x048	ADC Channel 10 Data register. This register contains the result of the most recent conversion completed on channel 7.	-	28.6.7
DAT11	RO	0x04C	ADC Channel 11 Data register. This register contains the result of the most recent conversion completed on channel 7.	-	28.6.7

Table 460. Register overview: ADC (base address 0x400A 0000)

Name	Access	Offset	Description	Reset value	Section
THR0_LOW	R/W	0x050	ADC Low Compare Threshold register 0: Contains the lower threshold level for automatic threshold comparison for any channels linked to threshold pair 0.	0x0	28.6.8
THR1_LOW	R/W	0x054	ADC Low Compare Threshold register 1: Contains the lower threshold level for automatic threshold comparison for any channels linked to threshold pair 1.	0x0	28.6.8
THR0_HIGH	R/W	0x058	ADC High Compare Threshold register 0: Contains the upper threshold level for automatic threshold comparison for any channels linked to threshold pair 0.	0x0	28.6.10
THR1_HIGH	R/W	0x05C	ADC High Compare Threshold register 1: Contains the upper threshold level for automatic threshold comparison for any channels linked to threshold pair 1.	0x0	28.6.10
CHAN_ THRSEL	R/W	0x060	ADC Channel-Threshold Select register. Specifies which set of threshold compare registers are to be used for each channel	0x0	28.6.12
INTEN	R/W	0x064	ADC Interrupt Enable register. This register contains enable bits that enable the sequence-A, sequence-B, threshold compare and data overrun interrupts to be generated.	0x0	28.6.13
FLAGS	RO	0x068	ADC Flags register. Contains the four interrupt/DMA trigger flags and the individual component overrun and threshold-compare flags. (The overrun bits replicate information stored in the result registers).	0x0	28.6.14
STARTUP	R/W	0x06C	ADC Startup register.	0x0	28.6.15
CALIB	R/W/RO	0x070	ADC Calibration register.	0x0	28.6.16

28.6.1 ADC Control register

This register specifies the clock divider value to be used to generate the ADC clock **in synchronous mode** and general operating mode bits including resolution and sampling time.

Table 461. ADC Control register (CTRL, offset 0x0) bit description

Bit	Symbol	Value	Description	Reset value
7:0	CLKDIV	-	In synchronous mode only, the system clock is divided by this value plus one to produce the clock for the ADC converter, which should be less than or equal to 80 MHz. Typically, software should program the smallest value in this field that yields this maximum clock rate or slightly less, but in certain cases (such as a high-impedance analog source) a slower clock may be desirable. **Remark:** This field is ignored in the asynchronous operating mode.	0x0
8	ASYNMODE		Select clock mode.	0x0
		0	Synchronous mode. The ADC clock is derived from the system clock based on the divide value selected in the CLKDIV field. The ADC clock will be started in a controlled fashion in response to a trigger to eliminate any uncertainty in the launching of an ADC conversion in response to any synchronous (on-chip) trigger. In Synchronous mode with the SYNCBYPASS bit (in a sequence control register) set, sampling of the ADC input and start of conversion will initiate 2 system clocks after the leading edge of a (synchronous) trigger pulse.	
		1	Asynchronous mode. The ADC clock is based on the output of the ADC clock divider ADCCLKSEL in the SYSCON block.	
10:9	RESOL		The number of bits of ADC resolution. Accuracy can be reduced to achieve higher conversion rates. A single conversion (including one conversion in a burst or sequence) requires the selected number of bits of resolution plus 3 ADC clocks. **Remark:** This field must only be altered when the ADC is fully idle. Changing it during any kind of ADC operation may have unpredictable results. **Remark:** ADC clock frequencies for various resolutions must not exceed: - 5x the system clock rate for 12-bit resolution - 4.3x the system clock rate for 10-bit resolution - 3.6x the system clock for 8-bit resolution - 3x the bus clock rate for 6-bit resolution	0x3
		0x0	6-bit resolution. An ADC conversion requires 9 ADC clocks, plus any clocks specified by the TSAMP field. This accounts for any additional sample clocks if TSAMP is not 0. With TSAMP = 0, the sample rate would be 8.88 Msps with an 80 MHz ADC clock.	
		0x1	8-bit resolution. An ADC conversion requires 11 ADC clocks, plus any clocks specified by the TSAMP field. This accounts for any additional sample clocks if TSAMP is not 0. With TSAMP = 0, the sample rate would be 7.27 Msps with an 80 MHz ADC clock.	
		0x2	10-bit resolution. An ADC conversion requires 13 ADC clocks, plus any clocks specified by the TSAMP field. This accounts for any additional sample clocks if TSAMP is not 0. With TSAMP = 0, the sample rate would be 6.15 Msps with an 80 MHz ADC clock.	
		0x3	12-bit resolution. An ADC conversion requires 15 ADC clocks, plus any clocks specified by the TSAMP field. This accounts for any additional sample clocks if TSAMP is not 0. With TSAMP = 0, the sample rate would be 5.33 Msps with an 80 MHz ADC clock.	

Table 461. ADC Control register (CTRL, offset 0x0) bit description

Bit	Symbol	Value	Description	Reset value
11	BYPASSCAL		Bypass Calibration. This bit may be set to avoid the need to calibrate if offset error is not a concern in the application.	0x0
		0	Calibrate. The stored calibration value will be applied to the ADC during conversions to compensated for offset error. A calibration cycle must be performed each time the chip is powered-up. Re-calibration may be warranted periodically - especially if operating conditions have changed.	
		1	Bypass calibration. Calibration is not utilized. Less time is required when enabling the ADC - particularly following chip power-up. Attempts to launch a calibration cycle are blocked when this bit is set.	
14:12	TSAMP	-	Sample Time. The default sampling period (TSAMP = “000”) at the start of each conversion is 2.5 ADC clock periods. Depending on a variety of factors, including operating conditions and the output impedance of the analog source, longer sampling times may be required. See Section 28.7.10. The TSAMP field specifies the number of additional ADC clock cycles, from zero to seven, by which the sample period will be extended. The total conversion time will increase by the same number of clocks. 000 - The sample period will be the default 2.5 ADC clocks. A complete conversion with 12-bits of accuracy will require 15 clocks. 001- The sample period will be extended by one ADC clock to a total of 3.5 clock periods. A complete 12-bit conversion will require 16 clocks. 010 - The sample period will be extended by two clocks to 4.5 ADC clock cycles. A complete 12-bit conversion will require 17 ADC clocks. ... 111 - The sample period will be extended by seven clocks to 9.5 ADC clock cycles. A complete 12-bit conversion will require 22 ADC clocks.	0x0
31:15	-	-	Reserved. Read value is undefined, only zero should be written.	-

28.6.2 Input Select register

This register allows the temperature sensor to be used as an alternate input for ADC channel 0.

Table 462. Input Select register (INSEL, offset 0x04) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SEL		Selects the input source for channel 0. All other values are reserved.	0x0
		0x0	ADC0_IN0 function.	
		0x3	Internal temperature sensor.	
31:2	-	-	Reserved. Read value is undefined, only zero should be written.	0x0

28.6.3 ADC Conversion Sequence A Control register

There are two independent conversion sequences that can be configured, each consisting of a set of conversions on one or more channels. This control register specifies the channel selection and trigger conditions for the A sequence and contains bits to allow software to initiate that conversion sequence.

Remark: Set the BURST and SEQU_ENA bits at the same time.

Table 463: ADC Conversion Sequence A Control register (SEQA_CTRL, offset 0x08) bit description

Bit	Symbol	Value	Description	Reset value
11:0	CHANNELS	-	Selects which one or more of the ADC channels will be sampled and converted when this sequence is launched. A 1 in any bit of this field will cause the corresponding channel to be included in the conversion sequence, where bit 0 corresponds to channel 0, bit 1 to channel 1 and so forth. When this conversion sequence is triggered, either by a hardware trigger or via software command, ADC conversions will be performed on each enabled channel, in sequence, beginning with the lowest-ordered channel. **Remark:** This field can ONLY be changed while SEQA_ENA (bit 31) is LOW. It is allowed to change this field and set bit 31 in the same write.	0x00
17:12	TRIGGER	-	Selects which of the available hardware trigger sources will cause this conversion sequence to be initiated. Program the trigger input number in this field. See Table 477. **Remark:** In order to avoid generating a spurious trigger, it is recommended writing to this field only when SEQA_ENA (bit 31) is low. It is safe to change this field and set bit 31 in the same write.	0x0
18	TRIGPOL		Select the polarity of the selected input trigger for this conversion sequence. **Remark:** In order to avoid generating a spurious trigger, it is recommended writing to this field only when SEQA_ENA (bit 31) is low. It is safe to change this field and set bit 31 in the same write.	0x0
		0	Negative edge. A negative edge launches the conversion sequence on the selected trigger input.	
		1	Positive edge. A positive edge launches the conversion sequence on the selected trigger input.	
19	SYNCBYPASS		Setting this bit allows the hardware trigger input to bypass synchronization flip-flop stages and therefore shorten the time between the trigger input signal and the start of a conversion. There are slightly different criteria for whether or not this bit can be set depending on the clock operating mode: Synchronous mode (the ASYNMODE in the CTRL register = 0): Synchronization may be bypassed (this bit may be set) if the selected trigger source is already synchronous with the main system clock (eg. coming from an on-chip, system-clock-based timer). Whether this bit is set or not, a trigger pulse must be maintained for at least one system clock period. Asynchronous mode (the ASYNMODE in the CTRL register = 1): Synchronization may be bypassed (this bit may be set) if it is certain that the duration of a trigger input pulse will be at least one cycle of the ADC clock (regardless of whether the trigger comes from and on-chip or off-chip source). If this bit is NOT set, the trigger pulse must at least be maintained for one system clock period.	0x0
		0	Enable trigger synchronization. The hardware trigger bypass is not enabled.	
		1	Bypass trigger synchronization. The hardware trigger bypass is enabled.	
25:20	-	-	Reserved. Read value is undefined, only zero should be written.	N/A

Table 463: ADC Conversion Sequence A Control register (SEQA_CTRL, offset 0x08) bit description

Bit	Symbol	Value	Description	Reset value
26	START	-	Writing a 1 to this field will launch one pass through this conversion sequence. The behavior will be identical to a sequence triggered by a hardware trigger. Do not write 1 to this bit if the BURST bit is set. **Remark:** This bit is only set to a 1 momentarily when written to launch a conversion sequence. It will consequently always read back as a zero.	0x0
27	BURST	-	Writing a 1 to this bit will cause this conversion sequence to be continuously cycled through. Other sequence A triggers will be ignored while this bit is set. Repeated conversions can be halted by clearing this bit. The sequence currently in progress will be completed before conversions are terminated. Note that a new sequence could begin just before BURST is cleared.	0x0
28	SINGLESTEP	-	When this bit is set, a hardware trigger or a write to the START bit will launch a single conversion on the next channel in the sequence instead of the default response of launching an entire sequence of conversions. Once all of the channels comprising a sequence have been converted, a subsequent trigger will repeat the sequence beginning with the first enabled channel. Interrupt generation will still occur either after each individual conversion or at the end of the entire sequence, depending on the state of the MODE bit.	0x0
29	LOWPRIO		Set priority for sequence A. Sequence A may be set to low priority, allowing sequence B to interrupt it if triggered while Sequence A is running.	0x0
		0	High priority. Sequence B will not interrupt sequence A. Any sequence B trigger which occurs while an A conversion sequence is active will be ignored and lost.	
		1	Low priority. Any enabled B sequence trigger (including a B sequence software start) will immediately interrupt sequence A and launch a B sequence in it's place. If a conversion is currently in progress, it will be terminated. The A sequence that was interrupted will automatically resume after the B sequence completes. The channel whose conversion was terminated will be re-sampled and the conversion sequence will resume from that point.	
30	MODE		Indicates whether the primary method for retrieving conversion results for this sequence will be accomplished via reading the global data register (SEQA_GDAT) at the end of each conversion, or the individual channel result registers at the end of the entire sequence. Impacts when conversion-complete interrupt/DMA trigger for sequence-A will be generated and which overrun conditions contribute to an overrun interrupt as described below.	0x0
		0	End of conversion. The sequence A interrupt/DMA trigger will be set at the end of each individual ADC conversion performed under sequence A. This flag will mirror the DATAVALID bit in the SEQA_GDAT register. The OVERRUN bit in the SEQA_GDAT register will contribute to generation of an overrun interrupt/DMA trigger if enabled.	
		1	End of sequence. The sequence A interrupt/DMA trigger will be set when the entire set of sequence-A conversions completes. This flag will need to be explicitly cleared by software or by the DMA-clear signal in this mode. The OVERRUN bit in the SEQA_GDAT register will NOT contribute to generation of an overrun interrupt/DMA trigger since it is assumed this register may not be utilized in this mode.	

Table 463: ADC Conversion Sequence A Control register (SEQA_CTRL, offset 0x08) bit description

Bit	Symbol	Value	Description	Reset value
31	SEQA_ENA		Sequence Enable. **Remark:** In order to avoid spuriously triggering the sequence, care should be taken to only set the SEQA_ENA bit when the selected trigger input is in its INACTIVE state (as defined by the TRIGPOL bit). If this condition is not met, the sequence will be triggered immediately upon being enabled.	0x0
		0	Disabled. Sequence A is disabled. Sequence A triggers are ignored. If this bit is cleared while sequence A is in progress, the sequence will be halted at the end of the current conversion. After the sequence is re-enabled, a new trigger will be required to restart the sequence beginning with the next enabled channel.	
		1	Enabled. Sequence A is enabled.	

28.6.4 ADC Conversion Sequence B Control register

There are two independent conversion sequences that can be configured, each consisting of a set of conversions on one or more channels. This control register specifies the channel selection and trigger conditions for the B sequence, as well bits to allow software to initiate that conversion sequence.

Table 464: ADC Conversion Sequence B Control register (SEQB_CTRL, offset 0x0C) bit description

Bit	Symbol	Value	Description	Reset value
11:0	CHANNELS	-	Selects which one or more of the ADC channels will be sampled and converted when this sequence is launched. A 1 in any bit of this field will cause the corresponding channel to be included in the conversion sequence, where bit 0 corresponds to channel 0, bit 1 to channel 1 and so forth. When this conversion sequence is triggered, either by a hardware trigger or via software command, ADC conversions will be performed on each enabled channel, in sequence, beginning with the lowest-ordered channel. **Remark:** This field can ONLY be changed while SEQB_ENA (bit 31) is LOW. It is allowed to change this field and set bit 31 in the same write.	0x00
17:12	TRIGGER	-	Selects which of the available hardware trigger sources will cause this conversion sequence to be initiated. Program the trigger input number in this field. See Table 477. **Remark:** To avoid generating a spurious trigger, it is recommended writing to this field only when SEQB_ENA (bit 31) is low. It is safe to change this field and set bit 31 in the same write.	0x0
18	TRIGPOL		Select the polarity of the selected input trigger for this conversion sequence. **Remark:** To avoid generating a spurious trigger, it is recommended writing to this field only when SEQB_ENA (bit 31) is low. It is safe to change this field and set bit 31 in the same write.	0x0
		0	Negative edge. A negative edge launches the conversion sequence on the selected trigger input.	
		1	Positive edge. A positive edge launches the conversion sequence on the selected trigger input.	
19	SYNCBYPASS		Setting this bit allows the hardware trigger input to bypass synchronization flip-flop stages and therefore shorten the time between the trigger input signal and the start of a conversion. There are slightly different criteria for whether or not this bit can be set depending on the clock operating mode: Synchronous mode (the ASYNMODE in the CTRL register = 0): Synchronization may be bypassed (this bit may be set) if the selected trigger source is already synchronous with the main system clock (eg. coming from an on-chip, system-clock-based timer). Whether this bit is set or not, a trigger pulse must be maintained for at least one system clock period. Asynchronous mode (the ASYNMODE in the CTRL register = 1): Synchronization may be bypassed (this bit may be set) if it is certain that the duration of a trigger input pulse will be at least one cycle of the ADC clock (regardless of whether the trigger comes from and on-chip or off-chip source). If this bit is NOT set, the trigger pulse must at least be maintained for one system clock period.	0x0
		0	Enable synchronization. The hardware trigger bypass is not enabled.	
		1	Bypass synchronization. The hardware trigger bypass is enabled.	
25:20	-	-	Reserved. Read value is undefined, only zero should be written.	N/A

Table 464: ADC Conversion Sequence B Control register (SEQB_CTRL, offset 0x0C) bit description

Bit	Symbol	Value	Description	Reset value
26	START	-	Writing a 1 to this field will launch one pass through this conversion sequence. The behavior will be identical to a sequence triggered by a hardware trigger. Do not write 1 to this bit if the BURST bit is set. **Remark:** This bit is only set to a 1 momentarily when written to launch a conversion sequence. It will consequently always read back as a zero.	0x0
27	BURST	-	Writing a 1 to this bit will cause this conversion sequence to be continuously cycled through. Other sequence B triggers will be ignored while this bit is set. Repeated conversions can be halted by clearing this bit. The sequence currently in progress will be completed before conversions are terminated.	0x0
28	SINGLESTEP	-	When this bit is set, a hardware trigger or a write to the START bit will launch a single conversion on the next channel in the sequence instead of the default response of launching an entire sequence of conversions. Once all of the channels comprising a sequence have been converted, a subsequent trigger will repeat the sequence beginning with the first enabled channel. Interrupt generation will still occur either after each individual conversion or at the end of the entire sequence, depending on the state of the MODE bit.	0x0
29	-	-	Reserved. Read value is undefined, only zero should be written.	N/A
30	MODE		Indicates whether the primary method for retrieving conversion results for this sequence will be accomplished via reading the global data register (SEQB_GDAT) at the end of each conversion, or the individual channel result registers at the end of the entire sequence. Impacts when conversion-complete interrupt/DMA trigger for sequence-B will be generated and which overrun conditions contribute to an overrun interrupt as described below.	0x0
		0	End of conversion. The sequence B interrupt/DMA trigger will be set at the end of each individual ADC conversion performed under sequence B. This flag will mirror the DATAVALID bit in the SEQB_GDAT register. The OVERRUN bit in the SEQB_GDAT register will contribute to generation of an overrun interrupt/DMA trigger if enabled.	
		1	End of sequence. The sequence B interrupt/DMA trigger will be set when the entire set of sequence-B conversions completes. This flag will need to be explicitly cleared by software or by the DMA-clear signal in this mode. The OVERRUN bit in the SEQB_GDAT register will NOT contribute to generation of an overrun interrupt/DMA trigger since it is assumed this register may not be utilized in this mode.	
31	SEQB_ENA		Sequence Enable. **Remark:** In order to avoid spuriously triggering the sequence, care should be taken to only set the SEQB_ENA bit when the selected trigger input is in its INACTIVE state (as defined by the TRIGPOL bit). If this condition is not met, the sequence will be triggered immediately upon being enabled.	0x0
		0	Disabled. Sequence B is disabled. Sequence B triggers are ignored. If this bit is cleared while sequence B is in progress, the sequence will be halted at the end of the current conversion. After the sequence is re-enabled, a new trigger will be required to restart the sequence beginning with the next enabled channel.	
		1	Enabled. Sequence B is enabled.	

28.6.5 ADC Global Data register A

The ADC Global Data registers contain the result of the most recent ADC conversion completed under each conversion sequence.

Results of ADC conversions can be read in one of two ways. One is to use these ADC Global Data registers to read data from the ADC at the end of each ADC conversion. Another is to read the individual ADC Channel Data registers, typically after the entire sequence has completed. It is recommended to use one method consistently for a given conversion sequence.

The global registers are useful in conjunction with DMA operation - particularly when the channels selected for conversion are not sequential (hence the addresses of the individual result registers will not be sequential, making it difficult for the DMA engine to address them). For interrupt-driven code it will more likely be advantageous to wait for an entire sequence to complete and then retrieve the results from the individual channel registers.

Remark: The method to be employed for each sequence should be reflected in the MODE bit in the corresponding SEQn_CTRL register since this will impact interrupt and overrun flag generation.

Table 465: ADC Sequence A Global Data register (SEQA_GDAT, offset 0x10) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved.	-
15:4	RESULT	This field contains the 12-bit ADC conversion result from the most recent conversion performed under conversion sequence associated with this register. The result is a binary fraction representing the voltage on the currently-selected input channel as it falls within the range of VREFP to VREFN. Zero in the field indicates that the voltage on the input pin was less than, equal to, or close to that on VREFN, while 0xFFF indicates that the voltage on the input was close to, equal to, or greater than that on VREFP. DATAVALID = 1 indicates that this result has not yet been read.	-
17:16	THCMPRANGE	Indicates whether the result of the last conversion performed was above, below or within the range established by the designated threshold comparison registers (THRn_LOW and THRn_HIGH).	-
19:18	THCMPCROSS	Indicates whether the result of the last conversion performed represented a crossing of the threshold level established by the designated LOW threshold comparison register (THRn_LOW) and, if so, in what direction the crossing occurred.	-
25:20	-	Reserved.	-
29:26	CHN	These bits contain the channel from which the RESULT bits were converted (e.g. 0000 identifies channel 0, 0001 channel 1, etc.).	-
30	OVERRUN	This bit is set if a new conversion result is loaded into the RESULT field before a previous result has been read - i.e. while the DATAVALID bit is set. This bit is cleared, along with the DATAVALID bit, whenever this register is read. This bit will contribute to an overrun interrupt/DMA trigger if the MODE bit (in SEQAA_CTRL) for the corresponding sequence is set to ‘0’ (and if the overrun interrupt is enabled).	0x0
31	DATAVALID	This bit is set to ‘1’ at the end of each conversion when a new result is loaded into the RESULT field. It is cleared whenever this register is read. This bit will cause a conversion-complete interrupt for the corresponding sequence if the MODE bit (in SEQA_CTRL) for that sequence is set to 0 (and if the interrupt is enabled).	0x0

28.6.6 ADC Global Data register B

See the description of ADC Global Data register A in Section 28.6.3.

