Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

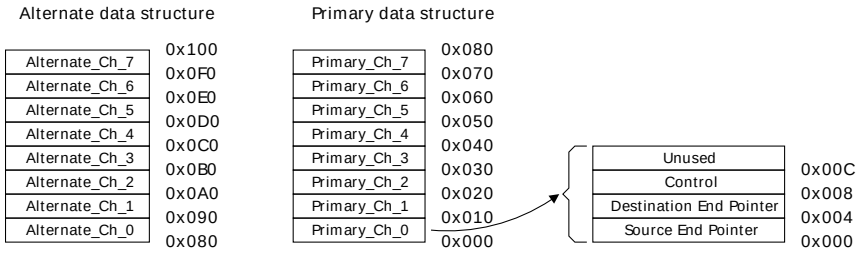
Applications of "Embedded - Microcontrollers"

Details	
Product Status	Discontinued at Digi-Key
Core Processor	ARM® Cortex®-M3
Core Size	32-Bit Single-Core
Speed	32MHz
Connectivity	EBI/EMI, I²C, IrDA, SmartCard, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, LCD, POR, PWM, WDT
Number of I/O	56
Program Memory Size	128KB (128K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	16K x 8
Voltage - Supply (Vcc/Vdd)	1.98V ~ 3.8V
Data Converters	A/D 8x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-VFQFN Exposed Pad
Supplier Device Package	64-QFN (9x9)
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/efm32g30f128-qfn64

- provide a contiguous area of system memory that the controller and host processor can access
- have a base address that is an integer multiple of the total size of the channel control data structure.

Figure 8.6 (p. 54) shows the memory that the controller requires for the channel control data structure, when all 8 channels and the optional alternate data structure are in use.

Figure 8.6. Memory map for 8 channels, including the alternate data structure



This structure in Figure 8.6 (p. 54) uses 256 bytes of system memory. The controller uses the lower 8 address bits to enable it to access all of the elements in the structure and therefore the base address must be at `0xXXXXXX00`.

You can configure the base address for the primary data structure by writing the appropriate value in the DMA_CTRLBASE register.

You do not need to set aside the full 256 bytes if all dma channels are not used or if all alternate descriptors are not used. If, for example, only 4 channels are used and they only need the primary descriptors, then only 64 bytes need to be set aside.

Table 8.6 (p. 54) lists the address bits that the controller uses when it accesses the elements of the channel control data structure.

Table 8.6. Address bit settings for the channel control data structure

Address bits					
	[7]	[6]	[5]	[4]	[3:0]
	A	C[2]	C[1]	C[0]	0x0, 0x4, or 0x8

Where:

A — Selects one of the channel control data structures:
- A = 0 Selects the primary data structure.
- A = 1 Selects the alternate data structure.

C[2:0] — Selects the DMA channel.

Address[3:0] — Selects one of the control elements:
- `0x0` Selects the source data end pointer.
- `0x4` Selects the destination data end pointer.
- `0x8` Selects the control data configuration.
- `0xC` The controller does not access this address location. If required, you can enable the host processor to use this memory location as system memory.

Note

It is not necessary for you to calculate the base address of the alternate data structure because the DMA_ALTCTRLBASE register provides this information.