Table 466: ADC Sequence B Global Data register (SEQB_GDAT, offset 0x14) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved.	-
15:4	RESULT	This field contains the 12-bit ADC conversion result from the most recent conversion performed under conversion sequence associated with this register. The result is a binary fraction representing the voltage on the currently-selected input channel as it falls within the range of VREFP to VREFN. Zero in the field indicates that the voltage on the input pin was less than, equal to, or close to that on VREFN, while 0xFFF indicates that the voltage on the input was close to, equal to, or greater than that on VREFP. DATAVALID = 1 indicates that this result has not yet been read.	-
17:16	THCMPRANGE	Indicates whether the result of the last conversion performed was above, below or within the range established by the designated threshold comparison registers (THRn_LOW and THRn_HIGH).	
19:18	THCMPCROSS	Indicates whether the result of the last conversion performed represented a crossing of the threshold level established by the designated LOW threshold comparison register (THRn_LOW) and, if so, in what direction the crossing occurred.	
25:20	-	Reserved.	-
29:26	CHN	These bits contain the channel from which the RESULT bits were converted (e.g. 0000 identifies channel 0, 0001 channel 1, etc.).	-
30	OVERRUN	This bit is set if a new conversion result is loaded into the RESULT field before a previous result has been read - i.e. while the DATAVALID bit is set. This bit is cleared, along with the DATAVALID bit, whenever this register is read. This bit will contribute to an overrun interrupt/DMA trigger if the MODE bit (in SEQB_CTRL) for the corresponding sequence is set to ‘0’ (and if the overrun interrupt is enabled).	0x0
31	DATAVALID	This bit is set to ‘1’ at the end of each conversion when a new result is loaded into the RESULT field. It is cleared whenever this register is read. This bit will cause a conversion-complete interrupt for the corresponding sequence if the MODE bit (in SEQB_CTRL) for that sequence is set to 0 (and if the interrupt is enabled).	0x0

28.6.7 ADC Channel Data registers 0 to 11

The ADC Channel Data registers hold the result of the last conversion completed for each ADC channel. They also include status bits to indicate when a conversion has been completed, when a data overrun has occurred, and where the most recent conversion fits relative to the range dictated by the high and low threshold registers.

Results of ADC conversion can be read in one of two ways. One is to use the ADC Global Data registers for each of the sequences to read data from the ADC at the end of each ADC conversion. Another is to use these individual ADC Channel Data registers, typically after the entire sequence has completed. It is recommended to use one method consistently for a given conversion sequence.

Remark: The method to be employed for each sequence should be reflected in the MODE bit in the corresponding SEQ_CTRL register since this will impact interrupt and overrun flag generation.

The information presented in the DAT registers always pertains to the most recent conversion completed on that channel regardless of what sequence requested the conversion or which trigger caused it.

The OVERRUN fields for each channel are also replicated in the FLAGS register.

Table 467. ADC Data registers (DAT[0:11], offset [0x020:0x04C]) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved.	-
15:4	RESULT	This field contains the 12-bit ADC conversion result from the last conversion performed on this channel. This will be a binary fraction representing the voltage on the AD0[n] pin, as it falls within the range of V_{REFP} to V_{REFN}. Zero in the field indicates that the voltage on the input pin was less than, equal to, or close to that on V_{REFN}, while 0xFFF indicates that the voltage on the input was close to, equal to, or greater than that on V_{REFP}.	-
17:16	THCMPRANGE	Threshold Range Comparison result. 0x0 = In Range: The last completed conversion was greater than or equal to the value programmed into the designated LOW threshold register (THRn_LOW) but less than or equal to the value programmed into the designated HIGH threshold register (THRn_HIGH). 0x1 = Below Range: The last completed conversion on was less than the value programmed into the designated LOW threshold register (THRn_LOW). 0x2 = Above Range: The last completed conversion was greater than the value programmed into the designated HIGH threshold register (THRn_HIGH). 0x3 = Reserved.	-

Table 467. ADC Data registers (DAT[0:11], offset [0x020:0x04C]) bit description

Bit	Symbol	Description	Reset value
19:18	THCMPCROSS	Threshold Crossing Comparison result. 0x0 = No threshold Crossing detected: The most recent completed conversion on this channel had the same relationship (above or below) to the threshold value established by the designated LOW threshold register (THRn_LOW) as did the previous conversion on this channel. 0x1 = Reserved. 0x2 = Downward Threshold Crossing Detected. Indicates that a threshold crossing in the downward direction has occurred - i.e. the previous sample on this channel was above the threshold value established by the designated LOW threshold register (THRn_LOW) and the current sample is below that threshold. 0x3 = Upward Threshold Crossing Detected. Indicates that a threshold crossing in the upward direction has occurred - i.e. the previous sample on this channel was below the threshold value established by the designated LOW threshold register (THRn_LOW) and the current sample is above that threshold.	-
25:20	-	Reserved. Read value is undefined, only zero should be written.	-
29:26	CHANNEL	This field is hard-coded to contain the channel number that this particular register relates to (i.e. this field will contain 0b0000 for the DAT0 register, 0b0001 for the DAT1 register, etc)	-
30	OVERRUN	This bit will be set to a 1 if a new conversion on this channel completes and overwrites the previous contents of the RESULT field before it has been read - i.e. while the DONE bit is set. This bit is cleared, along with the DONE bit, whenever this register is read or when the data related to this channel is read from either of the global SEQn_GDAT registers. This bit (in any of the 12 registers) will cause an overrun interrupt/DMA trigger to be asserted if the overrun interrupt is enabled. **Remark:** While it is allowed to include the same channels in both conversion sequences, doing so may cause erratic behavior of the DONE and OVERRUN bits in the data registers associated with any of the channels that are shared between the two sequences. Any erratic OVERRUN behavior will also affect overrun interrupt generation, if enabled.	-
31	DATAVALID	This bit is set to 1 when an ADC conversion on this channel completes. This bit is cleared whenever this register is read or when the data related to this channel is read from either of the global SEQn_GDAT registers. **Remark:** While it is allowed to include the same channels in both conversion sequences, doing so may cause erratic behavior of the DONE and OVERRUN bits in the data registers associated with any of the channels that are shared between the two sequences. Any erratic OVERRUN behavior will also affect overrun interrupt generation, if enabled.	-

28.6.8 ADC Compare Low Threshold register 0

These registers set the LOW threshold levels against which ADC conversions on all channels will be compared.

Each channel will either be compared to the THR0_LOW/HIGH registers or to the THR1_LOW/HIGH registers depending on what is specified for that channel in the CHAN_THRSEL register.

A conversion result LESS THAN this value on any channel will cause the THCMP_RANGE status bits for that channel to be set to 0b01. This result will also generate an interrupt/DMA trigger if enabled to do so via the ADCMPINTEN bits associated with each channel in the INTEN register.

If, for two successive conversions on a given channel, one result is below this threshold and the other is equal-to or above this threshold, than a threshold crossing has occurred. In this case the MSB of the THCMP_CROSS status bits will indicate that a threshold crossing has occurred and the LSB will indicate the direction of the crossing. A threshold crossing event will also generate an interrupt/DMA trigger if enabled to do so via the ADCMPINTEN bits associated with each channel in the INTEN register.

Table 468. ADC Compare Low Threshold register 0 (THR0_LOW, offset 0x50) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved. Read value is undefined, only zero should be written.	-
15:4	THRLOW	Low threshold value against which ADC results will be compared	0x000
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

28.6.9 ADC Compare Low Threshold register 1

See the description of ADC Compare Low Threshold register 0 in Section 28.6.8.

Table 469. ADC Compare Low Threshold register 1 (THR1_LOW, offset 0x54) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved. Read value is undefined, only zero should be written.	-
15:4	THRLOW	Low threshold value against which ADC results will be compared	0x000
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

28.6.10 ADC Compare High Threshold register 0

These registers set the HIGH threshold level against which ADC conversions on all channels will be compared.

Each channel will either be compared to the THR0_LOW/HIGH registers or to the THR1_LOW/HIGH registers depending on what is specified for that channel in the CHAN_THRSEL register.

A conversion result greater than this value on any channel will cause the THCMP status bits for that channel to be set to 0b10. This result will also generate an interrupt/DMA trigger if enabled to do so via the ADCMPINTEN bits associated with each channel in the INTEN register.

Table 470: Compare High Threshold register0 (THR0_HIGH, offset 0x58) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved. Read value is undefined, only zero should be written.	-
15:4	THRHIGH	High threshold value against which ADC results will be compared	0x000
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

28.6.11 ADC Compare High Threshold register 1

See the description of ADC Compare High Threshold register 0 in Section 28.6.10.

Table 471: Compare High Threshold register 1 (THR1_HIGH, offset 0x5C) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved. Read value is undefined, only zero should be written.	-
15:4	THRHIGH	High threshold value against which ADC results will be compared	0x000
31:16	-	Reserved. Read value is undefined, only zero should be written.	-

28.6.12 ADC Channel Threshold Select register

For each channel, this register indicates which pair of threshold registers conversion results should be compared to.

Table 472: ADC Channel Threshold Select register (CHAN_THRSEL, offset 0x60) bit description

Bit	Symbol	Value	Description	Reset value
0	CH0_THRSEL		Threshold select for channel 0.	0x0
		0	Threshold 0. Results for this channel will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers.	
		1	Threshold 1. Results for this channel will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers.	
1	CH1_THRSEL	-	Threshold select for channel 1. See description for channel 0.	0x0
2	CH2_THRSEL	-	Threshold select for channel 2. See description for channel 0.	0x0
3	CH3_THRSEL	-	Threshold select for channel 3. See description for channel 0.	0x0
4	CH4_THRSEL	-	Threshold select for channel 4. See description for channel 0.	0x0
5	CH5_THRSEL	-	Threshold select for channel 5. See description for channel 0.	0x0
6	CH6_THRSEL	-	Threshold select for channel 6. See description for channel 0.	0x0
7	CH7_THRSEL	-	Threshold select for channel 7. See description for channel 0.	0x0
8	CH8_THRSEL	-	Threshold select for channel 8. See description for channel 0.	0x0
9	CH9_THRSEL	-	Threshold select for channel 9. See description for channel 0.	0x0
10	CH10_THRSEL	-	Threshold select for channel 10. See description for channel 0.	0x0
11	CH11_THRSEL	-	Threshold select for channel 11. See description for channel 0.	0x0
31:12	-	-	Reserved. Read value is undefined, only zero should be written.	-

28.6.13 ADC Interrupt Enable register

There are four separate interrupt requests generated by the ADC: conversion, these are -complete or sequence-complete interrupts for each of the two sequences, a threshold-comparison out-of-range interrupt, and a data overrun interrupt. The two conversion/sequence-complete interrupts can also serve as DMA triggers. The threshold and data overrun interrupts share a slot in the NVIC.

These interrupts may be combined into one request on some chips if there is a limited number of interrupt slots. This register contains the interrupt-enable bits for each interrupt.

In this register, threshold events selected in the ADCMPINTENn bits are described as follows:

- Disabled: Threshold comparisons on channel n will not generate an ADC threshold-compare interrupt/DMA trigger.
- Outside threshold: A conversion result on channel n which is outside the range specified by the designated HIGH and LOW threshold registers will set the channel n THCMP flag in the FLAGS register and generate an ADC threshold-compare interrupt/DMA trigger.
- Crossing threshold: Detection of a threshold crossing on channel n will set the channel n THCMP flag in the FLAGS register and generate an ADC threshold-compare interrupt/DMA trigger.

Remark: Overrun and threshold-compare interrupts related to a particular channel will occur regardless of which sequence was in progress at the time the conversion was performed or what trigger caused the conversion.

Table 473: ADC Interrupt Enable register (INTEN, offset 0x64) bit description

Bit	Symbol	Value	Description	Reset value
0	SEQA_INTEN		Sequence A interrupt enable.	0x0
		0	Disabled. The sequence A interrupt/DMA trigger is disabled.	
		1	Enabled. The sequence A interrupt/DMA trigger is enabled and will be asserted either upon completion of each individual conversion performed as part of sequence A, or upon completion of the entire A sequence of conversions, depending on the MODE bit in the SEQA_CTRL register.	
1	SEQB_INTEN		Sequence B interrupt enable.	0x0
		0	Disabled. The sequence B interrupt/DMA trigger is disabled.	
		1	Enabled. The sequence B interrupt/DMA trigger is enabled and will be asserted either upon completion of each individual conversion performed as part of sequence B, or upon completion of the entire B sequence of conversions, depending on the MODE bit in the SEQB_CTRL register.	
2	OVR_INTEN		Overrun interrupt enable.	0x0
		0	Disabled. The overrun interrupt is disabled.	
		1	Enabled. The overrun interrupt is enabled. Detection of an overrun condition on any of the 12 channel data registers will cause an overrun interrupt/DMA trigger. In addition, if the MODE bit for a particular sequence is 0, then an overrun in the global data register for that sequence will also cause this interrupt/DMA trigger to be asserted.	

Table 473: ADC Interrupt Enable register (INTEN, offset 0x64) bit description

Bit	Symbol	Value	Description	Reset value
4:3	ADCMPINTEN0		Threshold comparison interrupt enable for channel 0.	0x0
		0x0	Disabled.	
		0x1	Outside threshold.	
		0x2	Crossing threshold.	
		0x3	Reserved	
6:5	ADCMPINTEN1	-	Channel 1 threshold comparison interrupt enable. See description for channel 0.	0x0
8:7	ADCMPINTEN2	-	Channel 2 threshold comparison interrupt enable. See description for channel 0.	0x0
10:9	ADCMPINTEN3	-	Channel 3 threshold comparison interrupt enable. See description for channel 0.	0x0
12:11	ADCMPINTEN4	-	Channel 4 threshold comparison interrupt enable. See description for channel 0.	0x0
14:13	ADCMPINTEN5	-	Channel 5 threshold comparison interrupt enable. See description for channel 0.	0x0
16:15	ADCMPINTEN6	-	Channel 6 threshold comparison interrupt enable. See description for channel 0.	0x0
18:17	ADCMPINTEN7	-	Channel 7 threshold comparison interrupt enable. See description for channel 0.	0x0
20:19	ADCMPINTEN8	-	Channel 8 threshold comparison interrupt enable. See description for channel 0.	0x0
22:21	ADCMPINTEN9	-	Channel 9 threshold comparison interrupt enable. See description for channel 0.	0x0
24:23	ADCMPINTEN10	-	Channel 10 threshold comparison interrupt enable. See description for channel 0.	0x0
26:25	ADCMPINTEN11	-	Channel 21 threshold comparison interrupt enable. See description for channel 0.	0x0
31:27	-	-	Reserved. Read value is undefined, only zero should be written.	-

28.6.14 ADC Flags register

The ADC Flags registers contains the four interrupt/DMA trigger flags along with the individual overrun flags that contribute to an overrun interrupt and the component threshold-comparison flags that contribute to that interrupt. Note that the threshold and overrun interrupts hare a slot in the NVIC.

The channel OVERRUN flags mirror those appearing in the individual DAT registers for each channel and indicate a data overrun in each of those registers.

Likewise, the SEQA_OVR and SEQB_OVR bits mirror the OVERRUN bits in the two global data registers (SEQA_GDAT and SEQB_GDAT).

Remark: The SEQn_INT conversion/sequence-complete flags also serve as DMA triggers.

Table 474: ADC Flags register (FLAGS, offset 0x68) bit description

Bit	Symbol	Description	Reset value
0	THCMP0	Threshold comparison event on Channel 0. Set to 1 upon either an out-of-range result or a threshold-crossing result if enabled to do so in the INTEN register. This bit is cleared by writing a 1.	0x0
1	THCMP1	Threshold comparison event on Channel 1. See description for channel 0.	0x0
2	THCMP2	Threshold comparison event on Channel 2. See description for channel 0.	0x0
3	THCMP3	Threshold comparison event on Channel 3. See description for channel 0.	0x0
4	THCMP4	Threshold comparison event on Channel 4. See description for channel 0.	0x0
5	THCMP5	Threshold comparison event on Channel 5. See description for channel 0.	0x0
6	THCMP6	Threshold comparison event on Channel 6. See description for channel 0.	0x0
7	THCMP7	Threshold comparison event on Channel 7. See description for channel 0.	0x0
8	THCMP8	Threshold comparison event on Channel 8. See description for channel 0.	0x0
9	THCMP9	Threshold comparison event on Channel 9. See description for channel 0.	0x0
10	THCMP10	Threshold comparison event on Channel 10. See description for channel 0.	0x0
11	THCMP11	Threshold comparison event on Channel 11. See description for channel 0.	0x0
12	OVERRUN0	Mirrors the OVERRRUN status flag from the result register for ADC channel 0	0x0
13	OVERRUN1	Mirrors the OVERRRUN status flag from the result register for ADC channel 1	0x0
14	OVERRUN2	Mirrors the OVERRRUN status flag from the result register for ADC channel 2	0x0
15	OVERRUN3	Mirrors the OVERRRUN status flag from the result register for ADC channel 3	0x0
16	OVERRUN4	Mirrors the OVERRRUN status flag from the result register for ADC channel 4	0x0
17	OVERRUN5	Mirrors the OVERRRUN status flag from the result register for ADC channel 5	0x0
18	OVERRUN6	Mirrors the OVERRRUN status flag from the result register for ADC channel 6	0x0
19	OVERRUN7	Mirrors the OVERRRUN status flag from the result register for ADC channel 7	0x0
20	OVERRUN8	Mirrors the OVERRRUN status flag from the result register for ADC channel 8	0x0
21	OVERRUN9	Mirrors the OVERRRUN status flag from the result register for ADC channel 9	0x0
22	OVERRUN10	Mirrors the OVERRRUN status flag from the result register for ADC channel 10	0x0
23	OVERRUN11	Mirrors the OVERRRUN status flag from the result register for ADC channel 11	0x0
24	SEQA_OVR	Mirrors the global OVERRUN status flag in the SEQA_GDAT register	0x0
25	SEQB_OVR	Mirrors the global OVERRUN status flag in the SEQB_GDAT register	0x0
27:26	-	Reserved.	-

Table 474: ADC Flags register (FLAGS, offset 0x68) bit description

Bit	Symbol	Description	Reset value
28	SEQA_INT	Sequence A interrupt/DMA trigger. If the MODE bit in the SEQA_CTRL register is 0, this flag will mirror the DATAVALID bit in the sequence A global data register (SEQA_GDAT), which is set at the end of every ADC conversion performed as part of sequence A. It will be cleared automatically when the SEQA_GDAT register is read. If the MODE bit in the SEQA_CTRL register is 1, this flag will be set upon completion of an entire A sequence. In this case it must be cleared by writing a 1 to this SEQA_INT bit. This interrupt must be enabled in the INTEN register.	0x0
29	SEQB_INT	Sequence A interrupt/DMA trigger. If the MODE bit in the SEQB_CTRL register is 0, this flag will mirror the DATAVALID bit in the sequence A global data register (SEQB_GDAT), which is set at the end of every ADC conversion performed as part of sequence B. It will be cleared automatically when the SEQB_GDAT register is read. If the MODE bit in the SEQB_CTRL register is 1, this flag will be set upon completion of an entire B sequence. In this case it must be cleared by writing a 1 to this SEQB_INT bit. This interrupt must be enabled in the INTEN register.	0x0
30	THCMP_INT	Threshold Comparison Interrupt. This bit will be set if any of the THCMP flags in the lower bits of this register are set to 1 (due to an enabled out-of-range or threshold-crossing event on any channel). Each type of threshold comparison interrupt on each channel must be individually enabled in the INTEN register to cause this interrupt. This bit will be cleared when all of the individual threshold flags are cleared via writing 1s to those bits.	0x0
31	OVR_INT	Overrun Interrupt flag. Any overrun bit in any of the individual channel data registers will cause this interrupt. In addition, if the MODE bit in either of the SEQn_CTRL registers is 0 then the OVERRUN bit in the corresponding SEQn_GDAT register will also cause this interrupt. This interrupt must be enabled in the INTEN register. This bit will be cleared when all of the individual overrun bits have been cleared via reading the corresponding data registers.	0x0

28.6.15 ADC Startup register

This register is used exclusively when enabling the ADC. This register should never be accessed during normal ADC operation. The ADC clock should be selected and running at full frequency prior to writing to this register.

Table 475: ADC Startup register (STARTUP, offset 0x6C) bit description

Bit	Symbol	Description	Reset value
0	ADC_ENA	ADC Enable bit. This bit can only be set to a 1 by software. It is cleared automatically whenever the ADC is powered down. This bit must not be set until at least 10 microseconds after the ADC is powered up (typically by altering a system-level ADC power control bit).	0x0
1	ADC_INIT	ADC Initialization. After enabling the ADC (setting the ADC_ENA bit), the API calibration function will EITHER set this bit or the CALIB bit in the CALIB register, depending on whether or not calibration is required. Setting this bit will launch the “dummy” conversion cycle that is required if a calibration is not performed. It will also reload the stored calibration value from a previous calibration unless the BYPASSCAL bit is set. This bit should only be set AFTER the ADC_ENA bit is set and after the CALIREQD bit is tested to determine whether a calibration or an ADC dummy conversion cycle is required. It should not be set during the same write that sets the ADC_ENA bit. This bit can only be set to a ‘1’ by software. It is cleared automatically when the “dummy” conversion cycle completes.	0x0
31:2	-	Reserved. Read value is undefined, only zero should be written.	0x0

28.6.16 ADC Calibration register

This register is used to perform ADC offset calibration. **The maximum ADC clock frequency during calibration is 30 MHz**. If the operating ADC frequency exceeds this, a slower clock should be selected for calibration (eg. increasing the synchronous divided clock value). See Section 28.7.6.