Bit	Name	Description
		b1000 Arbitrates after 256 DMA transfers. b1001 Arbitrates after 512 DMA transfers. b1010 - b1111 Arbitrates after 1024 DMA transfers. This means that no arbitration occurs during the DMA transfer because the maximum transfer size is 1024.
[13:4]	n_minus_1	Prior to the DMA cycle commencing, these bits represent the total number of DMA transfers that the DMA cycle contains. You must set these bits according to the size of DMA cycle that you require. The 10-bit value indicates the number of DMA transfers, minus one. The possible values are: b000000000 = 1 DMA transfer b000000001 = 2 DMA transfers b000000010 = 3 DMA transfers b000000011 = 4 DMA transfers b000000100 = 5 DMA transfers . . . b111111111 = 1024 DMA transfers. The controller updates this field immediately prior to it entering the arbitration process. This enables the controller to store the number of outstanding DMA transfers that are necessary to complete the DMA cycle.
[3]	next_useburst	Controls if the chnl_useburst_set [C] bit is set to a 1, when the controller is performing a peripheral scatter-gather and is completing a DMA cycle that uses the alternate data structure. ***Note*** Immediately prior to completion of the DMA cycle that the alternate data structure specifies, the controller sets the chnl_useburst_set [C] bit to 0 if the number of remaining transfers is less than 2^R. The setting of the next_useburst bit controls if the controller performs an additional modification of the chnl_useburst_set [C] bit. In peripheral scatter-gather DMA cycle then after the DMA cycle that uses the alternate data structure completes , either: 0 = the controller does not change the value of the chnl_useburst_set [C] bit. If the chnl_useburst_set [C] bit is 0 then for all the remaining DMA cycles in the peripheral scatter-gather transaction, the controller responds to requests on `dma_req[ ]` and `dma_sreq[ ]`, when it performs a DMA cycle that uses an alternate data structure. 1 = the controller sets the chnl_useburst_set [C] bit to a 1. Therefore, for the remaining DMA cycles in the peripheral scatter-gather transaction, the controller only responds to requests on `dma_req[ ]`, when it performs a DMA cycle that uses an alternate data structure.
[2:0]	cycle_ctrl	The operating mode of the DMA cycle. The modes are: b000 Stop. Indicates that the data structure is invalid. b001 Basic. The controller must receive a new request, prior to it entering the arbitration process, to enable the DMA cycle to complete. b010 Auto-request. The controller automatically inserts a request for the appropriate channel during the arbitration process. This means that the initial request is sufficient to enable the DMA cycle to complete. b011 Ping-pong. The controller performs a DMA cycle using one of the data structures. After the DMA cycle completes, it performs a DMA cycle using the other data structure. After the DMA cycle completes and provided that the host processor has updated the original data structure, it performs a DMA cycle using the original data structure. The controller continues to perform DMA cycles until it either reads an invalid data structure or the host processor changes the cycle_ctrl bits to b001 or b010. See Section 8.4.2.3.4 (p. 47) . b100 Memory scatter/gather. See Section 8.4.2.3.5 (p. 49) . When the controller operates in memory scatter-gather mode, you must only use this value in the primary data structure. b101 Memory scatter/gather. See Section 8.4.2.3.5 (p. 49) . When the controller operates in memory scatter-gather mode, you must only use this value in the alternate data structure. b110 Peripheral scatter/gather. See Section 8.4.2.3.6 (p. 51) .

Example 8.1. DMA Transfer

1. Configure the channel select for using USART1 with DMA channel 0
   a. Write SOURCESEL=0b001101 and SIGSEL=XX to DMA_CHCTRL0
2. Configure the primary channel descriptor for DMA channel 0
   a. Write XX (read address of USART1) to src_data_end_ptr
   b. Write 0x20003420 + 40 to dst_data_end_ptr c
   c. Write these values to channel_cfg for channel 0:
      i. dst_inc=b01 (destination halfword address increment)
      ii. dst_size=b01 (halfword transfer size)
      iii. src_inc=b11 (no address increment for source)
      iv. src_size=01 (halfword transfer size)
      v. dst_prot_ctrl=000 (no cache/buffer/privilege)
      vi. src_prot_ctrl=000 (no cache/buffer/privilege)
      vii.R_power=b0000 (arbitrate after each DMA transfer)
      viiin_minus_1=d20 (transfer 21 halfwords)
      ix. next_useburst=b0 (not applicable)
      x. cycle_ctrl=b001 (basic operating mode)
3. Enable the DMA
   a. Write EN=1 to DMA_CONFIG
4. Disable the single requests for channel 0 (i.e., do not react to data available, wait for buffer full)
   a. Write DMA_CHUSEBURSTS[0]=1
5. Enable buffer-full requests for channel 0
   a. Write DMA_CHREQMASKC[0]=1
6. Use the primary data structure for channel 0
   a. Write DMA_CHALTC[0]=1
7. Enable channel 0
   a. Write DMA_CHENS[0]=1

Bit	Name	Reset	Access	Description

Value	Mode	Description
0	SINGLEANDBURST	Channel responds to both single and burst requests
1	BURSTONLY	Channel responds to burst requests only

8.7.8 DMA_CHUSEBURSTC - Channel Useburst Clear Register

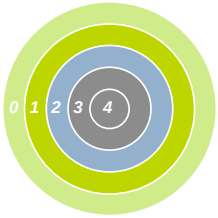
Offset	Bit Position																													
0x01C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0	0	0	0	0	0	0	0
Access																									W1	W1	W1	W1	W1	W1	W1	W1
Name																									CH7USEBURSTC	CH6USEBURSTC	CH5USEBURSTC	CH4USEBURSTC	CH3USEBURSTC	CH2USEBURSTC	CH1USEBURSTC	CH0USEBURSTC