Table 476: ADC Calibration register (CALIB, offset 0x70) bit description

Bit	Symbol	Description	Reset value
0	CALIB	Calibration request. Setting this bit will launch an ADC calibration cycle. This bit can only be set to a ‘1’ by software. It is cleared automatically when the calibration cycle completes.	0x0
1	CALREQD	Calibration required. This read-only bit indicates if calibration is required when enabling the ADC. CALREQD will be ‘1’ if no calibration has been run since the chip was powered-up and if the BYPASSCAL bit in the CTRL register is low. Software will test this bit to determine whether to initiate a calibration cycle or whether to set the ADC_INIT bit (in the STARTUP register) to launch the ADC initialization process which includes a “dummy” conversion cycle. Note: A “dummy” conversion cycle requires approximately 6 ADC clocks as opposed to 81 clocks required for calibration.	0x1
8:2	CALVALUE	Calibration Value. This read-only field displays the calibration value established during last calibration cycle. This value is not typically of any use to the user.	0x0
31:9	-	Reserved. Read value is undefined, only zero should be written.	0x0

28.7 Functional description

28.7.1 Conversion Sequences

A conversion sequence is a single pass through a series of ADC conversions performed on a selected set of ADC channels. Software can configure up to two independent conversion sequences, either of which can be triggered by software or by a transition on one of the hardware triggers. Each sequence can be triggered by a different hardware trigger. One of these conversion sequences is referred to as the A sequence and the other as the B sequence.

An optional single-step mode allows advancing through the channels of a sequence one at a time on each successive occurrence of a trigger.

The user can select whether a trigger on the B sequence can interrupt an already in-progress A sequence. The B sequence, however, can never be interrupted by an A trigger.

28.7.2 Hardware-triggered conversion

Software can select among hardware triggers will launch each conversion sequence and it can specify the active edge for the selected trigger independently for each conversion sequence.

For each conversion sequence, if a designated trigger event occurs, one single cycle through that conversion sequence will be launched unless:

- The BURST bit in the SEQn_CTRL register for this sequence is set to 1.
- The requested conversion sequence is already in progress.
- A set of conversions for the alternate conversion sequence is already in progress except in the case of a B trigger interrupting an A sequence if the A sequence is set to LOWPRIO.

If any of these conditions is true, the new trigger event will be ignored and will have no effect.

In addition, if the single-step bit for a sequence is set, each new trigger will cause a single conversion to be performed on the next channel in the sequence rather that launching a pass through the entire sequence.

If the A sequence is enabled to be interrupted (i.e. the LOWPRIO bit in the SEQA_CTRL register is set) and a B trigger occurs while an A sequence is in progress, then the following will occur:

- The ADC conversion which is currently in-progress will be aborted.
- The A sequence will be paused, and the B sequence will immediately commence.
- The interrupted A sequence will resume after the B sequence completes, beginning with the conversion that was aborted when the interruption occurred. The channel for that conversion will be re-sampled.

28.7.2.1 Avoiding spurious hardware triggers

Care should be taken to avoid generating a spurious trigger when writing to the SEQn_CTRL register to change the trigger selected for the sequence, switch the polarity of the selected trigger, or to enable the sequence for operation.

In general, the TRIGGER and TRIGPOL bits in the SEQn_ENA bit is should only be written when the sequence is disabled (while the SEQn_ENA bit = 0). The SEQn_ENA bit itself should only be set when the selected trigger input is in its INACTIVE state (as designated by the TRIGPOL bit). If this condition is not met, a trigger will be generated immediately upon enabling the sequence - even though no actual transition has occurred on the trigger input.

28.7.3 Software-triggered conversion

There are two ways that software can trigger a conversion sequence:

1. **Start Bit:** Setting the START bit in the corresponding SEQn_CTRL register. The response to this is identical to occurrence of a hardware trigger on that sequence. Specifically, one cycle of conversions through that conversion sequence will be immediately triggered except as indicated above.
2. **Burst Mode**: Set the BURST bit in the SEQn_CTRL register. As long as this bit is 1 the designated conversion sequence will be continuously and repetitively cycled through. Any new software or hardware trigger on this sequence will be ignored.

If a bursting A sequence is allowed to be interrupted (i.e. the LOWPRIO bit in its SEQA_CTRL register is set to 1) and a software or hardware trigger for the B sequence occurs, then the burst will be immediately interrupted and a B sequence will be initiated. The interrupted A sequence will resume continuous cycling, starting with the aborted conversion, after the alternate sequence has completed.

28.7.4 Interrupts

There are four interrupts that can be generated by the ADC:

- Conversion-Complete or Sequence-Complete interrupt for sequence A
- Conversion-Complete or Sequence-Complete interrupt for sequence B
- Threshold-Compare Out-of-Range Interrupt
- Data Overrun Interrupt

Any of these interrupt requests may be individually enabled or disabled in the INTEN register. Note that the threshold and overrun interrupts share a slot in the NVIC.

28.7.4.1 Conversion-Complete or Sequence-Complete interrupts

For each of the two sequences, an interrupt/DMA trigger can either be asserted at the end of each ADC conversion performed as part of that sequence or when the entire sequence of conversions is completed. The MODE bits in the SEQn_CTRL registers select between these alternative behaviors.

If the MODE bit for a sequence is 0 (conversion-complete mode), then the interrupt flag/DMA request for that sequence will reflect the state of the DATAVALID bit in the global data register (SEQn_GDAT) for that sequence. In this case, reading the SEQn_GDAT register will automatically clear the interrupt/DMA trigger.

If the MODE bit for the sequence is 1 (sequence-complete mode) then the interrupt flag/DMA request must be written-to by software to clear it (except when used as a DMA trigger, in which case it will be cleared in hardware by the DMA engine).

28.7.4.2 Threshold-Compare Out-of-Range Interrupt

Every conversion performed on any channel is automatically compared against a designated set of low and high threshold levels specified in the THRn_HIGH and THRn_LOW registers. The results of this comparison on any individual channel(s) can be enabled to cause a threshold-compare interrupt if that result was above or below the range specified by the two thresholds or, alternatively, if the result represented a crossing of the low threshold in either direction.

This flag must be cleared by a software write to clear the individual THCMP flags in the FLAGS register.

28.7.4.3 Data Overrun Interrupt

This interrupt/DMA trigger will be asserted if any of the OVERRUN bits in the individual channel data registers are set. In addition, the OVERRUN bits in the two sequence global data (SEQn_GDAT) registers will cause this interrupt/DMA trigger IF the MODE bit for that sequence is set to 0 (conversion-complete mode).

This flag will be cleared when the OVERRUN bit that caused it is cleared via reading the register containing it.

Note that the OVERRUN bits in the individual data registers are cleared when data related to that channel is read from either of the global data registers as well as when the individual data registers themselves are read.

28.7.5 Optional Operating Modes

There are three optional modes of ADC operation which may be selected in the CTRL register.

Four alternative ADC accuracy settings are available ranging from 12 bits down to 6 bits of resolution. Lowering the ADC resolution results in faster conversion times. A single ADC conversion (including one conversion in a burst or sequence) requires (resolution+3) ADC clocks when the minimum sampling period is selected. When reduced accuracy is selected, the unused LSBs of result data will automatically be forced to zero.

Two clocking modes are available, synchronous mode and asynchronous mode. The synchronous clocking mode uses the system clock in conjunction with an internal. programmable divider. The main advantage of this mode is determinism. The start of ADC sampling is always a fixed number of system clocks following any ADC trigger. The alternative asynchronous mode (on chips where this mode is supported) uses an independent clock source. In this mode the user has greater flexibility in selecting the ADC clock frequency to better achieve the maximum ADC conversion rate without restricting the clock rate for other peripherals. The penalty for using this mode may be longer latency and greater uncertainty in response to a hardware trigger.

28.7.6 Offset calibration and enabling the ADC

The A/D converter includes a built-in, self-calibration mode which can be used to minimize offset error. For applications where offset error is not a concern, calibration may be disabled by setting the BYPASSCAL bit in the CTRL register. If this bit is not set, a calibration cycle must be performed following chip power-up (including exit from deep-sleep, or deep power-down mode) prior to using the ADC.

Additional calibrations may be performed at any time by setting the CALIB bit in the CALIB register. Re-calibration is recommended if the temperature or voltage operating conditions have changed (including if the chip has been in a low-power mode for a considerable period of time). Re-calibration should also be performed if the ADC clock rate is changed.

A calibration cycle requires approximately 81 ADC clocks to complete. Normal ADC conversions cannot be launched, and the ADC Control register must not be written while calibration is in progress.

Remark: Enabling the ADC (following chip power-up, exit from any low-power mode, or when the ADC has been manually disabled) requires a specific start-up procedure. It is strongly recommended that the calibration function of the previously noted API be used to enable or re-enable the ADC.

Important: The ADC clock must be running at its full operating frequency prior to calling the API routine to enable the ADC. This means that the desired clocking mode and clock divide value (for sync mode) must be programmed into the CTRL register. If offset calibration is not desired, the BYPASSCAL bit in the CTRL register should also be set prior to calling the API routine to avoid wasting time on an unnecessary calibration cycle.

The ADC cannot be utilized until the startup routine has completed.

28.7.7 ADC vs. digital receiver

The analog ADC input must be selected via IOCON registers in order to get accurate voltage readings on the monitored pin. In the IOCON, the pull-up and pull-down resistors should be both disabled using the MODE bits. For a pin hosting an ADC input, it is not possible to have a have the digital function enabled and yet get valid ADC readings. Software must write a 0 to the DIGIMODE bit in the related IOCON register.

28.7.8 DMA control

The sequence A or sequence B conversion sequence complete interrupts may also be used to generate a DMA transfer trigger. To generate a DMA transfer the same conditions must be met as the conditions for generating an interrupt (see Section 28.7.4 and Section 28.6.13).

Remark: If DMA is used for a sequence, the corresponding sequence interrupt must be disabled in the INTEN register.

For DMA transfers, only burst requests are supported. The burst size can be set to one in the DMA channel control register. If the number of ADC channels is not equal to one of the other DMA-supported burst sizes (applicable DMA burst sizes are 1, 4, 8), set the burst size to one.

The DMA transfer size determines when a DMA interrupt is generated. The transfer size can be set to the number of ADC channels being converted. Non-contiguous channels can be transferred by the DMA using the scatter/gather linked lists.

28.7.9 ADC hardware trigger inputs

An analog-to-digital conversion can be initiated by a hardware trigger. The trigger can be selected independently for each of the two conversion sequences in the ADC SEQA_CTRL or SEQB_CTRL registers by programming the hardware trigger input # into the TRIGGER bits.

Related registers:

- Table 463 “ADC Conversion Sequence A Control register (SEQA_CTRL, offset 0x08) bit description”
- Table 464 “ADC Conversion Sequence B Control register (SEQB_CTRL, offset 0x0C) bit description”

Table 477. ADC0 hardware trigger inputs

Input #	Source	Description
0	PINT0	See Chapter 13 Group GPIO input interrupt (GINT0/1)
1	PINT1	See Chapter 13 Group GPIO input interrupt (GINT0/1)
2	SCT0_OUT7	See Chapter 15 SCTimer/PWM
4:3	-	Unused
5	ARM_TXEV	Transmit Event output from either CPU
7:6	-	Unused

28.7.10 Sample time

The analog input from the selected channel is sampled at the start of each new A/D conversion. The default (and shortest) duration of this sample period is 2.5 ADC clock cycles. Under some conditions, longer sample times may be required. A variety of factors including operating conditions, the ADC clock frequency, the selected ADC resolution, and the impedance of the analog source will influence the required sample period.

ADC channels 6 to 11 are somewhat slower than channels 0 to 5. Lower analog-source impedances may be required for these slow channels for a given sample period and set of operating conditions.

The following table provides guidelines for the required sample times. The “TSAMP” values displayed in the tables refer to the TSAMP field in the ADCTRL register. This value represents the number of additional clock cycles that must be added to the minimum 2.5-clock sample period in order to meet or exceed the required minimum sample time for the maximum ADC clock rate of 80 MHz under worst-case operating conditions. At slower clock frequencies fewer sample clocks will be needed to achieve the required sample times.

Table 478. Minimum required sample times

Selected ADC Resolution	Analog signal source impedance	Fast channels (ADC5:0)		Slow channels (ADC11:6)	
		Min. sample time	TSAMP field	Min. sample time	TSAMP field
12 bits	under 0.05k ohms	20 ns	0	43 ns	1
	0.05 to 0.1k ohms	23 ns	0	46 ns	1
	0.1K to 0.2k ohms	26 ns	0	50 ns	2
	0.2k to 0.5k ohms	31 ns	0	56 ns	2
	0.5k to 1.0k ohms	47 ns	1	74 ns	3
	1k to 5k ohms	75 ns	3	105 ns	6
10 bits	under 0.05k ohms	15 ns	0	35 ns	1
	0.05 to 0.1k ohms	18 ns	0	38 ns	1
	0.1K to 0.2k ohms	20 ns	0	40 ns	1
	0.2k to 0.5k ohms	24 ns	0	46 ns	1
	0.5k to 1.0k ohms	38 ns	1	61 ns	2
	1k to 5k ohms	62 ns	2	86 ns	4
8 bits	under 0.05k ohms	12 ns	0	27 ns	0
	0.05 to 0.1k ohms	13 ns	0	29 ns	0
	0.1K to 0.2k ohms	15 ns	0	32 ns	0
	0.2k to 0.5k ohms	19 ns	0	36 ns	1
	0.5k to 1.0k ohms	30 ns	0	48 ns	1
	1k to 5k ohms	48 ns	1	69 ns	3
6 bits	under 0.05k ohms	9 ns	0	20 ns	0
	0.05 to 0.1k ohms	10 ns	0	22 ns	0
	0.1K to 0.2k ohms	11 ns	0	23 ns	0
	0.2k to 0.5k ohms	13 ns	0	26 ns	0
	0.5k to 1.0k ohms	22 ns	0	36 ns	1
	1k to 5k ohms	36 ns	1	51 ns	2

28.8 Examples

28.8.1 Perform a single ADC conversion triggered by software

Remark: When ADC conversions are triggered by software only and hardware triggers are not used in the conversion sequence, follow these steps to avoid spurious conversions:

1. Before changing the trigger set-up, disable the conversion sequence by setting the SEQ_ENA bit to 0 in the SEQA_CTRL register.
2. Set the trigger source to an unused setting using the TRIGGER bits in the SEQA_CTRL register. The value 3, for example, is not used on this device.
3. Set the TRIGPOL bit to 1 in the in the SEQA_CTRL register.

Once the sequence is enabled again, the ADC converts a sample whenever the START bit is written to.

The ADC converts an analog input signal VIN on the ADC0_[11:0] pins. The VREFP and VREFN pins provide a positive and negative reference voltage input. The result of the conversion is (4095 x VIN)/(VREFP - VREFN). The result of an input voltage below VREFN is 0, and the result of an input voltage above VREFP is 4095 (0xFFF).

To perform a single ADC conversion for ADC0 channel 1 using the analog signal on pin ADC0_1, follow these steps:

1. Enable the analog function on pin ADC0_1 via IOCON See Table 193 and Table 194.
2. Configure the system clock to be 48 MHz and select a CLKDIV value of 0 for a sampling rate of 48 MHz using the ADC CTRL register. This example clock rate does not exceed the 80 MHz limit, so does not need to be further divided.
3. Select the synchronous mode in the CTRL register.
4. Select ADC channel 1 to perform the conversion by setting the CHANNELS bits to 0x2 in the SEQA_CTL register.
5. Set the TRIGPOL bit to 1 and the SEQA_ENA bit to 1 in the SEQA_CTRL register.
6. Set the START bit to 1 in the SEQA_CTRL register.
7. Read the RESULT bits in the DAT1 register for the conversion result.

28.8.2 Perform a sequence of conversions triggered by an external pin

The ADC can perform conversions on a sequence of selected channels. Each individual conversion of the sequence (single-step) or the entire sequence can be triggered by hardware. Hardware triggers are either a signal from an external pin or an internal signal. See Section 28.7.9.

To perform a single-step conversion on the first four channels of ADC0 triggered by rising edges on pin PIO1_0, follow these steps:

1. Enable the analog function on pin ADC0_0 to ADC0_3 via IOCON. See Table 193 and Table 194.
2. Configure PINT1 to respond to PIO1_0, see Chapter 12 for details.
3. Configure the system clock to be 80 MHz and select a CLKDIV value of 0 for a sampling rate of 80 MHz using the ADC CTRL register.
4. Select the synchronous mode in the CTRL register.
5. Select ADC channels 0 to 3 to perform the conversion by setting the CHANNELS bits to 0xF in the SEQA_CTRL register.
6. Select trigger PINT1 by writing 0x1 the TRIGGER bits in the SEQA_CTRL register.
7. To generate one interrupt at the end of the entire sequence, set the MODE bit to 1 in the SEQA_CTRL register.
8. Select single-step mode by setting the SINGLESTEP bit in the SEQA_CTRL register to 1.
9. Enable the Sequence A by setting the SEQA_ENA bit.

   A conversion on ADC0 channel 0 will be triggered whenever the pin PIO1_0 goes from LOW to HIGH. The conversion on the next channel (initially channel 1) is triggered on the next 0 to 1 transition of PINT1. The ADC0 interrupt is generated when the sequence has finished after four 0 to 1 transitions of PINT1.
10. Read the RESULT bits in the DAT0 to DAT3 registers for the conversion result.

UM11071

Chapter 29: LPC51U68 Temperature sensor

Rev. 1.1 — 17 May 2018 User manual

29.1 How to read this chapter

The temperature sensor is available on all LPC51U68 devices.

29.2 Features

- Linear temperature sensor.
- Sensor output internally connected to the ADC for temperature monitoring

29.3 Basic configuration

- Enable the power to the temperature sensor by setting the TS_PD bit in the PDRUNCFG register. See Section 6.5.48.
- To monitor the temperature continually, select the temperature sensor as source for channel 0 of ADC0. See Chapter 28. The digital temperature reading is available after an analog-to-digital conversion.

Remark: To convert the ADC conversion result into a temperature reading, see the device data sheet for information on temperature calibration of the sensor.

29.3.1 Perform a single ADC conversion with the temperature sensor as ADC input

To perform a single ADC conversion for ADC0 channel 0 using the temperature sensor output:

1. Enable the temperature sensor output as input to ADC channel 0.
2. Configure the system clock and the ADC for operation.
3. Select the synchronous mode in the ADC CTRL register.
4. Select ADC channel 0 to perform the conversion by setting the CHANNELS bits to 0x1 in the SEQA_CTL register.
5. Set the START bit to 1 in the SEQA_CTRL register.
6. Read the RESULT bits in the DAT0 register for the conversion result.

29.4 Pin description

The temperature sensor has no configurable pins.

29.5 Register description

The temperature sensor has no configurable registers.

UM11071

Chapter 30: LPC51U68 CRC engine

Rev. 1.1 — 17 May 2018 **User manual**

30.1 How to read this chapter

The CRC engine is available on all LPC51U68 parts.

30.2 Features

- Supports three common polynomials CRC-CCITT, CRC-16, and CRC-32.
  - CRC-CCITT: $x^{16} + x^{12} + x^5 + 1$
  - CRC-16: $x^{16} + x^{15} + x^2 + 1$
  - CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
- Bit order reverse and 1’s complement programmable setting for input data and CRC sum.
- Programmable seed number setting.
- Supports CPU PIO back-to-back transfer.
- Accept any size of data width per write: 8, 16 or 32-bit.
  - 8-bit write: 1-cycle operation
  - 16-bit write: 2-cycle operation (8-bit x 2-cycle)
  - 32-bit write: 4-cycle operation (8-bit x 4-cycle)

30.3 Basic configuration

Set the CRC bit in the AHBCLKCTRL0 register (Table 115) to enable the clock to the CRC engine.

30.4 Pin description

The CRC engine has no configurable pins.

30.5 General description

The Cyclic Redundancy Check (CRC) generator with programmable polynomial settings supports several CRC standards commonly used.