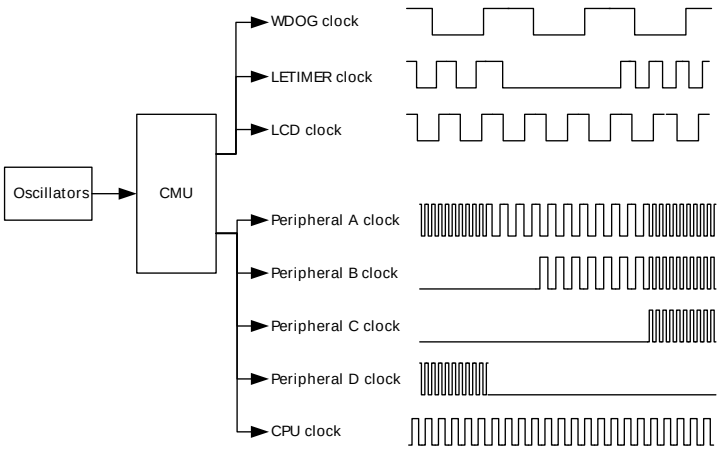
Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
7	CH7USEBURSTC	0	W1	**Channel 7 Useburst Clear**
	Write to 1 to disable useburst setting for this channel.			
6	CH6USEBURSTC	0	W1	**Channel 6 Useburst Clear**
	Write to 1 to disable useburst setting for this channel.			
5	CH5USEBURSTC	0	W1	**Channel 5 Useburst Clear**
	Write to 1 to disable useburst setting for this channel.			
4	CH4USEBURSTC	0	W1	**Channel 4 Useburst Clear**
	Write to 1 to disable useburst setting for this channel.			
3	CH3USEBURSTC	0	W1	**Channel 3 Useburst Clear**
	Write to 1 to disable useburst setting for this channel.			
2	CH2USEBURSTC	0	W1	**Channel 2 Useburst Clear**
	Write to 1 to disable useburst setting for this channel.			
1	CH1USEBURSTC	0	W1	**Channel 1 Useburst Clear**
	Write to 1 to disable useburst setting for this channel.			
0	CH0USEBURSTC	0	W1	**Channel 0 Useburst Clear**
	Write to 1 to disable useburst setting for this channel.			

8.7.9 DMA_CHREQMASKS - Channel Request Mask Set Register

Offset	Bit Position																													
0x020	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
Reset																									0	0	0	0	0	0	0	0
Access																									RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1
Name																									CH7REQMASKS	CH6REQMASKS	CH5REQMASKS	CH4REQMASKS	CH3REQMASKS	CH2REQMASKS	CH1REQMASKS	CH0REQMASKS

11 CMU - Clock Management Unit





Quick Facts

What?

The CMU controls oscillators and clocks. EFM32G supports five different oscillators with minimized power consumption and short start-up time. An additional separate RC oscillator is used for flash programming and debug trace. The CMU also has HW support for calibration of RC oscillators.

Why?

Oscillators and clocks contribute significantly to the power consumption of the MCU. With the low power oscillators combined with the flexible clock control scheme, it is possible to minimize the energy consumption in any given application.

How?

The CMU can configure different clock sources, enable/disable clocks to peripherals on an individual basis and set the prescaler for the different clocks. The short oscillator start-up times makes duty-cycling between active mode and the different low energy modes (EM2-EM4) very efficient. The calibration feature ensures high accuracy RC oscillators. Several interrupts are available to avoid CPU polling of flags.

11.1 Introduction

The Clock Management Unit (CMU) is responsible for controlling the oscillators and clocks on-board the EFM32G. The CMU provides the capability to turn on and off the clock on an individual basis to all peripheral modules in addition to enable/disable and configure the available oscillators. The high degree of flexibility enables software to minimize energy consumption in any specific application by not wasting power on peripherals and oscillators that are inactive.

11.2 Features

- Multiple clock sources available:
  - 1-28 MHz High Frequency RC Oscillator (HFRCO)
  - 4-32 MHz High Frequency Crystal Oscillator (HFXO)
  - 32.768 Hz Low Frequency RC Oscillator (LFRCO)
  - 32.768 Hz Low Frequency Crystal Oscillator (LFXO)
  - 1 kHz Ultra Low Frequency RC Oscillator (ULFRCO)
- Low power oscillators
- Low start-up times
- Separate prescaler for High Frequency Core Clocks (HFCORECLK) and Peripheral Clocks (HFPERCLK)
- Individual clock prescaler selection for each Low Energy Peripheral

Bit	Name	Reset	Access	Description

Value	Mode	Description
0	XTAL	32.768 kHz crystal oscillator.
1	BUFEXTCLK	An AC coupled buffer is coupled in series with LFXTAL_N pin, suitable for external sinus wave (32.768 kHz).
2	DIGEXTCLK	Digital external clock on LFXTAL_N pin. Oscillator is effectively bypassed.

10:9 HFXOTIMEOUT 0x3 RW **HFXO Timeout**

Configures the start-up delay for HFXO.

Value	Mode	Description
0	8CYCLES	Timeout period of 8 cycles.
1	256CYCLES	Timeout period of 256 cycles.
2	1KCYCLES	Timeout period of 1024 cycles.
3	16KCYCLES	Timeout period of 16384 cycles.

8 Reserved *To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*

7 HFXOGLITCHDETEN 0 RW **HFXO Glitch Detector Enable**

This bit enables the glitch detector which is active as long as the start-up ripple-counter is counting. A detected glitch will reset the ripple-counter effectively increasing the start-up time. Once the ripple-counter has timed-out, glitches will not be detected.

6:5 HFXOBUFCUR 0x1 RW **HFXO Boost Buffer Current**

This value has been set during calibration and should not be changed.

4 Reserved *To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*

3:2 HFXOBOOST 0x3 RW **HFXO Start-up Boost Current**

Used to adjust start-up boost current for HFXO.

Value	Mode	Description
0	50PCENT	50 %.
1	70PCENT	70 %.
2	80PCENT	80 %.
3	100PCENT	100 % (default).

1:0 HFXOMODE 0x0 RW **HFXO Mode**

Set this to configure the external source for the HFXO. The oscillator setting takes effect when 1 is written to HFXOEN in CMU_OSCENCMD. The oscillator setting is reset to default when 1 is written to HFXODIS in CMU_OSCENCMD.

Value	Mode	Description
0	XTAL	4-32 MHz crystal oscillator.
1	BUFEXTCLK	An AC coupled buffer is coupled in series with HFXTAL_N, suitable for external sine wave (4-32 MHz). The sine wave should have a minimum of 200 mV peak to peak.
2	DIGEXTCLK	Digital external clock on HFXTAL_N pin. Oscillator is effectively bypassed.

11.5.2 CMU_HFCORECLKDIV - High Frequency Core Clock Division Register

Offset	Bit Position																													
0x004	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
Reset																														0x0		
Access																														RW		
Name																														HFCORECLKDIV		

11.5.6 CMU_AUXHFRCOCTRL - AUXHFRCO Control Register

Offset	Bit Position																															
0x014	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
Reset																									0x80							
Access																									RW							
Name																									TUNING							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
7:0	TUNING	0x80	RW	**AUXHFRCO Tuning Value**
	Writing this field adjusts the AUXHFRCO frequency (the higher value, the higher frequency).This field is updated with the production calibrated value during reset, and the reset value might therefore vary between devices.			

11.5.7 CMU_CALCTRL - Calibration Control Register

Offset	Bit Position																															
0x018	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
Reset																															0x0	
Access																															RW	
Name																															UPSEL	

Bit	Name	Reset	Access	Description
31:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
2:0	UPSEL	0x0	RW	**Calibration Up-counter Select**
	Selects clock source for the calibration up-counter.			

Value	Mode	Description
0	HFXO	Select HFXO as up-counter.
1	LFXO	Select LFXO as up-counter.
2	HFRCO	Select HFRCO as up-counter.
3	LFRCO	Select LFRCO as up-counter.
4	AUXHFRCO	Select AUXHFRCO as up-counter.

Bit	Name	Reset	Access	Description
31:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
2	LOCATION	0	RW	**I/O Location**
	Decides the location of the CMU I/O pins.			

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1

1	CLKOUT1PEN	0	RW	**CLKOUT1 Pin Enable**
	When set, the CLKOUT1 pin is enabled.			
0	CLKOUT0PEN	0	RW	**CLKOUT0 Pin Enable**
	When set, the CLKOUT0 pin is enabled.			

11.5.28 CMU_LOCK - Configuration Lock Register

Offset	Bit Position																													
0x084	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
Reset																	0x0000															
Access																	RW															
Name																	LOCKKEY															

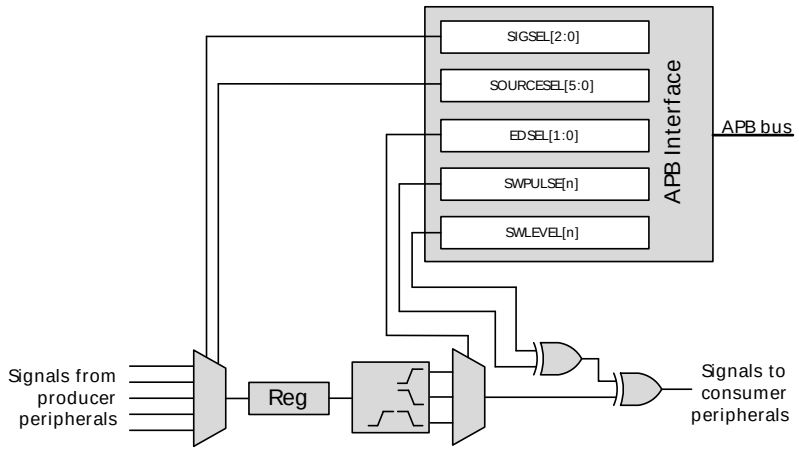
Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
15:0	LOCKKEY	0x0000	RW	**Configuration Lock Key**
	Write any other value than the unlock code to lock CMU_CTRL, CMU_HFCORECLKDIV, CMU_HFPERCLKDIV, CMU_HFRCOCTRL, CMU_LFRCOCTRL, CMU_AUXHFRCOCTRL, CMU_OSCENCMD, CMU_CMD, CMU_LFCLKSEL, CMU_HFCORECLKEN0, CMU_HFPERCLKEN0, CMU_LFACLKEN0, CMU_LFBCLKEN0, CMU_LFAPRESC0, CMU_LFBPRESC0, and CMU_PCNTCTRL from editing. Write the unlock code to unlock. When reading the register, bit 0 is set when the lock is enabled.			