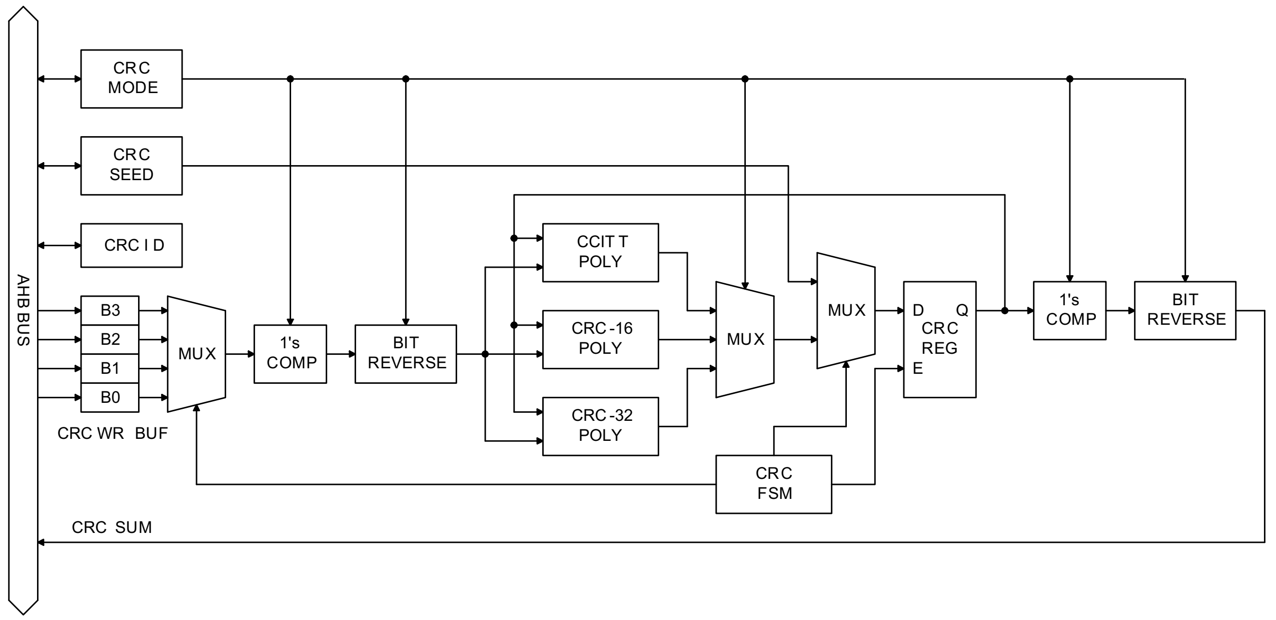


Fig 80. CRC block diagram

30.6 Register description

Table 479. Register overview: CRC engine (base address 0x4009 5000)

Name	Access	Offset	Description	Reset value	Section
MODE	R/W	0x000	CRC mode register	0x0	30.6.1
SEED	R/W	0x004	CRC seed register	0xFFFF	30.6.2
SUM	RO	0x008	CRC checksum register	0xFFFF	30.6.3
WR_DATA	WO	0x008	CRC data register	-	30.6.4

30.6.1 CRC mode register

Table 480. CRC mode register (MODE, offset 0x000) bit description

Bit	Symbol	Description	Reset value
1:0	CRC_POLY	CRC polynomial: 1X = CRC-32 polynomial 01 = CRC-16 polynomial 00 = CRC-CCITT polynomial	0x0
2	BIT_RVS_WR	Data bit order: 1 = Bit order reverse for CRC_WR_DATA (per byte) 0 = No bit order reverse for CRC_WR_DATA (per byte)	0x0
3	CMPL_WR	Data complement: 1 = 1's complement for CRC_WR_DATA 0 = No 1's complement for CRC_WR_DATA	0x0
4	BIT_RVS_SUM	CRC sum bit order: 1 = Bit order reverse for CRC_SUM 0 = No bit order reverse for CRC_SUM	0x0
5	CMPL_SUM	CRC sum complement: 1 = 1's complement for CRC_SUM 0 = No 1's complement for CRC_SUM	0x0
31:6	Reserved	Always 0 when read	0x0

30.6.2 CRC seed register

Table 481. CRC seed register (SEED, offset 0x004) bit description

Bit	Symbol	Description	Reset value
31:0	CRC_SEED	A write access to this register will load the CRC seed value to the SUM register with selected bit order and 1's complement pre-processes. **Remark:** A write access to this register will overrule the CRC calculation in progress.	0xFFFF

30.6.3 CRC checksum register

This register is a Read-only register containing the most recent checksum.

Table 482. CRC checksum register (SUM, offset 0x008) bit description

Bit	Symbol	Description	Reset value
31:0	CRC_SUM	The most recent CRC sum can be read through this register with selected bit order and 1's complement post-processes.	0x0000 FFFF

30.6.4 CRC data register

This register is a Write-only register containing the data block for which the CRC sum will be calculated.

Table 483. CRC data register (WR_DATA, offset 0x008) bit description

Bit	Symbol	Description	Reset value
31:0	CRC_WR_DATA	Data written to this register will be taken to perform CRC calculation with selected bit order and 1’s complement pre-process. Any write size 8, 16 or 32-bit are allowed and accept back-to-back transactions.	-

30.7 Functional description

30.7.1 Timing

The CRC engine uses some time to process data, which can depend on how it is accessed.

A write followed by another write:

For a 16-bit write to the CRC data register followed by another write to the same register, there is 1 wait state added.

For a 32-bit write to the CRC data register followed by another write to the same register, there are 3 wait states added.

A write followed by a read:

For an 8-bit write to the CRC data register followed by a read of the CRC checksum register, there is 1 wait state added.

For a 16-bit write to the CRC data register followed by a read of the CRC checksum register, there are 2 wait states added.

For a 32-bit write to the CRC data register followed by a read of the CRC checksum register, there are 4 wait states added.

30.7.2 Setup

The following sections describe the register settings for each supported CRC standard:

CRC-CCITT set-up

Polynomial = $x^{16} + x^{12} + x^5 + 1$

Seed Value = 0xFFFF

Bit order reverse for data input: NO

1's complement for data input: NO

Bit order reverse for CRC sum: NO

1's complement for CRC sum: NO

CRC_MODE = 0x0000 0000

CRC_SEED = 0x0000 FFFF

CRC-16 set-up

Polynomial = $x^{16} + x^{15} + x^{2} + 1$

Seed Value = 0x0000

Bit order reverse for data input: YES

1's complement for data input: NO

Bit order reverse for CRC sum: YES

1's complement for CRC sum: NO

CRC_MODE = 0x0000 0015

CRC_SEED = 0x0000 0000

CRC-32 set-up

Polynomial = $x^{32}+ x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$

Seed Value = 0xFFFF FFFF

Bit order reverse for data input: YES

1's complement for data input: NO

Bit order reverse for CRC sum: YES

1's complement for CRC sum: YES

CRC_MODE = 0x0000 0036

CRC_SEED = 0xFFFF FFFF

UM11071

Chapter 31: LPC51U68 Serial Wire Debug (SWD)

Rev. 1.1 — 17 May 2018 **User manual**

31.1 How to read this chapter

Debug functionality is available on all LPC51U68 devices.

31.2 Features

- Supports ARM Serial Wire Debug mode for the Cortex-M0+.
- Direct debug access to all memories, registers, and peripherals.
- No target resources are required for the debugging session.
- Four breakpoints.
- Two data watchpoints that can also be used as triggers.
- Supports JTAG boundary scan.

31.3 Basic configuration

The serial wire debug pins are enabled by default. The JTAG pins for boundary scan are selected by hardware after a reset.

31.4 Pin description

The tables below indicate the various pin functions related to debug. Some of these functions share pins with other functions which therefore may not be used at the same time.

Table 484. Serial Wire Debug pin description

Function	Type	Connect to	Description
SWCLK	In	PIO0_16	**Serial Wire Clock.** This pin is the clock for SWD debug logic when in the Serial Wire Debug mode (SWD). This pin is pulled up internally.
SWDIO	I/O	PIO0_17	**Serial wire debug data input/output.** The SWDIO pin is used by an external debug tool to communicate with and control the part. This pin is pulled up internally.

The JTAG boundary pin functions are selected by hardware at reset. See Section 31.6.3.

Table 485. JTAG boundary scan pin description

Function	Type	Connect to	Description
TCK	In	PIO0_14	**JTAG Test Clock.** This pin is the clock for JTAG boundary scan when the $\overline{\text{RESET}}$ pin is LOW.
TMS	In	PIO0_20	**JTAG Test Mode Select.** The TMS pin selects the next state in the TAP state machine. This pin includes an internal pull-up and is used for JTAG boundary scan when the $\overline{\text{RESET}}$ pin is LOW.

Table 485. JTAG boundary scan pin description

Function	Type	Connect to	Description
TDI	In	PIO0_19	**JTAG Test Data In.** This is the serial data input for the shift register. This pin includes an internal pull-up and is used for JTAG boundary scan when the $\overline{\text{RESET}}$ pin is LOW.
TDO	Out	PIO0_15	**JTAG Test Data Output.** This is the serial data output from the shift register. Data is shifted out of the device on the negative edge of the TCK signal. This pin is used for JTAG boundary scan when the $\overline{\text{RESET}}$ pin is LOW.
$\overline{\text{TRST}}$	In	PIO0_18	**JTAG Test Reset.** The $\overline{\text{TRST}}$ pin can be used to reset the test logic within the debug logic. This pin includes an internal pull-up and is used for JTAG boundary scan when the $\overline{\text{RESET}}$ pin is LOW.

31.5 General description

Serial wire debug functions are integrated into CPU, with up to four breakpoints and two watchpoints. Boundary scan is also available.

31.6 Functional description

31.6.1 Debug limitations

Important: Due to limitations of the CPU, the part cannot wake up in the usual manner from deep-sleep mode during debugging.

The debug mode changes the way in which reduced power modes work internal to the CPU. Therefore power measurements should not be made while debugging, power consumption is higher than during normal operation.

During a debugging session, the System Tick Timer is automatically stopped whenever the CPU is stopped. Other peripherals are not affected.

31.6.2 Debug connections for SWD

For debugging purposes, it is useful to provide access to the ISP entry pins (see Chapter 3 "LPC51U68 Boot process"). The ISP entry pins can be used to recover the part from configurations which would disable the SWD port such as improper PLL configuration, assigning another function to the SWD pins via IOCON, entry into deep power-down mode out of reset, etc. The ISP entry pins can be used for other functions such as GPIO but should not be held LOW on power-up or reset.

Internal and external SWD connections are shown in Figure 81 and Figure 82.

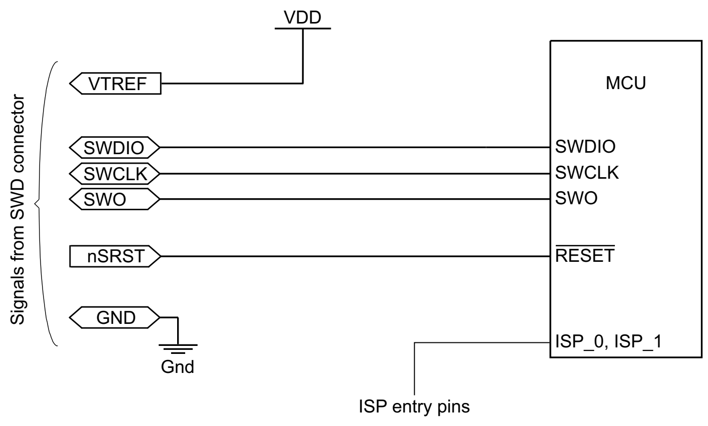


Fig 81. Connecting the SWD pins to a standard SWD connector

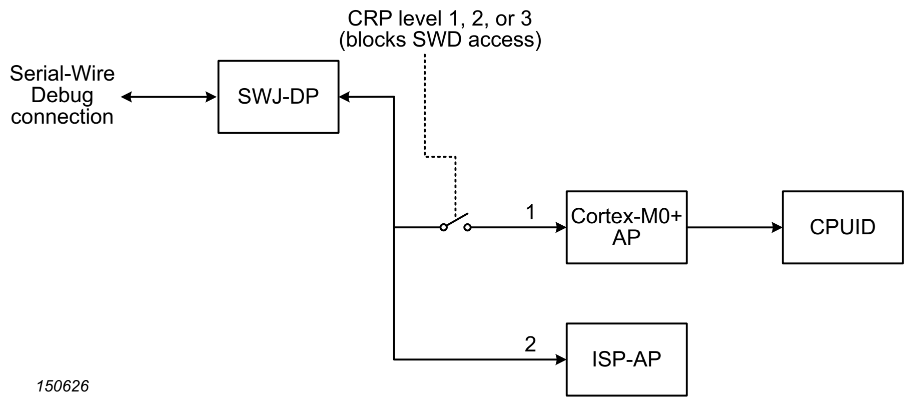


Fig 82. Serial Wire Debug internal connections

31.6.3 Boundary scan

The $\overline{\text{RESET}}$ pin selects between the test TAP controller for JTAG boundary scan ($\overline{\text{RESET}}$ = LOW) and the ARM SWD debug port TAP controller ($\overline{\text{RESET}}$ = HIGH). The ARM SWD debug port is disabled while the part is in reset. A LOW on the $\overline{\text{TRST}}$ pin resets the test TAP controller.

Remark: Boundary scan operations should not be started until 250 μs after POR. The test TAP must be reset after the boundary scan and left in either TLR or RTO state. Boundary scan is not affected by Code Read Protection.

Remark: POR, BOD reset, or a LOW on the TRST pin puts the test TAP controller in the Test-Logic Reset state. The first TCK clock while $\overline{\text{RESET}}$ = HIGH places the test TAP in Run-Test Idle mode.

31.6.4 In-System Programming Access Port (ISP-AP)

The ISP-AP is essentially a register-based communication port that may be accessed by the CPU and the device debug port.

This port is used to implement certain commands that can operate even when the device has been programmed to the highest Code Read Protection level (CRP level 3). The ISP-AP is active whenever the device is attached to a debugger.

31.6.4.1 Resynchronization request

Communication with the ISP-AP is initiated by the debugger. The debugger first sets the RESYNCH_REQ bit in the CSW register. The debugger must then reset the device by either writing a 1 to the CHIP_RESET_REQ bit in the CSW, or by driving the actual reset pin of the device if it is able to do so.

31.6.4.2 Acknowledgement of resynchronization request

After requesting a resynchronization and resetting the device, the debugger reads the CSW register. This stalls the debugger if the device has not yet completed the resynchronization process. The debugger can repeat this process until it is able to read the CSW and find a 0 there.

31.6.4.3 Return phase

Following the initial resynchronization, communication by the debugger to the device is in the form of 32-bit writes to the REQUEST register. The debugger can read the result in the RETURN register. The debugger polls the RETURN register in the same manner as it polled the CSW following a resynchronization request.

31.6.4.4 Error handling

If an overrun occurs from either side of the communication, the appropriate error flag is set in the CSW. Once such an error occurs, the debugger will need to start with a new resynchronization request in order to clear the error flag.

31.6.4.5 Register description

The registers in the ISP-AP are show below. These registers are readable by the CPU and are intended primarily to allow on-chip ROM routines to implement requests from an external debugger.

Table 486. Register overview: ISP-AP (base address 0x4009 C000)

Name	Access	Offset	Description	Reset value	Section
CSW	R/W	0x000	Command and status word.	0x0	31.6.4.5.1
REQUEST	R/W	0x004	Request from the debugger to the device.	0x0	31.6.4.5.2
RETURN	R/W	0x008	Return value from the device to the debugger.	0x0	31.6.4.5.3
ID	RO	0x0FC	Identification register.	0x002A 0000	31.6.4.5.4

31.6.4.5.1 Command and Status Word register

The CSW register contains command and status bits to facilitate communication between the debugger and the device.

Table 487. Command and Status Word register (CSW, offset 0x000) bit description

Bit	Symbol	Description	Reset value
0	RESYNCH_REQ	The debugger sets this bit to requests a re-synchronization.	0x0
1	REQ_PENDING	A request is pending for the debugger: a value is waiting to be read from the REQUEST register.	0x0
2	DBG_OR_ERR	When 1, a debug overrun has occurred: a REQUEST value has been overwritten by the debugger before it was read by the device.	0x0
3	AHB_OR_ERR	When 1, an AHB overrun has occurred: a RETURN value has been overwritten by the device before it was read by the debugger.	0x0
4	SOFT_RESET	This bit is write-only by the device and resets the ISP-AP.	0x0
5	CHIP_RESET_REQ	This write -only bit causes the device (but not the ISP-AP) to be reset.	0x0
31:6	-	Reserved	-

31.6.4.5.2 Request value register

The REQUEST register is used by a debugger to send action requests to the device.

Table 488. Request value register (REQUEST, offset 0x004) bit description

Bit	Symbol	Description	Reset value
31:0	REQUEST	Request value. Reads as 0 when no new request is present. Cleared by the device. Can be read back by the debugger in order to confirm communication.	0x0

31.6.4.5.3 Return value register

The RETURN register provides any response from the device to the debugger.

Table 489. Return value register (RETURN, offset 0x008) bit description

Bit	Symbol	Description	Reset value
31:0	RETURN	Return value. This is any response from the device to the debugger. If no new data is present, a debugger read will be stalled until new data is available.	0x0

31.6.4.5.4 Identification register

The ID register provides an identification of the ISP-AP interface.

Table 490. Identification register (ID, offset 0x0FC) bit description

Bit	Symbol	Description	Reset value
31:0	ID	Identification value.	0x002A 0000

31.6.4.6 ISP-AP commands

Commands for the ISP-AP are listed below. These would be written to the REQUEST register.

Table 491. ISP-AP commands

Name	Command code	Description
Enter ISP-AP	1	Cause the device to enter ISP-AP command mode. This must be done prior to sending other commands.
Bulk Erase	2	Erase the entire on-chip flash memory.
Query CRP Level	3	Queries the Code Read Protection (CRP) level currently in force on the device.
Exit ISP_AP	4	Cause the device to exit ISP-AP command mode. The Device returns to normal mode.

31.6.4.7 ISP-AP return codes

Return codes for ISP-AP commands are listed below. These would be read from the RETURN register.

Table 492. Register overview: ISP-AP return codes

Return code	Description
0x0000 0000	Command succeeded. Applies to commands other than “Query CRP level”.
0x0010 0001	Debug mode not entered. This is returned if other commands are sent prior to the “Enter ISP-AP” command.
0x0010 0002	Command not recognized. A command was received other than the ones defined above.
0x0010 0003	Command failed.

31.7 Debug configuration

31.7.1 Cortex-M0+

- Four breakpoints.
- Two data Watchpoints.

UM11071

Chapter 32: LPC51U68 USB ROM API

Rev. 1.1 — 17 May 2018 User manual

32.1 How to read this chapter

The USB ROM driver routines are available on all LPC51U68 devices. USB on-chip drivers are provided via the USB Stack in SDK and LPCOpen software packages.

32.2 Features

- ROM-base USB drivers.
- Communication Device Class (CDC) device class.
- Human Interface Device (HID) device class.
- Mass Storage Device (MSC) class.
- Device Firmware Upgrade (DFU) class.

32.3 General description

The boot ROM contains a USB driver to simplify the USB application development. The USB driver implements the Communication Device Class (CDC), the Human Interface Device (HID), Mass Storage Device (MSC) device class, and Device Firmware Upgrade (DFU) class. The USB on-chip drivers support composite device.

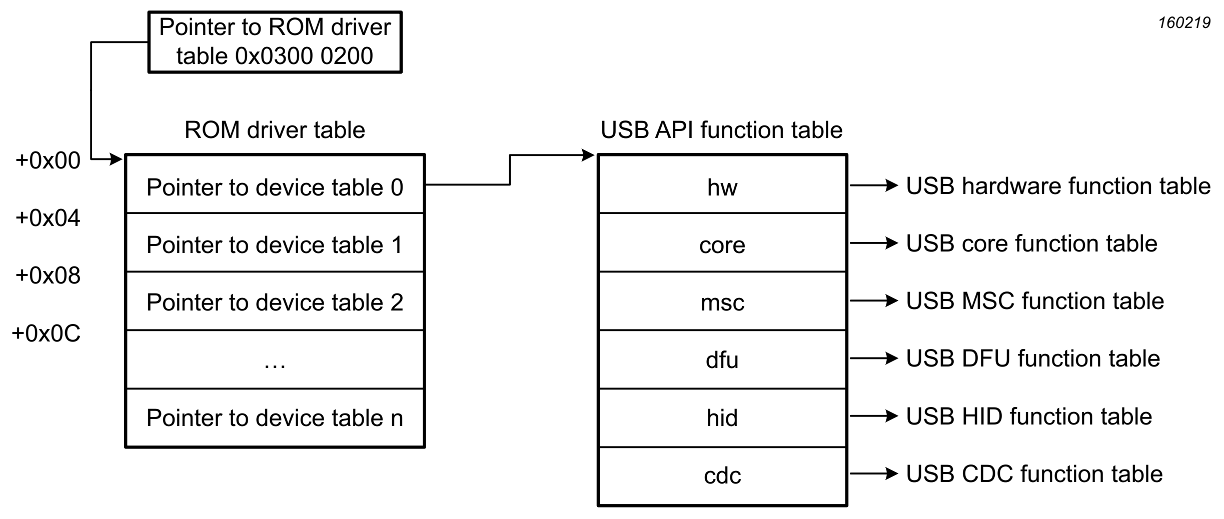


Fig 83. USB driver routines pointer structure

32.3.1 USB driver functions

The USB device driver ROM API consists of the following modules:

- Communication Device Class (CDC) function driver. This module contains an internal implementation of the USB CDC Class. User applications can use this class driver instead of implementing the CDC-ACM class manually via the low-level USBD_HW and USBD_Core APIs. This module is designed to simplify the user code by exposing only the required interface needed to interface with Devices using the USB CDC-ACM Class.

  - Communication Device Class function driver initialization parameter data structure (Table 520 “USBD_CDC_INIT_PARAM class structure”).
  - CDC class API functions structure. This module exposes functions which interact directly with USB device controller hardware (Table 519 “USBD_CDC_API class structure”).

- USB core layer
  - struct (Table 516 “_WB_T class structure”)
  - union (Table 493 “__WORD_BYTE class structure”)
  - struct (Table 494 “_BM_T class structure”)
  - struct (Table 507 “_REQUEST_TYPE class structure”)
  - struct (Table 514 “_USB_SETUP_PACKET class structure”)
  - struct (Table 510 “_USB_DEVICE_QUALIFIER_DESCRIPTOR class structure”)
  - struct USB device descriptor
  - struct (Table 510 “_USB_DEVICE_QUALIFIER_DESCRIPTOR class structure”)
  - struct USB configuration descriptor
  - struct (Table 512 “_USB_INTERFACE_DESCRIPTOR class structure”)
  - struct USB endpoint descriptor
  - struct (Table 515 “_USB_STRING_DESCRIPTOR class structure”)
  - struct (Table 508 “_USB_COMMON_DESCRIPTOR class structure”)
  - struct (Table 513 “_USB_OTHER_SPEED_CONFIGURATION class structure”)
  - USB descriptors data structure (Table 509 “_USB_CORE_DESCS_T class structure”)
  - USB device stack initialization parameter data structure (Table 518 “USBD_API_INIT_PARAM class structure”).
  - USB device stack core API functions structure (Table 521 “USBD_CORE_API class structure”).
- Device Firmware Upgrade (DFU) class function driver
  - DFU descriptors data structure (Table 523 “USBD_DFU_INIT_PARAM class structure”).
  - DFU class API functions structure. This module exposes functions which interact directly with the USB device controller hardware (Table 522 “USBD_DFU_API class structure”).
- HID class function driver
  - struct (Table 502 “_HID_DESCRIPTOR class structure”).
  - struct (Table 504 “_HID_REPORT_T class structure”).
  - USB descriptors data structure (Table 525 “USBD_HID_INIT_PARAM class structure”).