Mode	Value	Description
Read Operation		
UNLOCKED	0	CMU registers are unlocked.
LOCKED	1	CMU registers are locked.
Write Operation		
LOCK	0	Lock CMU registers.
UNLOCK	0x580E	Unlock CMU registers.

13.3.1 Channel Functions

Different functions can be applied to a reflex signal within the PRS. Each channel includes an edge detector to enable generation of pulse signals from level signals. It is also possible to generate output reflex signals by configuring the SWPULSE and SWLEVEL bits. SWLEVEL is a programmable level for each channel and holds the value it is programmed to. The SWPULSE will give out a one-cycle high pulse if it is written to 1, otherwise a 0 is asserted. The SWLEVEL and SWPULSE signals are then XOR'ed with the selected input from the producers to form the output signal sent to the consumers listening to the channel.

Figure 13.1. PRS Overview



13.3.2 Producers

Each PRS channel can choose between signals from several producers, which is configured in SOURCESEL in PRS_CHx_CTRL. Each of these producers outputs one or more signals which can be selected by setting the SIGSEL field in PRS_CHx_CTRL. Setting the SOURCESEL bits to 0 (Off) leads to a constant 0 output from the input mux. An overview of the available producers is given in Table 13.1 (p. 129) .

Table 13.1. Reflex Producers

Module	Reflex Output	Output Format
ACMP	Comparator Output	Level
ADC	Single Conversion Done	Pulse
	Scan Conversion Done	Pulse
DAC	Channel 0 Conversion Done	Pulse
	Channel 0 Conversion Done	Pulse
GPIO	Pin 0 Input	Level
	Pin 1 Input	Level
	Pin 2 Input	Level
	Pin 3 Input	Level
	Pin 4 Input	Level
	Pin 5 Input	Level

Bit	Name	Reset	Access	Description
31:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
5	DDJIT	0	R	**Direct Drive Jitter Interrupt Flag**
	Set when DCLKPERIOD is not met.			
4	DDEMPTY	0	R	**Direct Drive Data Empty Interrupt Flag**
	Set when Direct Drive engine EBI_TFTDD data is empty.			
3	VFPORCH	0	R	**Vertical Front Porch Interrupt Flag**
	Set at beginning of Vertical Front Porch.			
2	VBPORCH	0	R	**Vertical Back Porch Interrupt Flag**
	Set at end of Vertical Back Porch.			
1	HSYNC	0	R	**Horizontal Sync Interrupt Flag**
	Set at Horizontal Sync pulse.			
0	VSYNC	0	R	**Vertical Sync Interrupt Flag**
	Set at Vertical Sync pulse.			

14.5.40 EBI_IFS - Interrupt Flag Set Register

Offset	Bit Position																											
0x09C	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
Reset																											0	0	0	0	0	0
Access																											W1	W1	W1	W1	W1	W1
Name																											DDJIT	DDEMPTY	VFPORCH	VBPORCH	HSYNC	VSYNC

Bit	Name	Reset	Access	Description
31:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
5	DDJIT	0	W1	**Direct Drive Jitter Interrupt Flag Set**
	Write to 1 to set Direct Drive Jitter Interrupt flag.			
4	DDEMPTY	0	W1	**Direct Drive Data Empty Interrupt Flag Set**
	Write to 1 to set Direct Drive Data Empty Interrupt flag.			
3	VFPORCH	0	W1	**Vertical Front Porch Interrupt Flag Set**
	Write to 1 to set Vertical Front Porch Interrupt flag.			
2	VBPORCH	0	W1	**Vertical Back Porch Interrupt Flag Set**
	Write to 1 to set Vertical Back Porch Interrupt flag.			
1	HSYNC	0	W1	**Horizontal Sync Interrupt Flag Set**
	Write to 1 to set Horizontal Sync interrupt flag.			
0	VSYNC	0	W1	**Vertical Sync Interrupt Flag Set**
	Write to 1 to set Vertical Sync interrupt flag.			

15.5.3 I2Cn_STATE - State Register

Offset	Bit Position																															
0x008	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
Reset																									0x0			0	0	0	0	1
Access																									R			R	R	R	R	R
Name																									STATE			BUSHOLD	NACKED	TRANSMITTER	MASTER	BUSY

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
7:5	STATE	0x0	R	**Transmission State**

The state of any current transmission. Cleared if the I^{2}C module is idle.