  - HID class API functions structure. This structure contains pointers to all the functions exposed by the HID function driver module (Table 526 “USBD_HW_API class structure”).
- USB device controller driver
  - Hardware API functions structure. This module exposes functions which interact directly with the USB device controller hardware (Table 526 “USBD_HW_API class structure”).
- Mass Storage Class (MSC) function driver
  - Mass Storage Class function driver initialization parameter data structure (Table 528).
  - MSC class API functions structure. This module exposes functions which interact directly with the USB device controller hardware (Table 527).

32.3.2 Calling the USB device driver

A fixed location in ROM contains a pointer to the ROM driver table. The ROM driver table contains a pointer to the USB driver table. Pointers to the various USB driver functions are stored in this table. USB driver functions can be called by using a C structure. Figure 83 illustrates the pointer mechanism used to access the on-chip USB driver.

```
typedef struct USBD_API
{
const USBD_HW_API_T* hw;
const USBD_CORE_API_T* core;
const USBD_MSC_API_T* msc;
const USBD_DFU_API_T* dfu;
const USBD_HID_API_T* hid;
const USBD_CDC_API_T* cdc;
const uint32_t* reserved6;
const uint32_t version;
} USBD_API_T;
```

32.4 USB API

32.4.1 __WORD_BYTE

Table 493. __WORD_BYTE class structure

Member	Description
W	`uint16_t __WORD_BYTE::W` data member to do 16 bit access
WB	`WB_TWB_T __WORD_BYTE::WB` data member to do 8 bit access

32.4.2 _BM_T

Table 494. _BM_T class structure

Member	Description
Recipient	`uint8_t _BM_T::Recipient` Recipient type.
Type	`uint8_t _BM_T::Type` Request type.
Dir	`uint8_t _BM_T::Dir` Direction type.

32.4.3 _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR

Table 495. _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR class structure

Member	Description
bFunctionLength	`uint8_t _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR::bFunctionLength`
bDescriptorType	`uint8_t _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR::bDescriptorType`
bDescriptorSubtype	`uint8_t _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR::bDescriptorSubtype`
bmCapabilities	`uint8_t _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR::bmCapabilities`

32.4.4 _CDC_CALL_MANAGEMENT_DESCRIPTOR

Table 496. _CDC_CALL_MANAGEMENT_DESCRIPTOR class structure

Member	Description
bFunctionLength	`uint8_t _CDC_CALL_MANAGEMENT_DESCRIPTOR::bFunctionLength`
bDescriptorType	`uint8_t _CDC_CALL_MANAGEMENT_DESCRIPTOR::bDescriptorType`
bDescriptorSubtype	`uint8_t _CDC_CALL_MANAGEMENT_DESCRIPTOR::bDescriptorSubtype`
bmCapabilities	`uint8_t _CDC_CALL_MANAGEMENT_DESCRIPTOR::bmCapabilities`
bDataInterface	`uint8_t _CDC_CALL_MANAGEMENT_DESCRIPTOR::bDataInterface`

32.4.5 _CDC_HEADER_DESCRIPTOR

Table 497. _CDC_HEADER_DESCRIPTOR class structure

Member	Description
bFunctionLength	`uint8_t _CDC_HEADER_DESCRIPTOR::bFunctionLength`
bDescriptorType	`uint8_t _CDC_HEADER_DESCRIPTOR::bDescriptorType`
bDescriptorSubtype	`uint8_t _CDC_HEADER_DESCRIPTOR::bDescriptorSubtype`
bcdCDC	`uint16_t _CDC_HEADER_DESCRIPTOR::bcdCDC`

32.4.6 _CDC_LINE_CODING

Table 498. _CDC_LINE_CODING class structure

Member	Description
dwDTERate	`uint32_t _CDC_LINE_CODING::dwDTERate`
bCharFormat	`uint8_t _CDC_LINE_CODING::bCharFormat`
bParityType	`uint8_t _CDC_LINE_CODING::bParityType`
bDataBits	`uint8_t _CDC_LINE_CODING::bDataBits`

32.4.7 _CDC_UNION_1SLAVE_DESCRIPTOR

Table 499. _CDC_UNION_1SLAVE_DESCRIPTOR class structure

Member	Description
sUnion	`CDC_UNION_DESCRIPTORCDC_UNION_DESCRIPTOR _CDC_UNION_1SLAVE_DESCRIPTOR::sUnion`
bSlaveInterfaces	`uint8_t _CDC_UNION_1SLAVE_DESCRIPTOR::bSlaveInterfaces[1][1]`

32.4.8 _CDC_UNION_DESCRIPTOR

Table 500. _CDC_UNION_DESCRIPTOR class structure

Member	Description
bFunctionLength	`uint8_t _CDC_UNION_DESCRIPTOR::bFunctionLength`
bDescriptorType	`uint8_t _CDC_UNION_DESCRIPTOR::bDescriptorType`
bDescriptorSubtype	`uint8_t _CDC_UNION_DESCRIPTOR::bDescriptorSubtype`
bMasterInterface	`uint8_t _CDC_UNION_DESCRIPTOR::bMasterInterface`

32.4.9 _DFU_STATUS

Table 501. _DFU_STATUS class structure

Member	Description
bStatus	`uint8_t _DFU_STATUS::bStatus`
bwPollTimeout	`uint8_t _DFU_STATUS::bwPollTimeout[3][3]`
bState	`uint8_t _DFU_STATUS::bState`
iString	`uint8_t _DFU_STATUS::iString`

32.4.10 _HID_DESCRIPTOR

HID class-specific HID Descriptor.

Table 502. _HID_DESCRIPTOR class structure

Member	Description
bLength	`uint8_t _HID_DESCRIPTOR::bLength` Size of the descriptor, in bytes.
bDescriptorType	`uint8_t _HID_DESCRIPTOR::bDescriptorType` Type of HID descriptor.
bcdHID	`uint16_t _HID_DESCRIPTOR::bcdHID` BCD encoded version that the HID descriptor and device complies to.
bCountryCode	`uint8_t _HID_DESCRIPTOR::bCountryCode` Country code of the localized device, or zero if universal.
bNumDescriptors	`uint8_t _HID_DESCRIPTOR::bNumDescriptors` Total number of HID report descriptors for the interface.
DescriptorList	`PRE_PACK struct POST_PACK _HID_DESCRIPTOR::_HID_DESCRIPTOR_LISTPRE_PACK struct POST_PACK _HID_DESCRIPTOR::_HID_DESCRIPTOR_LIST _HID_DESCRIPTOR::DescriptorList[1][1]` Array of one or more descriptors

32.4.11 _HID_DESCRIPTOR::_HID_DESCRIPTOR_LIST

Table 503. _HID_DESCRIPTOR::_HID_DESCRIPTOR_LIST class structure

Member	Description
bDescriptorType	`uint8_t _HID_DESCRIPTOR::_HID_DESCRIPTOR_LIST::bDescriptorType` Type of HID report.
wDescriptorLength	`uint16_t _HID_DESCRIPTOR::_HID_DESCRIPTOR_LIST::wDescriptorLength` Length of the associated HID report descriptor, in bytes.

32.4.12 _HID_REPORT_T

HID report descriptor data structure.

Table 504. _HID_REPORT_T class structure

Member	Description
len	`uint16_t _HID_REPORT_T::len` Size of the report descriptor in bytes.
idle_time	`uint8_t _HID_REPORT_T::idle_time` This value is used by stack to respond to Set_Idle & GET_Idle requests for the specified report ID. The value of this field specified the rate at which duplicate reports are generated for the specified Report ID. For example, a device with two input reports could specify an idle rate of 20 milliseconds for report ID 1 and 500 milliseconds for report ID 2.
__pad	`uint8_t _HID_REPORT_T::__pad` Padding space.
desc	`uint8_t * _HID_REPORT_T::desc` Report descriptor.

32.4.13 _MSC_CBW

Table 505. _MSC_CBW class structure

Member	Description
dSignature	`uint32_t _MSC_CBW::dSignature`
dTag	`uint32_t _MSC_CBW::dTag`
dDataLength	`uint32_t _MSC_CBW::dDataLength`
bmFlags	`uint8_t _MSC_CBW::bmFlags`
bLUN	`uint8_t _MSC_CBW::bLUN`
bCBLength	`uint8_t _MSC_CBW::bCBLength`
CB	`uint8_t _MSC_CBW::CB[16][16]`

32.4.14 _MSC_CSW

Table 506. _MSC_CSW class structure

Member	Description
dSignature	`uint32_t _MSC_CSW::dSignature`
dTag	`uint32_t _MSC_CSW::dTag`
dDataResidue	`uint32_t _MSC_CSW::dDataResidue`
bStatus	`uint8_t _MSC_CSW::bStatus`

32.4.15 _REQUEST_TYPE

Table 507. _REQUEST_TYPE class structure

Member	Description
B	`uint8_t _REQUEST_TYPE::B` byte wide access member
BM	`BM_TBM_T _REQUEST_TYPE::BM` bitfield structure access member

32.4.16 _USB_COMMON_DESCRIPTOR

Table 508. _USB_COMMON_DESCRIPTOR class structure

Member	Description
bLength	`uint8_t _USB_COMMON_DESCRIPTOR::bLength` Size of this descriptor in bytes
bDescriptorType	`uint8_t _USB_COMMON_DESCRIPTOR::bDescriptorType` Descriptor Type

32.4.17 _USB_CORE_DESCS_T

USB descriptors data structure.

Table 509. _USB_CORE_DESCS_T class structure

Member	Description
device_desc	`uint8_t * _USB_CORE_DESCS_T::device_desc` Pointer to USB device descriptor
string_desc	`uint8_t * _USB_CORE_DESCS_T::string_desc` Pointer to array of USB string descriptors
full_speed_desc	`uint8_t * _USB_CORE_DESCS_T::full_speed_desc` Pointer to USB device configuration descriptor when device is operating in full speed mode.
high_speed_desc	`uint8_t * _USB_CORE_DESCS_T::high_speed_desc` Pointer to USB device configuration descriptor when device is operating in high speed mode. For full-speed only implementation this pointer should be same as full_speed_desc.
device_qualifier	`uint8_t * _USB_CORE_DESCS_T::device_qualifier` Pointer to USB device qualifier descriptor. For full-speed only implementation this pointer should be set to null (0).

32.4.18 _USB_DEVICE_QUALIFIER_DESCRIPTOR

Table 510. _USB_DEVICE_QUALIFIER_DESCRIPTOR class structure

Member	Description
bLength	`uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bLength` Size of descriptor
bDescriptorType	`uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bDescriptorType` Device Qualifier Type
bcdUSB	`uint16_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bcdUSB` USB specification version number (e.g., 0200H for V2.00)
bDeviceClass	`uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bDeviceClass` Class Code
bDeviceSubClass	`uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bDeviceSubClass` SubClass Code
bDeviceProtocol	`uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bDeviceProtocol` Protocol Code
bMaxPacketSize0	`uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bMaxPacketSize0` Maximum packet size for other speed
bNumConfigurations	`uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bNumConfigurations` Number of Other-speed Configurations
bReserved	`uint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bReserved` Reserved for future use, must be zero

32.4.19 _USB_DFU_FUNC_DESCRIPTOR

Table 511. _USB_DFU_FUNC_DESCRIPTOR class structure

Member	Description
bLength	`uint8_t _USB_DFU_FUNC_DESCRIPTOR::bLength`
bDescriptorType	`uint8_t _USB_DFU_FUNC_DESCRIPTOR::bDescriptorType`
bmAttributes	`uint8_t _USB_DFU_FUNC_DESCRIPTOR::bmAttributes`
wDetachTimeOut	`uint16_t _USB_DFU_FUNC_DESCRIPTOR::wDetachTimeOut`
wTransferSize	`uint16_t _USB_DFU_FUNC_DESCRIPTOR::wTransferSize`
bcdDFUVersion	`uint16_t _USB_DFU_FUNC_DESCRIPTOR::bcdDFUVersion`

32.4.20 _USB_INTERFACE_DESCRIPTOR

Table 512. _USB_INTERFACE_DESCRIPTOR class structure

Member	Description
bLength	`uint8_t _USB_INTERFACE_DESCRIPTOR::bLength` Size of this descriptor in bytes
bDescriptorType	`uint8_t _USB_INTERFACE_DESCRIPTOR::bDescriptorType` INTERFACE Descriptor Type
bInterfaceNumber	`uint8_t _USB_INTERFACE_DESCRIPTOR::bInterfaceNumber` Number of this interface. Zero-based value identifying the index in the array of concurrent interfaces supported by this configuration.
bAlternateSetting	`uint8_t _USB_INTERFACE_DESCRIPTOR::bAlternateSetting` Value used to select this alternate setting for the interface identified in the prior field
bNumEndpoints	`uint8_t _USB_INTERFACE_DESCRIPTOR::bNumEndpoints` Number of endpoints used by this interface (excluding endpoint zero). If this value is zero, this interface only uses the Default Control Pipe.
bInterfaceClass	`uint8_t _USB_INTERFACE_DESCRIPTOR::bInterfaceClass` Class code (assigned by the USB-IF).
bInterfaceSubClass	`uint8_t _USB_INTERFACE_DESCRIPTOR::bInterfaceSubClass` Subclass code (assigned by the USB-IF).
bInterfaceProtocol	`uint8_t _USB_INTERFACE_DESCRIPTOR::bInterfaceProtocol` Protocol code (assigned by the USB).
iInterface	`uint8_t _USB_INTERFACE_DESCRIPTOR::iInterface` Index of string descriptor describing this interface

32.4.21 _USB_OTHER_SPEED_CONFIGURATION

Table 513. _USB_OTHER_SPEED_CONFIGURATION class structure

Member	Description
bLength	`uint8_t _USB_OTHER_SPEED_CONFIGURATION::bLength` Size of descriptor
bDescriptorType	`uint8_t _USB_OTHER_SPEED_CONFIGURATION::bDescriptorType` Other_speed_Configuration Type
wTotalLength	`uint16_t _USB_OTHER_SPEED_CONFIGURATION::wTotalLength` Total length of data returned
bNumInterfaces	`uint8_t _USB_OTHER_SPEED_CONFIGURATION::bNumInterfaces` Number of interfaces supported by this speed configuration
bConfigurationValue	`uint8_t _USB_OTHER_SPEED_CONFIGURATION::bConfigurationValue` Value to use to select configuration
IConfiguration	`uint8_t _USB_OTHER_SPEED_CONFIGURATION::IConfiguration` Index of string descriptor
bmAttributes	`uint8_t _USB_OTHER_SPEED_CONFIGURATION::bmAttributes` Same as Configuration descriptor
bMaxPower	`uint8_t _USB_OTHER_SPEED_CONFIGURATION::bMaxPower` Same as Configuration descriptor

32.4.22 _USB_SETUP_PACKET

Table 514. _USB_SETUP_PACKET class structure

Member	Description
bmRequestType	`REQUEST_TYPE _USB_SETUP_PACKET::bmRequestType` This bit-mapped field identifies the characteristics of the specific request. _BM_T.
bRequest	`uint8_t _USB_SETUP_PACKET::bRequest` This field specifies the particular request. The Type bits in the bmRequestType field modify the meaning of this field. USBD_REQUEST.
wValue	`WORD_BYTE _USB_SETUP_PACKET::wValue` Used to pass a parameter to the device, specific to the request.
wIndex	`WORD_BYTE _USB_SETUP_PACKET::wIndex` Used to pass a parameter to the device, specific to the request. The wIndex field is often used in requests to specify an endpoint or an interface.
wLength	`uint16_t _USB_SETUP_PACKET::wLength` This field specifies the length of the data transferred during the second phase of the control transfer.

32.4.23 _USB_STRING_DESCRIPTOR

Table 515. _USB_STRING_DESCRIPTOR class structure

Member	Description
bLength	`uint8_t _USB_STRING_DESCRIPTOR::bLength` Size of this descriptor in bytes
bDescriptorType	`uint8_t _USB_STRING_DESCRIPTOR::bDescriptorType` STRING Descriptor Type
bString	`uint16_t _USB_STRING_DESCRIPTOR::bString` UNICODE encoded string

32.4.24 _WB_T

Table 516. _WB_T class structure

Member	Description
L	`uint8_t _WB_T::L` lower byte
H	`uint8_t _WB_T::H` upper byte

32.4.25 USBD_API

Main USBD API functions structure.This structure contains pointer to various USB Device stack's sub-module function tables. This structure is used as main entry point to access various methods (grouped in sub-modules) exposed by ROM based USB device stack.

Table 517. USBD_API class structure

Member	Description
hw	`const USBD_HW_API_T* USBD_API::hw` Pointer to function table which exposes functions which interact directly with USB device stack's core layer.
core	`const USBD_CORE_API_T* USBD_API::core` Pointer to function table which exposes functions which interact directly with USB device controller hardware.
msc	`const USBD_MSC_API_T* USBD_API::msc` Pointer to function table which exposes functions provided by MSC function driver module.
dfu	`const USBD_DFU_API_T* USBD_API::dfu` Pointer to function table which exposes functions provided by DFU function driver module.
hid	`const USBD_HID_API_T* USBD_API::hid` Pointer to function table which exposes functions provided by HID function driver module.

Table 517. USBD_API class structure

Member	Description
cdc	`const USBD_CDC_API_T* USBD_API::cdc` Pointer to function table which exposes functions provided by CDC-ACM function driver module.
reserved6	`const uint32_t* USBD_API::reserved6` Reserved for future function driver module.
version	`const uint32_t USBD_API::version` Version identifier of USB ROM stack. The version is defined as 0x0CHDMhCC where each nibble represents version number of the corresponding component. CC - 7:0 - 8bit core version number h - 11:8 - 4bit hardware interface version number M - 15:12 - 4bit MSC class module version number D - 19:16 - 4bit DFU class module version number H - 23:20 - 4bit HID class module version number C - 27:24 - 4bit CDC class module version number H - 31:28 - 4bit reserved

32.4.26 USBD_API_INIT_PARAM

USB device stack initialization parameter data structure.

Table 518. USBD_API_INIT_PARAM class structure

Member	Description
usb_reg_base	`uint32_t USBD_API_INIT_PARAM::usb_reg_base` USB device controller's base register address.
mem_base	`uint32_t USBD_API_INIT_PARAM::mem_base` Base memory location from where the stack can allocate data and buffers. **Remark:** The memory address set in this field should be accessible by USB DMA controller. Also this value should be aligned on 2048 byte boundary.
mem_size	`uint32_t USBD_API_INIT_PARAM::mem_size` The size of memory buffer which stack can use. **Remark:** The mem_size should be greater than the size returned by USBD_HW_API::GetMemSize() routine.
max_num_ep	`uint8_t USBD_API_INIT_PARAM::max_num_ep` max number of endpoints supported by the USB device controller instance (specified by
pad0	`uint8_t USBD_API_INIT_PARAM::pad0[3][3]`
USB_Reset_Event	`USB_CB_T USBD_API_INIT_PARAM::USB_Reset_Event` Event for USB interface reset. This event fires when the USB host requests that the device reset its interface. This event fires after the control endpoint has been automatically configured by the library. **Remark:** This event is called from USB_ISR context and hence is time-critical. Having delays in this callback will prevent the device from enumerating correctly or operate properly.
USB_Suspend_Event	`USB_CB_T USBD_API_INIT_PARAM::USB_Suspend_Event` Event for USB suspend. This event fires when the USB host suspends the device by halting its transmission of Start Of Frame pulses to the device. This is generally hooked in order to move the device over to a low power state until the host wakes up the device. **Remark:** This event is called from USB_ISR context and hence is time-critical. Having delays in this callback will cause other system issues.

Table 518. USBD_API_INIT_PARAM class structure

Member	Description
USB_Resume_Event	`USB_CB_T USBD_API_INIT_PARAM::USB_Resume_Event` Event for USB wake up or resume. This event fires when a the USB device interface is suspended and the host wakes up the device by supplying Start Of Frame pulses. This is generally hooked to pull the user application out of a low power state and back into normal operating mode. **Remark:** This event is called from USB_ISR context and hence is time-critical. Having delays in this callback will cause other system issues.
reserved_sbz	`USB_CB_T USBD_API_INIT_PARAM::reserved_sbz` Reserved parameter should be set to zero.
USB_SOF_Event	`USB_CB_T USBD_API_INIT_PARAM::USB_SOF_Event` Event for USB Start Of Frame detection, when enabled. This event fires at the start of each USB frame, once per millisecond in full-speed mode or once per 125 microseconds in high-speed mode, and is synchronized to the USB bus. This event is time-critical; it is run once per millisecond (full-speed mode) and thus long handlers will significantly degrade device performance. This event should only be enabled when needed to reduce device wake-ups. This event is not normally active - it must be manually enabled and disabled via the USB interrupt register. **Remark:** This event is not normally active - it must be manually enabled and disabled via the USB interrupt register.
USB_WakeUpCfg	`USB_PARAM_CB_T USBD_API_INIT_PARAM::USB_WakeUpCfg` Event for remote wake-up configuration, when enabled. This event fires when the USB host request the device to configure itself for remote wake-up capability. The USB host sends this request to device which report remote wake-up capable in their device descriptors, before going to low-power state. The application layer should implement this callback if they have any special on board circuit to trigger remote wake up event. Also application can use this callback to differentiate the following SUSPEND event is caused by cable plug-out or host SUSPEND request. The device can wake-up host only after receiving this callback and remote wake-up feature is enabled by host. To signal remote wake-up the device has to generate resume signaling on bus by calling usapi.hw->WakeUp() routine. Parameters: 1. hUsb = Handle to the USB device stack. 2. param1 = When 0 - Clear the wake-up configuration, 1 - Enable the wake-up configuration. Returns: the call back should return ErrorCode_t type to indicate success or error condition.
USB_Power_Event	`USB_PARAM_CB_T USBD_API_INIT_PARAM::USB_Power_Event` Reserved parameter should be set to zero.
USB_Error_Event	`USB_PARAM_CB_T USBD_API_INIT_PARAM:USB_Error_Event` Event for error condition. This event fires when USB device controller detect an error condition in the system. Parameters: 1. hUsb = Handle to the USB device stack. 2. param1 = USB device interrupt status register. Returns: the call back should return ErrorCode_t type to indicate success or error condition.