Value	Mode	Description
0	IDLE	No transmission is being performed.
1	WAIT	Waiting for idle. Will send a start condition as soon as the bus is idle.
2	START	Start transmitted or received
3	ADDR	Address transmitted or received
4	ADDRACK	Address ack/nack transmitted or received
5	DATA	Data transmitted or received
6	DATAACK	Data ack/nack transmitted or received

Bit	Name	Reset	Access	Description
4	BUSHOLD	0	R	**Bus Held**

Set if the bus is currently being held by this I^{2}C module.

Bit	Name	Reset	Access	Description
3	NACKED	0	R	**Nack Received**

Set if a NACK was received and STATE is ADDRACK or DATAACK.

Bit	Name	Reset	Access	Description
2	TRANSMITTER	0	R	**Transmitter**

Set when operating as a master transmitter or a slave transmitter. When cleared, the system may be operating as a master receiver, a slave receiver or the current mode is not known.

Bit	Name	Reset	Access	Description
1	MASTER	0	R	**Master**

Set when operating as an I^{2}C master. When cleared, the system may be operating as an I^{2}C slave.

Bit	Name	Reset	Access	Description
0	BUSY	1	R	**Bus Busy**

Set when the bus is busy. Whether the I^{2}C module is in control of the bus or not has no effect on the value of this bit. When the MCU comes out of reset, the state of the bus is not known, and thus BUSY is set. Use the ABORT command or a bus idle timeout to force the I^{2}C module out of the BUSY state.

15.5.4 I2Cn_STATUS - Status Register

Offset	Bit Position																															
0x00C	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
Reset																								0	1	0	0	0	0	0	0	0
Access																								R	R	R	R	R	R	R	R	R
Name																								RXDATAV	TXBL	TXC	PABORT	PCONT	PNACK	PACK	PSTOP	PSTART

Bit	Name	Reset	Access	Description
31:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		

15.5.9 I2Cn_RXDATAP - Receive Buffer Data Peek Register

Offset	Bit Position																															
0x020	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
Reset																													0x00			
Access																													R			
Name																													RXDATAP			

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
7:0	RXDATAP	0x00	R	**RX Data Peek**
	Use this register to read from the receive buffer. Buffer is not emptied on read access.			

15.5.10 I2Cn_TXDATA - Transmit Buffer Data Register

Offset	Bit Position																															
0x024	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
Reset																													0x00			
Access																													W			
Name																													TXDATA			

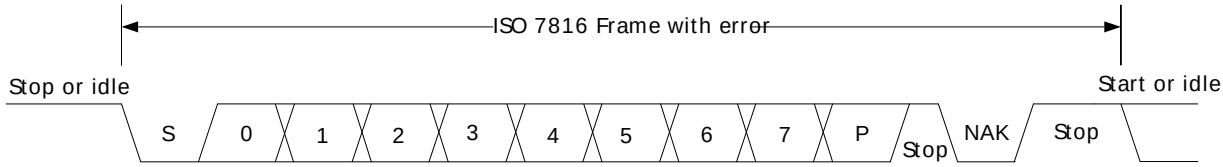
Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
7:0	TXDATA	0x00	W	**TX Data**
	Use this register to write a byte to the transmit buffer.			

15.5.11 I2Cn_IF - Interrupt Flag Register

Offset	Bit Position																															
0x028	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
Reset																0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Access																R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Name																SSTOP	CLTO	BITO	RXUF	TXOF	BUSHOLD	BUSERR	ARBLOST	MSTOP	NACK	ACK	RXDATAV	TXBL	TXC	ADDR	RSTART	START

Bit	Name	Reset	Access	Description
31:17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
16	SSTOP	0	R	**Slave STOP condition Interrupt Flag**
	Set when a STOP condition has been received. Will be set regardless of the EFM32 being involved in the transaction or not.			
15	CLTO	0	R	**Clock Low Timeout Interrupt Flag**

Figure 16.13. USART ISO 7816 Data Frame With Error



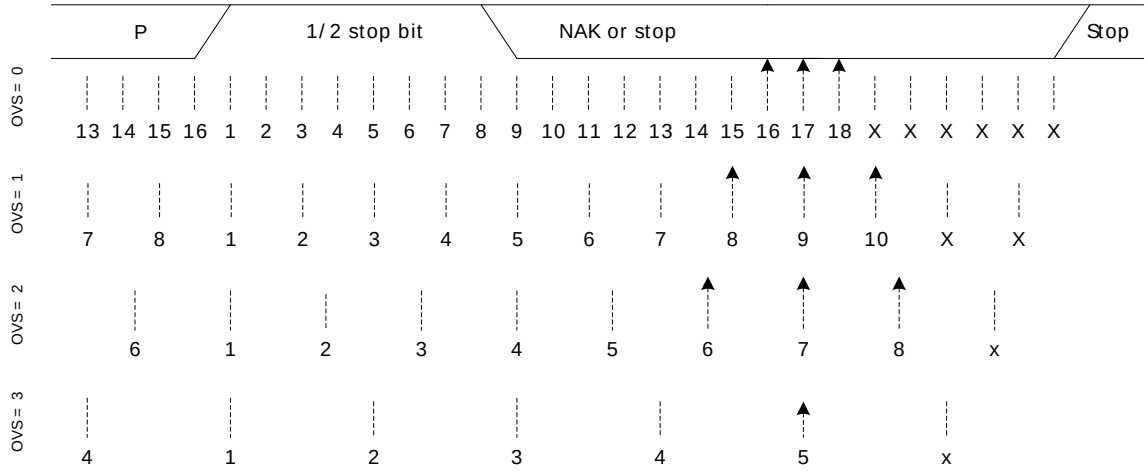
On a parity error, the NAK is generated by hardware. The NAK generated by the receiver is sampled as the stop-bit of the frame. Because of this, parity errors when in SmartCard mode are reported with both a parity error and a framing error.

When transmitting a T0 frame, the USART receiver on the transmitting side samples position 16, 17 and 18 in the stop-bit to detect the error signal when in 16x oversampling mode as shown in Figure 16.14 (p. 223) . Sampling at this location places the stop-bit sample in the middle of the bit-period used for the error signal (NAK).

If a NAK is transmitted by the receiver, it will thus appear as a framing error at the transmitter, and the FERR interrupt flag in USARTn_IF will be set. If SCRETRANS USARTn_CTRL is set, the transmitter will automatically retransmit a NACK'ed frame. The transmitter will retransmit the frame until it is ACK'ed by the receiver. This only works when the number of databits in a frame is configured to 8.

Set SKIPPERRF in USARTn_CTRL to make the receiver discard frames with parity errors. The PERR interrupt flag in USARTn_IF is set when a frame is discarded because of a parity error.

Figure 16.14. USART SmartCard Stop Bit Sampling



For communication with a SmartCard, a clock signal needs to be generated for the card. This clock output can be generated using one of the timers. See the ISO 7816 specification for more info on this clock signal.

SmartCard T1 mode is also supported. The T1 frame format used is the same as the asynchronous frame format with parity bit enabled and one stop bit. The USART must then be configured to operate in asynchronous half duplex mode.

16.3.3 Synchronous Operation

Most of the features in asynchronous mode are available in synchronous mode. Multi-processor mode can be enabled for 9-bit frames, loopback is available and collision detection can be performed.

Bit	Name	Reset	Access	Description

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected
1	PRSCH1	PRS Channel 1 selected
2	PRSCH2	PRS Channel 2 selected
3	PRSCH3	PRS Channel 3 selected
4	PRSCH4	PRS Channel 4 selected
5	PRSCH5	PRS Channel 5 selected
6	PRSCH6	PRS Channel 6 selected
7	PRSCH7	PRS Channel 7 selected

3 IRFILT 0 RW **IrDA RX Filter**

Set to enable filter on IrDA demodulator.

Value	Description
0	No filter enabled
1	Filter enabled. IrDA pulse must be high for at least 4 consecutive clock cycles to be detected

2:1 IRPW 0x0 RW **IrDA TX Pulse Width**

Configure the pulse width generated by the IrDA modulator as a fraction of the configured USART bit period.

Value	Mode	Description
0	ONE	IrDA pulse width is 1/16 for OVS=0 and 1/8 for OVS=1
1	TWO	IrDA pulse width is 2/16 for OVS=0 and 2/8 for OVS=1
2	THREE	IrDA pulse width is 3/16 for OVS=0 and 3/8 for OVS=1
3	FOUR	IrDA pulse width is 4/16 for OVS=0 and 4/8 for OVS=1

0 IREN 0 RW **Enable IrDA Module**

Enable IrDA module and rout USART signals through it.

16.5.22 USARTn_ROUTE - I/O Routing Register

Offset	Bit Position																													
0x054	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
Reset																							0x0						0	0	0	0
Access																							RW						RW	RW	RW	RW
Name																							LOCATION						CLKPEN	CSPEN	TXPEN	RXPEN

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		

9:8 LOCATION 0x0 RW **I/O Location**

Decides the location of the USART I/O pins.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3

7:4 Reserved *To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*

3 CLKPEN 0 RW **CLK Pin Enable**

When set, the CLK pin of the USART is enabled.

Value	Description
0	The USn_CLK pin is disabled

Bit	Name	Reset	Access	Description

Value	Description
1	The USn_CLK pin is enabled

Bit	Name	Reset	Access	Description
2	CSPEN	0	RW	**CS Pin Enable**

When set, the CS pin of the USART is enabled.