Table 518. USBD_API_INIT_PARAM class structure

Member	Description
USB_Configure_Event	`USB_CB_T USBD_API_INIT_PARAM::USB_Configure_Event` Event for USB configuration number changed. This event fires when a the USB host changes the selected configuration number. On receiving configuration change request from host, the stack enables/configures the endpoints needed by the new configuration before calling this callback function. **Remark:** This event is called from USB_ISR context and hence is time-critical. Having delays in this callback will prevent the device from enumerating correctly or operate properly.
USB_Interface_Event	`USB_CB_T USBD_API_INIT_PARAM::USB_Interface_Event` Event for USB interface setting changed. This event fires when a the USB host changes the interface setting to one of alternate interface settings. On receiving interface change request from host, the stack enables/configures the endpoints needed by the new alternate interface setting before calling this callback function. **Remark:** This event is called from USB_ISR context and hence is time-critical. Having delays in this callback will prevent the device from enumerating correctly or operate properly.
USB_Feature_Event	`USB_CB_T USBD_API_INIT_PARAM::USB_Feature_Event` Event for USB feature changed. This event fires when a the USB host send set/clear feature request. The stack handles this request for USB_FEATURE_REMOTE_WAKEUP, USB_FEATURE_TEST_MODE and USB_FEATURE_ENDPOINT_STALL features only. On receiving feature request from host, the stack handle the request appropriately and then calls this callback function. **Remark:** This event is called from USB_ISR context and hence is time-critical. Having delays in this callback will prevent the device from enumerating correctly or operate properly.
virt_to_phys	`uint32_t(* USBD_API_INIT_PARAM::virt_to_phys)(void *vaddr)` Reserved parameter for future use. should be set to zero.
cache_flush	`void(* USBD_API_INIT_PARAM::cache_flush)(uint32_t *start_adr, uint32_t *end_adr)` Reserved parameter for future use. should be set to zero.

32.4.27 USBD_CDC_API

CDC class API functions structure.This module exposes functions which interact directly with USB device controller hardware.

Table 519. USBD_CDC_API class structure

Member	Description
GetMemSize	`uint32_t(*uint32_t USBD_CDC_API::GetMemSize)(USBD_CDC_INIT_PARAM_T *param)` Function to determine the memory required by the CDC function driver module. This function is called by application layer before calling pUsbApi->CDC->Init(), to allocate memory used by CDC function driver module. The application should allocate the memory which is accessible by USB controller/DMA controller. **Remark:** Some memory areas are not accessible by all bus masters. Parameters: 1. param = Structure containing CDC function driver module initialization parameters. Returns: the required memory size in bytes.

Table 519. USBD_CDC_API class structure

Member	Description
init	`ErrorCode_t(*ErrorCode_t USBD_CDC_API::init)(USBD_HANDLE_T hUsb, USBD_CDC_INIT_PARAM_T *param, USBD_HANDLE_T *phCDC)` Function to initialize CDC function driver module. This function is called by application layer to initialize CDC function driver module. hUsbHandle to the USB device stack. paramStructure containing CDC function driver module initialization parameters. Parameters: 1. hUsb = Handle to the USB device stack. 2. param = Structure containing CDC function driver module initialization parameters. Returns: ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success 2. ERR_USBD_BAD_MEM_BUF = Memory buffer passed is not 4-byte aligned or smaller than required. 3. ERR_API_INVALID_PARAM2 = Either CDC_Write() or CDC_Read() or CDC_Verify() callbacks are not defined. 4. ERR_USBD_BAD_INTF_DESC = Wrong interface descriptor is passed. 5. ERR_USBD_BAD_EP_DESC = Wrong endpoint descriptor is passed.
SendNotification	`ErrorCode_t(*ErrorCode_t USBD_CDC_API::SendNotification)(USBD_HANDLE_T hCdc, uint8_t bNotification, uint16_t data)` Function to send CDC class notifications to host. This function is called by application layer to send CDC class notifications to host. See usbcdc11.pdf, section 6.3, Table 67 for various notification types the CDC device can send. **Remark:** The current version of the driver only supports following notifications allowed by ACM subclass: CDC_NOTIFICATION_NETWORK_CONNECTION, CDC_RESPONSE_AVAILABLE, CDC_NOTIFICATION_SERIAL_STATE. For all other notifications application should construct the notification buffer appropriately and call hw->USB_WriteEP() for interrupt endpoint associated with the interface. Parameters: 1. hCdc = Handle to CDC function driver. 2. bNotification = Notification type allowed by ACM subclass. Should be CDC_NOTIFICATION_NETWORK_CONNECTION, CDC_RESPONSE_AVAILABLE or CDC_NOTIFICATION_SERIAL_STATE. For all other types ERR_API_INVALID_PARAM2 is returned. See usbcdc11.pdf, section 3.6.2.1, table 5. 3. data = Data associated with notification. For CDC_NOTIFICATION_NETWORK_CONNECTION a non-zero data value is interpreted as connected state. For CDC_RESPONSE_AVAILABLE this parameter is ignored. For CDC_NOTIFICATION_SERIAL_STATE the data should use bitmap values defined in usbcdc11.pdf, section 6.3.5, Table 69. Returns: ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success 2. ERR_API_INVALID_PARAM2 = If unsupported notification type is passed.

32.4.28 USBD_CDC_INIT_PARAM

Communication Device Class function driver initialization parameter data structure.

Table 520. USBD_CDC_INIT_PARAM class structure

Member	Description
mem_base	`uint32_t USBD_CDC_INIT_PARAM::mem_base` Base memory location from where the stack can allocate data and buffers. **Remark:** The memory address set in this field should be accessible by USB DMA controller. Also this value should be aligned on 4 byte boundary.
mem_size	`uint32_t USBD_CDC_INIT_PARAM::mem_size` The size of memory buffer which stack can use. **Remark:** The mem_size should be greater than the size returned by USBD_CDC_API::GetMemSize() routine.
cif_intf_desc	`uint8_t * USBD_CDC_INIT_PARAM::cif_intf_desc` Pointer to the control interface descriptor within the descriptor array
dif_intf_desc	`uint8_t * USBD_CDC_INIT_PARAM::dif_intf_desc` Pointer to the data interface descriptor within the descriptor array
CIC_GetRequest	`ErrorCode_t(* USBD_CDC_INIT_PARAM::CIC_GetRequest)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuffer, uint16_t *length)` Communication Interface Class specific get request call-back function. This function is provided by the application software. This function gets called when host sends CIC management element get requests. **Remark:** Applications implementing Abstract Control Model subclass can set this param to NULL. As the default driver parses ACM requests and calls the individual ACM call-back routines defined in this structure. For all other subclasses this routine should be provided by the application. The setup packet data (pSetup) is passed to the call-back so that application can extract the CIC request type and other associated data. By default the stack will assign pBuffer pointer to EP0Buff allocated at init. The application code can directly write data into this buffer as long as data is less than 64 byte. If more data has to be sent then application code should update pBuffer pointer and length accordingly. Parameters: 1. hCdc = Handle to CDC function driver. 2. pSetup = Pointer to setup packet received from host. 3. pBuffer = Pointer to a pointer of data buffer containing request data. Pointer-to-pointer is used to implement zero-copy buffers. See USBD_ZeroCopy for more details on zero-copy concept. 4. length = Amount of data to be sent back to host. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

Table 520. USBD_CDC_INIT_PARAM class structure

Member	Description
CIC_SetRequest	`ErrorCode_t(* USBD_CDC_INIT_PARAM::CIC_SetRequest)(USBD_HANDLE_T hCdc, USB_SETUP_PACKET *pSetup, uint8_t **pBuffer, uint16_t length)` Communication Interface Class specific set request call-back function. This function is provided by the application software. This function gets called when host sends a CIC management element requests. **Remark:** Applications implementing Abstract Control Model subclass can set this param to NULL. As the default driver parses ACM requests and calls the individual ACM call-back routines defined in this structure. For all other subclasses this routine should be provided by the application. The setup packet data (pSetup) is passed to the call-back so that application can extract the CIC request type and other associated data. If a set request has data associated, then this call-back is called twice. (1) First when setup request is received, at this time application code could update pBuffer pointer to point to the intended destination. The length param is set to 0 so that application code knows this is first time. By default the stack will assign pBuffer pointer to EP0Buff allocated at init. Note, if data length is greater than 64 bytes and application code doesn't update pBuffer pointer the stack will send STALL condition to host. (2) Second when the data is received from the host. This time the length param is set with number of data bytes received. Parameters: 1. hCdc = Handle to CDC function driver. 2. pSetup = Pointer to setup packet received from host. 3. pBuffer = Pointer to a pointer of data buffer containing request data. Pointer-to-pointer is used to implement zero-copy buffers. See USBD_ZeroCopy for more details on zero-copy concept. 4. length = Amount of data copied to destination buffer. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.
CDC_BulkIN_Hdlr	`ErrorCode_t(* USBD_CDC_INIT_PARAM::CDC_BulkIN_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event)` Communication Device Class specific BULK IN endpoint handler. The application software should provide the BULK IN endpoint handler. Applications should transfer data depending on the communication protocol type set in descriptors. Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

Table 520. USBD_CDC_INIT_PARAM class structure

Member	Description
CDC_BulkOUT_Hdlr	`ErrorCode_t(* USBD_CDC_INIT_PARAM::CDC_BulkOUT_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event))(USBD_HANDLE_T hUsb, void *data, uint32_t event)` Communication Device Class specific BULK OUT endpoint handler. The application software should provide the BULK OUT endpoint handler. Applications should transfer data depending on the communication protocol type set in descriptors. Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.
SendEncpsCmd	`ErrorCode_t(* USBD_CDC_INIT_PARAM::SendEncpsCmd)(USBD_HANDLE_T hCDC, uint8_t *buffer, uint16_t len)` Abstract control model(ACM) subclass specific SEND_ENCAPSULATED_COMMAND request call-back function. This function is provided by the application software. This function gets called when host sends a SEND_ENCAPSULATED_COMMAND set request. Parameters: 1. hCdc = Handle to CDC function driver. 2. buffer = Pointer to the command buffer. 3. len = Length of the command buffer. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.
GetEncpsResp	`ErrorCode_t(* USBD_CDC_INIT_PARAM::GetEncpsResp)(USBD_HANDLE_T hCDC, uint8_t **buffer, uint16_t *len)` Abstract control model(ACM) subclass specific GET_ENCAPSULATED_RESPONSE request call-back function. This function is provided by the application software. This function gets called when host sends a GET_ENCAPSULATED_RESPONSE request. Parameters: 1. hCdc = Handle to CDC function driver. 2. buffer = Pointer to a pointer of data buffer containing response data. Pointer-to-pointer is used to implement zero-copy buffers. See USBD_ZeroCopy for more details on zero-copy concept. 3. len = Amount of data to be sent back to host. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

Table 520. USBD_CDC_INIT_PARAM class structure

Member	Description
SetCommFeature	`ErrorCode_t(* USBD_CDC_INIT_PARAM::SetCommFeature)(USBD_HANDLE_T hCDC, uint16_t feature, uint8_t *buffer, uint16_t len)` Abstract control model(ACM) subclass specific SET_COMM_FEATURE request call-back function. This function is provided by the application software. This function gets called when host sends a SET_COMM_FEATURE set request. Parameters: 1. hCdc = Handle to CDC function driver. 2. feature = Communication feature type. 3. buffer = Pointer to the settings buffer for the specified communication feature. 4. len = Length of the request buffer. Returns: The call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.
GetCommFeature	`ErrorCode_t(* USBD_CDC_INIT_PARAM::GetCommFeature)(USBD_HANDLE_T hCDC, uint16_t feature, uint8_t **pBuffer, uint16_t *len)` Abstract control model(ACM) subclass specific GET_COMM_FEATURE request call-back function. This function is provided by the application software. This function gets called when host sends a GET_ENCAPSULATED_RESPONSE request. Parameters: 1. hCdc = Handle to CDC function driver. 2. feature = Communication feature type. 3. buffer = Pointer to a pointer of data buffer containing current settings for the communication feature. Pointer-to-pointer is used to implement zero-copy buffers. 4. len = Amount of data to be sent back to host. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

Table 520. USBD_CDC_INIT_PARAM class structure

Member	Description
ClrCommFeature	`ErrorCode_t(* USBD_CDC_INIT_PARAM::ClrCommFeature)(USBD_HANDLE_T hCDC, uint16_t feature)` Abstract control model(ACM) subclass specific CLEAR_COMM_FEATURE request call-back function. This function is provided by the application software. This function gets called when host sends a CLEAR_COMM_FEATURE request. In the call-back the application should Clears the settings for a particular communication feature. Parameters: 1. hCdc = Handle to CDC function driver. 2. feature = Communication feature type. See usbcdc11.pdf, section 6.2.4, Table 47. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.
SetCtrlLineState	`ErrorCode_t(* USBD_CDC_INIT_PARAM::SetCtrlLineState)(USBD_HANDLE_T hCDC, uint16_t state)` Abstract control model(ACM) subclass specific SET_CONTROL_LINE_STATE request call-back function. This function is provided by the application software. This function gets called when host sends a SET_CONTROL_LINE_STATE request. RS-232 signal used to tell the DCE device the DTE device is now present Parameters: 1. hCdc = Handle to CDC function driver. 2. state = The state value uses bitmap values defined the *USB CDC class specification document* published by usb.org. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.
SendBreak	`ErrorCode_t(* USBD_CDC_INIT_PARAM::SendBreak)(USBD_HANDLE_T hCDC, uint16_t mstime)` Abstract control model(ACM) subclass specific SEND_BREAK request call-back function. This function is provided by the application software. This function gets called when host sends a SEND_BREAK request. Parameters: 1. hCdc = Handle to CDC function driver. 2. mstime = Duration of Break signal in milliseconds. If mstime is FFFFh, then the application should send break until another SendBreak request is received with the wValue of 0000h. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

Table 520. USBD_CDC_INIT_PARAM class structure

Member	Description
SetLineCode	`ErrorCode_t(* USBD_CDC_INIT_PARAM::SetLineCode)(USBD_HANDLE_T hCDC, CDC_LINE_CODING *line_coding)` Abstract control model(ACM) subclass specific SET_LINE_CODING request call-back function. This function is provided by the application software. This function gets called when host sends a SET_LINE_CODING request. The application should configure the device per DTE rate, stop-bits, parity, and number-of-character bits settings provided in command buffer. Parameters: 1. hCdc = Handle to CDC function driver. 2. line_coding = Pointer to the CDC_LINE_CODING command buffer. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.
CDC_InterruptEP_Hdlr	`ErrorCode_t(* USBD_CDC_INIT_PARAM::CDC_InterruptEP_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event)` Optional Communication Device Class specific INTERRUPT IN endpoint handler. The application software should provide the INT IN endpoint handler. Applications should transfer data depending on the communication protocol type set in descriptors. Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.
CDC_Ep0_Hdlr	`ErrorCode_t(* USBD_CDC_INIT_PARAM::CDC_Ep0_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event)` Optional user override-able function to replace the default CDC class handler. The application software could override the default EP0 class handler with their own by providing the handler function address as this data member of the parameter structure. Application which like the default handler should set this data member to zero before calling the USBD_CDC_API::Init(). Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

32.4.29 USBD_CORE_API

USBD stack Core API functions structure.

Table 521. USBD_CORE_API class structure

Member	Description
RegisterClassHandler	`ErrorCode_t(*ErrorCode_t USBD_CORE_API::RegisterClassHandler)(USBD_HANDLE_T hUsb, USB_EP_HANDLER_T pfn, void *data)` Function to register class specific EP0 event handler with USB device stack. The application layer uses this function when it has to register the custom class's EP0 handler. The stack calls all the registered class handlers on any EP0 event before going through default handling of the event. This gives the class handlers to implement class specific request handlers and also to override the default stack handling for a particular event targeted to the interface. Check USB_EP_HANDLER_T for more details on how the callback function should be implemented. Also application layer could use this function to register EP0 handler which responds to vendor specific requests. Parameters: 1. hUsb = Handle to the USB device stack. 2. pfn = Class specific EP0 handler function. 3. data = Pointer to the data which will be passed when callback function is called by the stack. Returns: Returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success 2. ERR_USBD_TOO_MANY_CLASS_HDLR(0x0004000c) = The number of class handlers registered is greater than the number of handlers allowed by the stack.
RegisterEpHandler	`ErrorCode_t(*ErrorCode_t USBD_CORE_API::RegisterEpHandler)(USBD_HANDLE_T hUsb, uint32_t ep_index, USB_EP_HANDLER_T pfn, void *data)` Function to register interrupt/event handler for the requested endpoint with USB device stack. The application layer uses this function to register the custom class's EP0 handler. The stack calls all the registered class handlers on any EP0 event before going through default handling of the event. This gives the class handlers to implement class specific request handlers and also to override the default stack handling for a particular event targeted to the interface. Check USB_EP_HANDLER_T for more details on how the callback function should be implemented. Parameters: 1. hUsb = Handle to the USB device stack. 2. ep_index = Class specific EP0 handler function. 3. pfn = Class specific EP0 handler function. 4. data = Pointer to the data which will be passed when callback function is called by the stack. Returns: ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success 2. ERR_USBD_TOO_MANY_CLASS_HDLR(0x0004000c) = Too many endpoint handlers.

Table 521. USBD_CORE_API class structure

Member	Description
SetupStage	`void(*void USBD_CORE_API::SetupStage)(USBD_HANDLE_T hUsb)` Function to set EP0 state machine in setup state. This function is called by USB stack and the application layer to set the EP0 state machine in setup state. This function will read the setup packet received from USB host into stack's buffer. **Remark:** This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly.Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. Returns: nothing.
DataInStage	`void(*void USBD_CORE_API::DataInStage)(USBD_HANDLE_T hUsb)` Function to set EP0 state machine in data_in state. This function is called by USB stack and the application layer to set the EP0 state machine in data_in state. This function will write the data present in EP0Data buffer to EP0 FIFO for transmission to host. **Remark:** This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly.Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. Returns: nothing.
DataOutStage	`void(*void USBD_CORE_API::DataOutStage)(USBD_HANDLE_T hUsb)` Function to set EP0 state machine in data_out state. This function is called by USB stack and the application layer to set the EP0 state machine in data_out state. This function will read the control data (EP0 out packets) received from USB host into EP0Data buffer. **Remark:** This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly.Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. Returns: nothing.
StatusInStage	`void(*void USBD_CORE_API::StatusInStage)(USBD_HANDLE_T hUsb)` Function to set EP0 state machine in status_in state. This function is called by USB stack and the application layer to set the EP0 state machine in status_in state. This function will send zero length IN packet on EP0 to host, indicating positive status. **Remark:** This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly.Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. Returns: nothing.

Table 521. USBD_CORE_API class structure

Member	Description
StatusOutStage	`void(*void USBD_CORE_API::StatusOutStage)(USBD_HANDLE_T hUsb)` Function to set EP0 state machine in status_out state. This function is called by USB stack and the application layer to set the EP0 state machine in status_out state. This function will read the zero length OUT packet received from USB host on EP0. **Remark:** This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly.Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. Returns: nothing.
StallEp0	`void(*void USBD_CORE_API::StallEp0)(USBD_HANDLE_T hUsb)` Function to set EP0 state machine in stall state. This function is called by USB stack and the application layer to generate STALL signalling on EP0 endpoint. This function will also reset the EP0Data buffer. **Remark:** This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly.Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. Returns: nothing.

32.4.30 USBD_DFU_API

DFU class API functions structure.This module exposes functions which interact directly with USB device controller hardware.

Table 522. USBD_DFU_API class structure

Member	Description
GetMemSize	`uint32_t(*uint32_t USBD_DFU_API::GetMemSize)(USBD_DFU_INIT_PARAM_T *param)` Function to determine the memory required by the DFU function driver module. This function is called by application layer before calling pUsbApi->dfu->Init(), to allocate memory used by DFU function driver module. The application should allocate the memory which is accessible by USB controller/DMA controller. **Remark:** Some memory areas are not accessible by all bus masters. Parameters: 1. param = Structure containing DFU function driver module initialization parameters. Returns: the required memory size in bytes.

Table 522. USBD_DFU_API class structure

Member	Description
init	`ErrorCode_t(*ErrorCode_t USBD_DFU_API::init)(USBD_HANDLE_T hUsb, USBD_DFU_INIT_PARAM_T *param, uint32_t init_state)` Function to initialize DFU function driver module. This function is called by application layer to initialize DFU function driver module. Parameters: 1. hUsb = Handle to the USB device stack. 2. param = Structure containing DFU function driver module initialization parameters. Returns:ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success 2. ERR_USBD_BAD_MEM_BUF = Memory buffer passed is not 4-byte aligned or smaller than required. 3. ERR_API_INVALID_PARAM2 = Either DFU_Write() or DFU_Done() or DFU_Read() callbacks are not defined. 4. ERR_USBD_BAD_DESC = USB_DFU_DESCRIPTOR_TYPE is not defined immediately after interface descriptor.wTransferSize in descriptor doesn't match the value passed in param->wTransferSize.DFU_Detach() is not defined while USB_DFU_WILL_DETACH is set in DFU descriptor. 5. ERR_USBD_BAD_INTF_DESC = Wrong interface descriptor is passed.

32.4.31 USBD_DFU_INIT_PARAM

USB descriptors data structure.

Table 523. USBD_DFU_INIT_PARAM class structure

Member	Description
mem_base	`uint32_t USBD_DFU_INIT_PARAM::mem_base` Base memory location from where the stack can allocate data and buffers. **Remark:** The memory address set in this field should be accessible by USB DMA controller. Also this value should be aligned on 4 byte boundary.
mem_size	`uint32_t USBD_DFU_INIT_PARAM::mem_size` The size of memory buffer which stack can use. **Remark:** The mem_size should be greater than the size returned by USBD_DFU_API::GetMemSize() routine.
wTransferSize	`uint16_t USBD_DFU_INIT_PARAM::wTransferSize` DFU transfer block size in number of bytes. This value should match the value set in DFU descriptor provided as part of the descriptor array (
pad	`uint16_t USBD_DFU_INIT_PARAM::pad`
intf_desc	`uint8_t * USBD_DFU_INIT_PARAM::intf_desc` Pointer to the DFU interface descriptor within the descriptor array (

Table 523. USBD_DFU_INIT_PARAM class structure

Member	Description
DFU_Write	`uint8_t(*uint8_t(* USBD_DFU_INIT_PARAM::DFU_Write)(uint32_t block_num, uint8_t **src, uint32_t length, uint8_t *bwPollTimeout))(uint32_t block_num, uint8_t **src, uint32_t length, uint8_t *bwPollTimeout)` DFU Write callback function. This function is provided by the application software. This function gets called when host sends a write command. For application using zero-copy buffer scheme this function is called for the first time with Parameters: 1. block_num = Destination start address. 2. src = Pointer to a pointer to the source of data. Pointer-to-pointer is used to implement zero-copy buffers. See Zero-Copy Data Transfer model for more details on zero-copy concept. 3. bwPollTimeout = Pointer to a 3 byte buffer which the callback implementer should fill with the amount of minimum time, in milliseconds, that the host should wait before sending a subsequent DFU_GETSTATUS request. 4. length = Number of bytes to be written. Returns: DFU_STATUS_ values defined in mw_usbd_dfu.h.
DFU_Read	`uint32_t(*uint32_t(* USBD_DFU_INIT_PARAM::DFU_Read)(uint32_t block_num, uint8_t **dst, uint32_t length))(uint32_t block_num, uint8_t **dst, uint32_t length)` DFU Read callback function. This function is provided by the application software. This function gets called when host sends a read command. Parameters: 1. block_num = Destination start address. 2. dst = Pointer to a pointer to the source of data. Pointer-to-pointer is used to implement zero-copy buffers. See Zero-Copy Data Transfer model for more details on zero-copy concept. 3. length = Amount of data copied to destination buffer. Returns: DFU_STATUS_ values defined in mw_usbd_dfu.h.
DFU_Done	`void(*USBD_DFU_INIT_PARAM::DFU_Done)(void)` DFU done callback function. This function is provided by the application software. This function gets called after download is finished. Returns: nothing.
DFU_Detach	`void(* USBD_DFU_INIT_PARAM::DFU_Detach)(USBD_HANDLE_T hUsb)` DFU detach callback function. This function is provided by the application software. This function gets called after USB_REQ_DFU_DETACH is received. Applications which set USB_DFU_WILL_DETACH bit in DFU descriptor should define this function. As part of this function application can call Connect() routine to disconnect and then connect back with host. For application which rely on WinUSB based host application should use this feature since USB reset can be invoked only by kernel drivers on Windows host. By implementing this feature host doesn't have to issue reset instead the device has to do it automatically by disconnect and connect procedure. hUsbHandle DFU control structure. Parameters: 1. hUsb = Handle DFU control structure. Returns: nothing.

Table 523. USBD_DFU_INIT_PARAM class structure

Member	Description
DFU_Ep0_Hdlr	`ErrorCode_t(* USBD_DFU_INIT_PARAM::DFU_Ep0_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event)` Optional user overridable function to replace the default DFU class handler. The application software could override the default EP0 class handler with their own by providing the handler function address as this data member of the parameter structure. Application which like the default handler should set this data member to zero before calling the USBD_DFU_API::Init(). Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

32.4.32 USBD_HID_API

HID class API functions structure.This structure contains pointers to all the function exposed by HID function driver module.

Table 524. USBD_HID_API class structure

Member	Description
GetMemSize	`uint32_t(*uint32_t USBD_HID_API::GetMemSize)(USBD_HID_INIT_PARAM_T *param)` Function to determine the memory required by the HID function driver module. This function is called by application layer before calling pUsbApi->hid->Init(), to allocate memory used by HID function driver module. The application should allocate the memory which is accessible by USB controller/DMA controller. **Remark:** Some memory areas are not accessible by all bus masters. Parameters: 1. param = Structure containing HID function driver module initialization parameters. Returns: the required memory size in bytes.

Table 524. USBD_HID_API class structure

Member	Description
init	`ErrorCode_t(*ErrorCode_t USBD_HID_API::init)(USBD_HANDLE_T hUsb, USBD_HID_INIT_PARAM_T *param)` Function to initialize HID function driver module. This function is called by application layer to initialize HID function driver module. On successful initialization the function returns a handle to HID function driver module in passed param structure. Parameters: 1. hUsb = Handle to the USB device stack. 2. param = Structure containing HID function driver module initialization parameters. Returns: ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success 2. ERR_USBD_BAD_MEM_BUF = Memory buffer passed is not 4-byte aligned or smaller than required. 3. ERR_API_INVALID_PARAM2 = Either HID_GetReport() or HID_SetReport() callback are not defined. 4. ERR_USBD_BAD_DESC = HID_HID_DESCRIPTOR_TYPE is not defined immediately after interface descriptor. 5. ERR_USBD_BAD_INTF_DESC = Wrong interface descriptor is passed. 6. ERR_USBD_BAD_EP_DESC = Wrong endpoint descriptor is passed.

32.4.33 USBD_HID_INIT_PARAM

USB descriptors data structure.

Table 525. USBD_HID_INIT_PARAM class structure

Member	Description
mem_base	`uint32_t USBD_HID_INIT_PARAM::mem_base` Base memory location from where the stack can allocate data and buffers. **Remark:** The memory address set in this field should be accessible by USB DMA controller. Also this value should be aligned on 4 byte boundary.
mem_size	`uint32_t USBD_HID_INIT_PARAM::mem_size` The size of memory buffer which stack can use. **Remark:** The mem_size should be greater than the size returned by USBD_HID_API::GetMemSize() routine.
max_reports	`uint8_t USBD_HID_INIT_PARAM::max_reports` Number of HID reports supported by this instance of HID class driver.
pad	`uint8_t USBD_HID_INIT_PARAM::pad[3][3]`
intf_desc	`uint8_t * USBD_HID_INIT_PARAM::intf_desc` Pointer to the HID interface descriptor within the descriptor array (
report_data	`USB_HID_REPORT_T *USB_HID_REPORT_T* USBD_HID_INIT_PARAM::report_data` Pointer to an array of HID report descriptor data structure (**Remark:** This array should be of global scope.

Table 525. USBD_HID_INIT_PARAM class structure

Member	Description
HID_GetReport	`ErrorCode_t(* USBD_HID_INIT_PARAM::HID_GetReport)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuffer, uint16_t *length)` HID get report callback function. This function is provided by the application software. This function gets called when host sends a HID_REQUEST_GET_REPORT request. The setup packet data (**Remark:** HID reports are sent via interrupt IN endpoint also. This function is called only when report request is received on control endpoint. Application should implement HID_EpIn_Hdlr to send reports to host via interrupt IN endpoint. Parameters: 1. hHid = Handle to HID function driver. 2. pSetup = Pointer to setup packet received from host. 3. pBuffer = Pointer to a pointer of data buffer containing report data. Pointer-to-pointer is used to implement zero-copy buffers. See Zero-Copy Data Transfer model for more details on zero-copy concept. 4. length = Amount of data copied to destination buffer. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.
HID_SetReport	`ErrorCode_t(* USBD_HID_INIT_PARAM::HID_SetReport)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuffer, uint16_t length)` HID set report callback function. This function is provided by the application software. This function gets called when host sends a HID_REQUEST_SET_REPORT request. The setup packet data (Parameters: 1. hHid = Handle to HID function driver. 2. pSetup = Pointer to setup packet received from host. 3. pBuffer = Pointer to a pointer of data buffer containing report data. Pointer-to-pointer is used to implement zero-copy buffers. See Zero-Copy Data Transfer model for more details on zero-copy concept. 4. length = Amount of data copied to destination buffer. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

Table 525. USBD_HID_INIT_PARAM class structure

Member	Description
HID_GetPhysDesc	`ErrorCode_t(* USBD_HID_INIT_PARAM::HID_GetPhysDesc)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuf, uint16_t *length)` Optional callback function to handle HID_GetPhysDesc request. The application software could provide this callback HID_GetPhysDesc handler to handle get physical descriptor requests sent by the host. When host requests Physical Descriptor set 0, application should return a special descriptor identifying the number of descriptor sets and their sizes. A Get_Descriptor request with the Physical Index equal to 1 should return the first Physical Descriptor set. A device could possibly have alternate uses for its items. These can be enumerated by issuing subsequent Get_Descriptor requests while incrementing the Descriptor Index. A device should return the last descriptor set to requests with an index greater than the last number defined in the HID descriptor. **Remark:** Applications which don't have physical descriptor should set this data member to zero before calling the USBD_HID_API::Init(). Parameters: 1. hHid = Handle to HID function driver. 2. pSetup = Pointer to setup packet received from host. 3. pBuf = Pointer to a pointer of data buffer containing physical descriptor data. If the physical descriptor is in USB accessible memory area application could just update the pointer or else it should copy the descriptor to the address pointed by this pointer. 4. length = Amount of data copied to destination buffer or descriptor length. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.
HID_SetIdle	`ErrorCode_t(* USBD_HID_INIT_PARAM::HID_SetIdle)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t idleTime)` Optional callback function to handle HID_REQUEST_SET_IDLE request. The application software could provide this callback to handle HID_REQUEST_SET_IDLE requests sent by the host. This callback is provided to applications to adjust timers associated with various reports, which are sent to host over interrupt endpoint. The setup packet data (**Remark:** Applications which don't send reports on Interrupt endpoint or don't have idle time between reports should set this data member to zero before calling the USBD_HID_API::Init(). Parameters: 1. hHid = Handle to HID function driver. 2. pSetup = Pointer to setup packet recived from host. 3. idleTime = Idle time to be set for the specified report. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

Table 525. USBD_HID_INIT_PARAM class structure

Member	Description
HID_SetProtocol	`ErrorCode_t(* USBD_HID_INIT_PARAM::HID_SetProtocol)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t protocol)` Optional callback function to handle HID_REQUEST_SET_PROTOCOL request. The application software could provide this callback to handle HID_REQUEST_SET_PROTOCOL requests sent by the host. This callback is provided to applications to adjust modes of their code between boot mode and report mode. **Remark:** Applications which don't support protocol modes should set this data member to zero before calling the USBD_HID_API::Init(). Parameters: 1. hHid = Handle to HID function driver. 2. pSetup = Pointer to setup packet recived from host. 3. protocol = Protocol mode. 0 = Boot Protocol 1 = Report Protocol Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.
HID_EpIn_Hdlr	`ErrorCode_t(* USBD_HID_INIT_PARAM::HID_EpIn_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event)` Optional Interrupt IN endpoint event handler. The application software could provide Interrupt IN endpoint event handler. Application which send reports to host on interrupt endpoint should provide an endpoint event handler through this data member. This data member is ignored if the interface descriptor Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Handle to HID function driver. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

Table 525. USBD_HID_INIT_PARAM class structure

Member	Description
HID_EpOut_Hdlr	`ErrorCode_t(* USBD_HID_INIT_PARAM::HID_EpOut_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event)` Optional Interrupt OUT endpoint event handler. The application software could provide Interrupt OUT endpoint event handler. Application which receives reports from host on interrupt endpoint should provide an endpoint event handler through this data member. This data member is ignored if the interface descriptor Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Handle to HID function driver. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.
HID_GetReportDesc	`ErrorCode_t(* USBD_HID_INIT_PARAM::HID_GetReportDesc)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuf, uint16_t *length)` Optional user overridable function to replace the default HID_GetReportDesc handler. The application software could override the default HID_GetReportDesc handler with their own by providing the handler function address as this data member of the parameter structure. Application which like the default handler should set this data member to zero before calling the USBD_HID_API::Init() and also provide report data array Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

Table 525. USBD_HID_INIT_PARAM class structure

Member	Description
HID_Ep0_Hdlr	`ErrorCode_t(* USBD_HID_INIT_PARAM::HID_Ep0_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event)` Optional user overridable function to replace the default HID class handler. The application software could override the default EP0 class handler with their own by providing the handler function address as this data member of the parameter structure. Application which like the default handler should set this data member to zero before calling the USBD_HID_API::Init(). Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

32.4.34 USBD_HW_API

Hardware API functions structure.This module exposes functions which interact directly with USB device controller hardware.

Table 526. USBD_HW_API class structure

Member	Description
GetMemSize	`uint32_t(*uint32_t USBD_HW_API::GetMemSize)(USBD_API_INIT_PARAM_T *param)` Function to determine the memory required by the USB device stack's DCD and core layers. This function is called by application layer before calling pUsbApi->hw-> **Remark:** Some memory areas are not accessible by all bus masters. Parameters: 1. param = Structure containing USB device stack initialization parameters. Returns: the required memory size in bytes.
Init	`ErrorCode_t(*ErrorCode_t USBD_HW_API::Init)(USBD_HANDLE_T *phUsb, USB_CORE_DESCS_T *pDesc, USBD_API_INIT_PARAM_T *param)` Function to initialize USB device stack's DCD and core layers. This function is called by application layer to initialize USB hardware and core layers. On successful initialization the function returns a handle to USB device stack which should be passed to the rest of the functions. Parameters: 1. phUsb = Pointer to the USB device stack handle of type USBD_HANDLE_T. 2. param = Structure containing USB device stack initialization parameters. Returns: ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK(0) = On success 2. ERR_USBD_BAD_MEM_BUF(0x0004000b) = When insufficient memory buffer is passed or memory is not aligned on 2048 boundary.

Table 526. USBD_HW_API class structure

Member	Description
Connect	`void(*void USBD_HW_API::Connect)(USBD_HANDLE_T hUsb, uint32_t con)` Function to make USB device visible/invisible on the USB bus. This function is called after the USB initialization. This function uses the soft connect feature to make the device visible on the USB bus. This function is called only after the application is ready to handle the USB data. The enumeration process is started by the host after the device detection. The driver handles the enumeration process according to the USB descriptors passed in the USB initialization function. Parameters: 1. hUsb = Handle to the USB device stack. 2. con = States whether to connect (1) or to disconnect (0). Returns: nothing.
ISR	`void(*void USBD_HW_API::ISR)(USBD_HANDLE_T hUsb)` Function to USB device controller interrupt events. When the user application is active the interrupt handlers are mapped in the user flash space. The user application must provide an interrupt handler for the USB interrupt and call this function in the interrupt handler routine. The driver interrupt handler takes appropriate action according to the data received on the USB bus. Parameters: 1. hUsb = Handle to the USB device stack. Returns: nothing.
Reset	`void(*void USBD_HW_API::Reset)(USBD_HANDLE_T hUsb)` Function to Reset USB device stack and hardware controller. Reset USB device stack and hardware controller. Disables all endpoints except EP0. Clears all pending interrupts and resets endpoint transfer queues. This function is called internally by pUsbApi->hw->init() and from reset event. Parameters: 1. hUsb = Handle to the USB device stack. Returns: nothing.
ForceFullSpeed	`void(*void USBD_HW_API::ForceFullSpeed)(USBD_HANDLE_T hUsb, uint32_t cfg)` Function to force high speed USB device to operate in full speed mode. This function is useful for testing the behavior of current device when connected to a full speed only hosts. Parameters: 1. hUsb = Handle to the USB device stack. 2. cfg = When 1 - set force full-speed or 0 - clear force full-speed. Returns: nothing.

Table 526. USBD_HW_API class structure

Member	Description
WakeUpCfg	`void(*void USBD_HW_API::WakeUpCfg)(USBD_HANDLE_T hUsb, uint32_t cfg)` Function to configure USB device controller to walk-up host on remote events. This function is called by application layer to configure the USB device controller to wake up on remote events. It is recommended to call this function from users's USB_WakeUpCfg() callback routine registered with stack. **Remark:** User's USB_WakeUpCfg() is registered with stack by setting the USB_WakeUpCfg member of USBD_API_INIT_PARAM_T structure before calling pUsbApi->hw->Init() routine. Certain USB device controllers needed to keep some clocks always on to generate resume signaling through pUsbApi->hw->WakeUp(). This hook is provided to support such controllers. In most controllers cases this is an empty routine. Parameters: 1. hUsb = Handle to the USB device stack. 2. cfg = When 1 - Configure controller to wake on remote events or 0 - Configure controller not to wake on remote events. Returns: nothing.
SetAddress	`void(*void USBD_HW_API::SetAddress)(USBD_HANDLE_T hUsb, uint32_t adr)` Function to set USB address assigned by host in device controller hardware. This function is called automatically when USB_REQUEST_SET_ADDRESS request is received by the stack from USB host. This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly. Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. 2. adr = USB bus Address to which the device controller should respond. Usually assigned by the USB host. Returns: nothing.
Configure	`void(*void USBD_HW_API::Configure)(USBD_HANDLE_T hUsb, uint32_t cfg)` Function to configure device controller hardware with selected configuration. This function is called automatically when USB_REQUEST_SET_CONFIGURATION request is received by the stack from USB host. This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly. Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. 2. cfg = Configuration index. Returns: nothing.

Table 526. USBD_HW_API class structure

Member	Description
ConfigEP	`void(*void USBD_HW_API::ConfigEP)(USBD_HANDLE_T hUsb, USB_ENDPOINT_DESCRIPTOR *pEPD)` Function to configure USB Endpoint according to descriptor. This function is called automatically when USB_REQUEST_SET_CONFIGURATION request is received by the stack from USB host. All the endpoints associated with the selected configuration are configured. This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly. Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. 2. pEPD = Endpoint descriptor structure defined in USB 2.0 specification. Returns: nothing.
DirCtrlEP	`void(*void USBD_HW_API::DirCtrlEP)(USBD_HANDLE_T hUsb, uint32_t dir)` Function to set direction for USB control endpoint EP0. This function is called automatically by the stack on need basis. This interface is provided to users to invoke this function in other scenarios which are not handle by current stack. In most user applications this function is not called directly. Also this function can be used by users who are selectively modifying the USB device stack's standard handlers through callback interface exposed by the stack. Parameters: 1. hUsb = Handle to the USB device stack. 2. cfg = When 1 - Set EP0 in IN transfer mode 0 - Set EP0 in OUT transfer mode Returns: nothing.
EnableEP	`void(*void USBD_HW_API::EnableEP)(USBD_HANDLE_T hUsb, uint32_t EPNum)` Function to enable selected USB endpoint. This function enables interrupts on selected endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. Returns: nothing. This function enables interrupts on selected endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number corresponding to the event as per USB specification. ie. An EP1_IN is represented by 0x81 number. For device events set this param to 0x0. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. 4. enable = 1 - enable event, 0 - disable event. Returns: ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK(0) = - On success 2. ERR_USBD_INVALID_REQ(0x00040001) = - Invalid event type.

Table 526. USBD_HW_API class structure

Member	Description
DisableEP	`void(*void USBD_HW_API::DisableEP)(USBD_HANDLE_T hUsb, uint32_t EPNum)` Function to disable selected USB endpoint. This function disables interrupts on selected endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. Returns: nothing.
ResetEP	`void(*void USBD_HW_API::ResetEP)(USBD_HANDLE_T hUsb, uint32_t EPNum)` Function to reset selected USB endpoint. This function flushes the endpoint buffers and resets data toggle logic. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. Returns: nothing.
SetStallEP	`void(*void USBD_HW_API::SetStallEP)(USBD_HANDLE_T hUsb, uint32_t EPNum)` Function to STALL selected USB endpoint. Generates STALL signalling for requested endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. Returns: nothing.
ClrStallEP	`void(*void USBD_HW_API::ClrStallEP)(USBD_HANDLE_T hUsb, uint32_t EPNum)` Function to clear STALL state for the requested endpoint. This function clears STALL state for the requested endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. Returns: nothing.
SetTestMode	`ErrorCode_t(*ErrorCode_t USBD_HW_API::SetTestMode)(USBD_HANDLE_T hUsb, uint8_t mode)` Function to set high speed USB device controller in requested test mode. USB-IF requires the high speed device to be put in various test modes for electrical testing. This USB device stack calls this function whenever it receives USB_REQUEST_CLEAR_FEATURE request for USB_FEATURE_TEST_MODE. Users can put the device in test mode by directly calling this function. Returns ERR_USBD_INVALID_REQ when device controller is full-speed only. Parameters: 1. hUsb = Handle to the USB device stack. 2. mode = Test mode defined in USB 2.0 electrical testing specification. Returns ErrorCode_t type to indicate success or error condition.: Return values: 1. LPC_OK(0) = - On success 2. ERR_USBD_INVALID_REQ(0x00040001) = - Invalid test mode or Device controller is full-speed only.

Table 526. USBD_HW_API class structure

Member	Description
ReadEP	`uint32_t(*uint32_t USBD_HW_API::ReadEP)(USBD_HANDLE_T hUsb, uint32_t EPNum, uint8_t *pData)` Function to read data received on the requested endpoint. This function is called by USB stack and the application layer to read the data received on the requested endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. 3. pData = Pointer to the data buffer where data is to be copied. Returns: Returns the number of bytes copied to the buffer.
ReadReqEP	`uint32_t(*uint32_t USBD_HW_API::ReadReqEP)(USBD_HANDLE_T hUsb, uint32_t EPNum, uint8_t *pData, uint32_t len)` Function to queue read request on the specified endpoint. This function is called by USB stack and the application layer to queue a read request on the specified endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. 3. pData = Pointer to the data buffer where data is to be copied. This buffer address should be accessible by USB DMA master. 4. len = Length of the buffer passed. Returns: the length of the requested buffer.
ReadSetupPkt	`uint32_t(*uint32_t USBD_HW_API::ReadSetupPkt)(USBD_HANDLE_T hUsb, uint32_t EPNum, uint32_t *pData)` Function to read setup packet data received on the requested endpoint. This function is called by USB stack and the application layer to read setup packet data received on the requested endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP0_IN is represented by 0x80 number. 3. pData = Pointer to the data buffer where data is to be copied. Returns: the number of bytes copied to the buffer.
WriteEP	`uint32_t(*uint32_t USBD_HW_API::WriteEP)(USBD_HANDLE_T hUsb, uint32_t EPNum, uint8_t *pData, uint32_t cnt)` Function to write data to be sent on the requested endpoint. This function is called by USB stack and the application layer to send data on the requested endpoint. Parameters: 1. hUsb = Handle to the USB device stack. 2. EPNum = Endpoint number as per USB specification. ie. An EP1_IN is represented by 0x81 number. 3. pData = Pointer to the data buffer from where data is to be copied. 4. cnt = Number of bytes to write. Returns: the number of bytes written.

Table 526. USBD_HW_API class structure

Member	Description
WakeUp	`void(*void USBD_HW_API::WakeUp)(USBD_HANDLE_T hUsb)` Function to generate resume signaling on bus for remote host wake-up. This function is called by application layer to remotely wake up host controller when system is in suspend state. Application should indicate this remote wake up capability by setting USB_CONFIG_REMOTE_WAKEUP in bmAttributes of Configuration Descriptor. Also this routine will generate resume signalling only if host enables USB_FEATURE_REMOTE_WAKEUP by sending SET_FEATURE request before suspending the bus. Parameters: 1. hUsb = Handle to the USB device stack. Returns: nothing.
EnableEvent	`ErrorCode_t(* USBD_HW_API::EnableEvent)(USBD_HANDLE_T hUsb, uint32_t EPNum, uint32_t event_type, uint32_t enable)`

32.4.35 USBD_MSC_API

MSC class API functions structure.This module exposes functions which interact directly with USB device controller hardware.

Table 527. USBD_MSC_API class structure

Member	Description
GetMemSize	`uint32_t(*uint32_t USBD_MSC_API::GetMemSize)(USBD_MSC_INIT_PARAM_T *param)` Function to determine the memory required by the MSC function driver module. This function is called by application layer before calling pUsbApi->msc->Init(), to allocate memory used by MSC function driver module. The application should allocate the memory which is accessible by USB controller/DMA controller. **Remark:** Some memory areas are not accessible by all bus masters. Parameters: 1. param = Structure containing MSC function driver module initialization parameters. Returns: the required memory size in bytes.
init	`ErrorCode_t(*ErrorCode_t USBD_MSC_API::init)(USBD_HANDLE_T hUsb, USBD_MSC_INIT_PARAM_T *param)` Function to initialize MSC function driver module. This function is called by application layer to initialize MSC function driver module. Parameters: 1. hUsb = Handle to the USB device stack. 2. param = Structure containing MSC function driver module initialization parameters. Returns: ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success 2. ERR_USBD_BAD_MEM_BUF = Memory buffer passed is not 4-byte aligned or smaller than required. 3. ERR_API_INVALID_PARAM2 = Either MSC_Write() or MSC_Read() or MSC_Verify() callbacks are not defined. 4. ERR_USBD_BAD_INTF_DESC = Wrong interface descriptor is passed. 5. ERR_USBD_BAD_EP_DESC = Wrong endpoint descriptor is passed.

32.4.36 USBD_MSC_INIT_PARAM

Mass Storage class function driver initialization parameter data structure.

Table 528. USBD_MSC_INIT_PARAM class structure

Member	Description
mem_base	`uint32_t USBD_MSC_INIT_PARAM::mem_base` Base memory location from where the stack can allocate data and buffers. **Remark:** The memory address set in this field should be accessible by USB DMA controller. Also this value should be aligned on 4 byte boundary.
mem_size	`uint32_t USBD_MSC_INIT_PARAM::mem_size` The size of memory buffer which stack can use. **Remark:** The mem_size should be greater than the size returned by USBD_MSC_API::GetMemSize() routine.
InquiryStr	`uint8_t * USBD_MSC_INIT_PARAM::InquiryStr` Pointer to the 28 character string. This string is sent in response to the SCSI Inquiry command. **Remark:** The data pointed by the pointer should be of global scope.
BlockCount	`uint32_t USBD_MSC_INIT_PARAM::BlockCount` Number of blocks present in the mass storage device
BlockSize	`uint32_t USBD_MSC_INIT_PARAM::BlockSize` Block size in number of bytes
MemorySize	`uint32_t USBD_MSC_INIT_PARAM::MemorySize` Memory size in number of bytes
intf_desc	`uint8_t * USBD_MSC_INIT_PARAM::intf_desc` Pointer to the interface descriptor within the descriptor array (
MSC_Write	`void(*void(* USBD_MSC_INIT_PARAM::MSC_Write)(uint32_t offset, uint8_t **src, uint32_t length))(uint32_t offset, uint8_t **src, uint32_t length)` MSC Write callback function. This function is provided by the application software. This function gets called when host sends a write command. Parameters: 1. offset = Destination start address. 2. src = Pointer to a pointer to the source of data. Pointer-to-pointer is used to implement zero-copy buffers. See Zero-Copy Data Transfer model for more details on zero-copy concept. 3. length = Number of bytes to be written. Returns: nothing.

Table 528. USBD_MSC_INIT_PARAM class structure

Member	Description
MSC_Read	`void(*void(* USBD_MSC_INIT_PARAM::MSC_Read)(uint32_t offset, uint8_t **dst, uint32_t length))(uint32_t offset, uint8_t **dst, uint32_t length)` MSC Read callback function. This function is provided by the application software. This function gets called when host sends a read command. Parameters: 1. offset = Source start address. 2. dst = Pointer to a pointer to the source of data. The MSC function drivers implemented in stack are written with zero-copy model. Meaning the stack doesn't make an extra copy of buffer before writing/reading data from USB hardware FIFO. Hence the parameter is pointer to a pointer containing address buffer (uint8_t** dst). So that the user application can update the buffer pointer instead of copying data to address pointed by the parameter. /note The updated buffer address should be access able by USB DMA master. If user doesn't want to use zero-copy model, then the user should copy data to the address pointed by the passed buffer pointer parameter and shouldn't change the address value. See Zero-Copy Data Transfer model for more details on zero-copy concept. 3. length = Number of bytes to be read. Returns: Nothing.
MSC_Verify	`ErrorCode_t(* USBD_MSC_INIT_PARAM::MSC_Verify)(uint32_t offset, uint8_t buf[], uint32_t length)` MSC Verify callback function. This function is provided by the application software. This function gets called when host sends a verify command. The callback function should compare the buffer with the destination memory at the requested offset and Parameters: 1. offset = Destination start address. 2. buf = Buffer containing the data sent by the host. 3. length = Number of bytes to verify. Returns: ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = If data in the buffer matches the data at destination 2. ERR_FAILED = At least one byte is different.
MSC_GetWriteBuf	`void(*void(* USBD_MSC_INIT_PARAM::MSC_GetWriteBuf)(uint32_t offset, uint8_t **buff_adr, uint32_t length))(uint32_t offset, uint8_t **buff_adr, uint32_t length)` Optional callback function to optimize MSC_Write buffer transfer. This function is provided by the application software. This function gets called when host sends SCSI_WRITE10/SCSI_WRITE12 command. The callback function should update the Parameters: 1. offset = Destination start address. 2. buf = Buffer containing the data sent by the host. 3. length = Number of bytes to write. Returns: nothing.

Table 528. USBD_MSC_INIT_PARAM class structure

Member	Description
MSC_Ep0_Hdlr	`ErrorCode_t(* USBD_MSC_INIT_PARAM::MSC_Ep0_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event)` Optional user overridable function to replace the default MSC class handler. The application software could override the default EP0 class handler with their own by providing the handler function address as this data member of the parameter structure. Application which like the default handler should set this data member to zero before calling the USBD_MSC_API::Init(). Parameters: 1. hUsb = Handle to the USB device stack. 2. data = Pointer to the data which will be passed when callback function is called by the stack. 3. event = Type of endpoint event. See USBD_EVENT_T for more details. Returns: the call back should returns ErrorCode_t type to indicate success or error condition. Return values: 1. LPC_OK = On success. 2. ERR_USBD_UNHANDLED = Event is not handled hence pass the event to next in line. 3. ERR_USBD_xxx = For other error conditions.

UM11071

Chapter 33: LPC51U68 Flash signature generator

Rev. 1.1 — 17 May 2018 User manual

33.1 How to read this chapter

The flash signature generator is present on all LPC51U68 devices.

33.2 Features

- Controls hardware flash signature generation.
- Signature generated for the entire flash or for a specified address range.

33.3 General description

The flash signature generator can generate a flash signature value for a specified address range under software control. It can also generate a flash signature value for the entire flash memory by using an IAP function call (see Section 4.5.17 for details).

The flash module contains a built-in signature generator. This generator can produce a 128-bit signature from a range of flash memory. A typical usage is to verify the flashed contents against a calculated signature (e.g. during programming). The signature generator can also be accessed via an IAP function call (Section 4.6.11) or ISP command (Section 4.5.17).

The address range for generating a signature must be aligned on flash-word boundaries, i.e. 128-bit boundaries. Once started, signature generation completes independently. While signature generation is in progress, the flash memory cannot be accessed for other purposes, and an attempted read will cause a wait state to be asserted until signature generation is complete. Code outside of the flash (e.g. internal RAM) can be executed during signature generation. This can include interrupt services, if the interrupt vector table is re-mapped to memory other than the flash memory. The code that initiates signature generation should also be placed outside of the flash memory.

33.4 Register description

Remark: To configure flash access times, use the FLASHCFG register in the SYSCON block. See Section 6.5.38.

Table 529. Register overview: FMC (base address 0x4003 4000)

Name	Access	Offset	Description	Reset value	Reference
FCTR	R/W	0x000	Control register	0x20005	33.4.1
FBWST	R/W	0x010	Wait state register	0xC005	33.4.2
FMSSTART	R/W	0x020	Signature Start address register	0	33.4.3.1
FMSSTOP	R/W	0x024	Signature Stop-address register	0	33.4.3.2
FMSW0	RO	0x02C	Word 0 of 128-bit signature word	-	33.4.4
FMSW1	RO	0x030	Word 1 of 128-bit signature word	-	33.4.4
FMSW2	RO	0x034	Word 2 of 128-bit signature word	-	33.4.4
FMSW3	RO	0x038	Word 3 of 128-bit signature word	-	33.4.4
FMSTAT	RO	0xFE0	Signature generation status register	0	33.4.5
FMSTATCLR	WO	0xFE8	Signature generation status clear register	-	33.4.6

33.4.1 Control register

The FCTR register controls the read mode for signature generation. Bits 3 and 4 must have the indicated values in order for signature generation to work correctly.

Table 530. Control register (FCTR, offset 0x000) bit description

Bit	Symbol	Description	Reset value
2:0	-	Reserved: leave existing value unchanged.	0x5
3	FS_RD0	Value must be 0 for signature generation.	0x0
4	FS_RD1	Value must be 1 for signature generation.	0x0
31:5	-	Reserved: leave existing value unchanged.	-

33.4.2 Wait state register

The FBWST registers sets the wait states used for flash signature generation. Signature generation does not use the flash timing value programmed in the FLASHTIM field of the FLASHCFG register that is used for normal flash accesses. Typically, The FBWST wait state value can be set to the same value used for normal flash operation.

Table 531. FBWST register bit description (FBWST, offset 0x02C)

Bit	Symbol	Description	Reset value
7:0	WAITSTATES	Wait states for signature generation.	0x05
31:8	-	Reserved: leave existing value unchanged.	0xC0

33.4.3 Signature generation address and control registers

These registers control automatic signature generation. A signature can be generated for any part of the flash memory contents. The address range to be used for generation is defined by writing the start address to the signature start address register (FMSSTART) and the stop address to the signature stop address register (FMSSTOP. The start and stop addresses must be aligned to 128-bit boundaries and can be derived by dividing the byte address by 16.

Signature generation is started by setting the SIG_START bit in the FMSSTOP register. Setting the SIG_START bit is typically combined with the signature stop address in a single write.

33.4.3.1 Flash signature start address register

Table 532. Signature Start register (FMSSTART, offset 0x020) bit description

Bit	Symbol	Description	Reset value
16:0	START	Signature generation start address (corresponds to AHB byte address bits[20:4]).	0
31:17	-	Reserved. Read value is undefined, only zero should be written.	-

33.4.3.2 Flash signature stop address register

Table 533. Signature Stop register (FMSSTOP, offset 0x024) bit description

Bit	Symbol	Description	Reset value
16:0	STOP	Stop address for signature generation (the word specified by STOP is included in the address range). The address is in units of memory words, not bytes.	0
17	SIG_START	When this bit is written to 1, signature generation starts. At the end of signature generation, this bit is automatically cleared.	0
31:18	-	Reserved. Read value is undefined, only zero should be written.	-

33.4.4 Signature generation result registers

The signature generation result registers return the flash signature produced by the embedded signature generator. The 128-bit signature is reflected by the four registers FMSW0, FMSW1, FMSW2 and FMSW3.

The generated flash signature can be used to verify the flash memory contents. The generated signature can be compared with an expected signature and thus saves time and code space. The method for generating the signature is described in Section 33.5.1.

Table 534. Signature Word 0 register bit description (FMSW0, offset 0x02C)

Bit	Symbol	Description	Reset value
31:0	SW0	Word 0 of 128-bit signature (bits 31 to 0).	-

Table 535. Signature Word 1 register bit description (FMSW1, offset 0x030)

Bit	Symbol	Description	Reset value
31:0	SW1	Word 1 of 128-bit signature (bits 63 to 32).	-

Table 536. Signature Word 2 register bit description (FMSW2, offset 0x034)

Bit	Symbol	Description	Reset value
31:0	SW2	Word 2 of 128-bit signature (bits 95 to 64).	-

Table 537. Signature Word 3 register bit description (FMSW3, offset 0x038)

Bit	Symbol	Description	Reset value
31:0	SW3	Word 3 of 128-bit signature (bits 127 to 96).	-

33.4.5 Signature status register

The read-only FMSTAT register provides a means of determining when signature generation has completed. Completion of signature generation can be checked by polling the SIG_DONE bit in FMSTAT. SIG_DONE should be cleared via the FMSTATCLR register before starting a signature generation operation, otherwise the status might indicate completion of a previous operation.

Table 538. Signature status register (FMSTAT, offset 0x0FE0) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved. Read value is undefined, only zero should be written.	-
2	SIG_DONE	When 1, a previously started signature generation has completed. See FMSTATCLR register description for clearing this flag.	0
31:2	-	Reserved. Read value is undefined, only zero should be written.	-

33.4.6 Signature status clear register

The FMSTATCLR register is used to clear the signature generation completion flag.

Table 539. Signature status clear register (FMSTATCLR, offset 0x0FE8) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved. Read value is undefined, only zero should be written.	-
2	SIG_DONE_CLR	Writing a 1 to this bits clears the signature generation completion flag (SIG_DONE) in the FMSTAT register.	0
31:2	-	Reserved. Read value is undefined, only zero should be written.	-

33.5 Functional description

33.5.1 Algorithm and procedure for signature generation

33.5.1.1 Signature generation

A signature can be generated for any part of the flash contents. The address range to be used for signature generation is defined by writing the start address to the FMSSTART register, and the stop address to the FMSSTOP register.

Reading of the flash memory for signature generation requires a specific setting of 2 bits in the FCTR register, and uses FBWST for wait state generation. FBWST settings are the same as for the FLASHTIM field of the FLASHCFG register (see Section 6.5.38).

The signature generation is started by writing a 1 to the SIG_START bit in the FMSSTOP register. Starting the signature generation is typically combined with defining the stop address, which is done in the STOP bits of the same register.

The time that the signature generation takes is proportional to the address range for which the signature is generated. An estimation for the duration of the signature generation is:

Duration = (WAITSTATES * tcy) + 3) x (FMSSTOP - FMSSTART + 1)

When signature generation is triggered via software, the duration is in AHB clock cycles, tcy is the time in ns for one AHB clock. The SIG_DONE bit in FMSTAT can be polled by software to determine when signature generation is complete.

After signature generation, a 128-bit signature can be read from the FMSW0 to FMSW3 registers. The 128-bit signature reflects the corrected data read from the flash. The 128-bit signature reflects flash parity bits and check bit values.

33.5.1.2 Content verification

The signature as it is read from the FMSW0 register must be equal to the reference signature. The following pseudo-code shows the algorithm to derive the reference signature:

```
sign = 0
FOR address = FMSSTART.START to FMSSTOP.STOPA
{
      FOR i = 0 TO 126
      {
            nextSign[i] = f_Q[address][i] XOR sign[i + 1]
      }
      nextSign[127] = f_Q[address][127] XOR sign[0] XOR sign[2] XOR sign[27] XOR
      sign[29]
      sign = nextSign
}
signature128 = sign
```

UM11071

Chapter 34: ARM Cortex Appendix

Rev. 1.1 — 17 May 2018 User manual

34.1 ARM Cortex-M0+ Details

ARM Limited publishes the document “Cortex™-M0+ Devices Generic User Guide”, which is available on their website at:

- For the online manual, go to “infocenter.arm.com”, then search for “cortex-mo+ user guide”. This will bring up links to chapters of the user guide.
- There are links at the bottom of user guide chapters to download a PDF file of the user guide.

This section of this manual describes the Cortex-M0+ implementation options and any other distinctions that apply for the LPC51U68 devices.

34.1.1 Cortex-M0+ implementation options

The Cortex™-M0+ provides a number of implementation options. These are given below for the LPC51U68.

- An MPU is not included for the Cortex-M0+.
- 32 interrupt slots are implemented for the Cortex-M0+. Not all interrupts are available on all part numbers.
- 2 interrupt priority bits are implemented on the Cortex-M0+, providing 4 priority levels.
- The vector table offset register is included.
- The multiplier configuration is the high speed, single-cycle version.
- Sleep mode power-saving: NXP microcontrollers extend the number of reduced power modes beyond what is directly supported by the Cortex-M0+. Details of reduced power modes and wake-up possibilities on the LPC51U68 can be found in Chapter 8.
- Reset of the Cortex-M0+ resets the CPU register bank.
- Memory features: The memory map for LPC51U68 devices is shown in Section 2.1.2.
- SysTick timer: The SysTick timer is included for the Cortex-M0+, for details see Section 20.5.

In addition, there are debug options, see Chapter 31.

UM11071

Chapter 35: Supplementary information

Rev. 1.1 — 17 May 2018 User manual

35.1 Abbreviations

Table 540. Abbreviations

Acronym	Description
ADC	Analog-to-Digital Converter
AHB	Advanced High-performance Bus
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
API	Application Programming Interface
BOD	BrownOut Detection
Boot	At power-up or chip reset, any method of importing code from an external source to execute from on-chip SRAM, or code executed in place from the external memory.
BSDL	Boundary-Scan Description Language
CRC	Cyclic Redundancy Check
DCC	Debug Communication Channel
DMA	Direct Memory Access
FIFO	First-In-First-Out
FMC	Flash Memory Controller
FRO	Internal Free-Running Oscillator, tuned to the factory specified frequency.
GPIO	General Purpose Input/Output
I2C or IIC	Inter-Integrated Circuit bus
IAP	In-Application Programming
I2S	Inter-IC Sound or Integrated Interchip Sound. A serial audio data communication method.
ISP	In-System Programming. These are methods of programming any on-chip flash on a blank or previously programmed device.
ISR	Interrupt Service Routine
JTAG	Joint Test Action Group
LIN	Local Interconnect Network
NVIC	Nested Vectored Interrupt Controller
PLL	Phase-Locked Loop
POR	Power-On Reset
PWM	Pulse Width Modulator
RAM	Random Access Memory
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
SWD	Serial-Wire Debug
TAP	Test Access Port
USART	Universal Synchronous/Asynchronous Receiver/Transmitter

35.2 References

[1] **Cortex-M0+ TRM —** ARM Cortex-M0+ Processor Technical Reference Manual

[2] **AN11538 —** AN11538 application note and code bundle (SCT cookbook)

[3] **UM10204 —** I^{2}C-bus specification and user manual

35.3 Legal information

35.3.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

35.3.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

35.3.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

I^{2}C-bus — logo is a trademark of NXP B.V.

35.4 Tables

35.5 Figures

35.6 Contents

continued >>

continued >>

continued >>

continued >>

continued >>

continued >>

continued >>

continued >>

continued >>

continued >>

Chapter 28: LPC51U68 12-bit ADC controller (ADC)

continued >>

Chapter 29: LPC51U68 Temperature sensor

Chapter 30: LPC51U68 CRC engine

Chapter 31: LPC51U68 Serial Wire Debug (SWD)

Chapter 32: LPC51U68 USB ROM API

continued >>

Date of release: 17 May 2018
Document identifier: UM11071