Value	Description
0	The USn_CS pin is disabled
1	The USn_CS pin is enabled

Bit	Name	Reset	Access	Description
1	TXPEN	0	RW	**TX Pin Enable**

When set, the TX/MOSI pin of the USART is enabled

Value	Description
0	The U(S)n_TX (MOSI) pin is disabled
1	The U(S)n_TX (MOSI) pin is enabled

Bit	Name	Reset	Access	Description
0	RXPEN	0	RW	**RX Pin Enable**

When set, the RX/MISO pin of the USART is enabled.

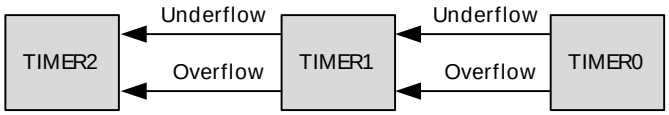
Value	Description
0	The U(S)n_RX (MISO) pin is disabled
1	The U(S)n_RX (MISO) pin is enabled

must not have a higher frequency than $f_{HFPERCLK}/3$ when running from a pin input or a PRS input with FILT enabled in TIMERn_CCx_CTRL. When running from PRS without FILT, the frequency can be as high as $f_{HFPERCLK}$. Note that when clocking the Timer from the same pulse that triggers a start (through RISEA/FALLA in TIMERn_CTRL), the starting pulse will not update the Counter Value.

19.3.1.3.3 Underflow/Overflow from Neighboring Timer

All Timers are linked together (see Figure 19.4 (p. 278)), allowing timers to count on overflow/underflow from the lower numbered neighbouring timers to form a 32-bit or 48-bit timer. Note that all timers must be set to same count direction and less significant timer(s) can only be set to count up or down.

Figure 19.4. TIMER Connections



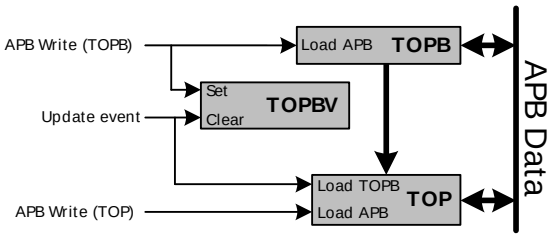
19.3.1.4 One-Shot Mode

By default, the counter counts continuously until it is stopped. If the OSMEN bit is set in the TIMERn_CTRL register, however, the counter is disabled by hardware on the first *update event*. Note that when the counter is running with CC1 as clock source (0b01 in CLKSEL in TIMERn_CTRL) and OSMEN is set, a CC1 capture event will not take place on the *update event* (CC1 rising edge) that stops the Timer.

19.3.1.5 Top Value Buffer

The TIMERn_TOP register can be altered either by writing it directly or by writing to the TIMER_TOPB (buffer) register. When writing to the buffer register the TIMERn_TOPB register will be written to TIMERn_TOP on the next update event. Buffering ensures that the TOP value is not set below the actual count value. The TOPBV flag in TIMERn_STATUS indicates whether the TIMERn_TOPB register contains data that have not yet been written to the TIMERn_TOP register (see Figure 19.5 (p. 278) .

Figure 19.5. TIMER TOP Value Update Functionality



19.3.1.6 Quadrature Decoder

Quadrature Decoding mode is used to track motion and determine both rotation direction and position. The Quadrature Decoder uses two input channels that are 90 degrees out of phase (see Figure 19.6 (p. 279)).

LETIMERn_CNT becomes equal to their value. When LETIMERn_CNT becomes equal to the value of LETIMERn_COMP0, the interrupt flag COMP0 in LETIMERn_IF is set, and when LETIMERn_CNT becomes equal to the value of LETIMERn_COMP1, the interrupt flag COMP1 in LETIMERn_IF is set.

21.3.3 Top Value

If COMP0TOP in LETIMERn_CTRL is set, the value of LETIMERn_COMP0 acts as the top value of the timer, and LETIMERn_COMP0 is loaded into LETIMERn_CNT on timer underflow. Else, the timer wraps around to 0xFFFF. The underflow interrupt flag UF in LETIMERn_IF is set when the timer reaches zero.

21.3.3.1 Buffered Top Value

If BUFTOP in LETIMERn_CTRL is set, the value of LETIMERn_COMP0 is buffered by LETIMERn_COMP1. In this mode, the value of LETIMERn_COMP1 is loaded into LETIMERn_COMP0 every time LETIMERn_REP0 is about to decrement to 0. This can for instance be used in conjunction with the buffered repeat mode to generate continually changing output waveforms.

Write operations to LETIMERn_COMP0 have priority over buffer loads.

21.3.3.2 Repeat Modes

By default, the timer wraps around to the top value or 0xFFFF on each underflow, and continues counting. The repeat counters can be used to get more control of the operation of the timer, including defining the number of times the counter should wrap around. Four different repeat modes are available, see Table 21.1 (p. 321) .

Table 21.1. LETIMER Repeat Modes

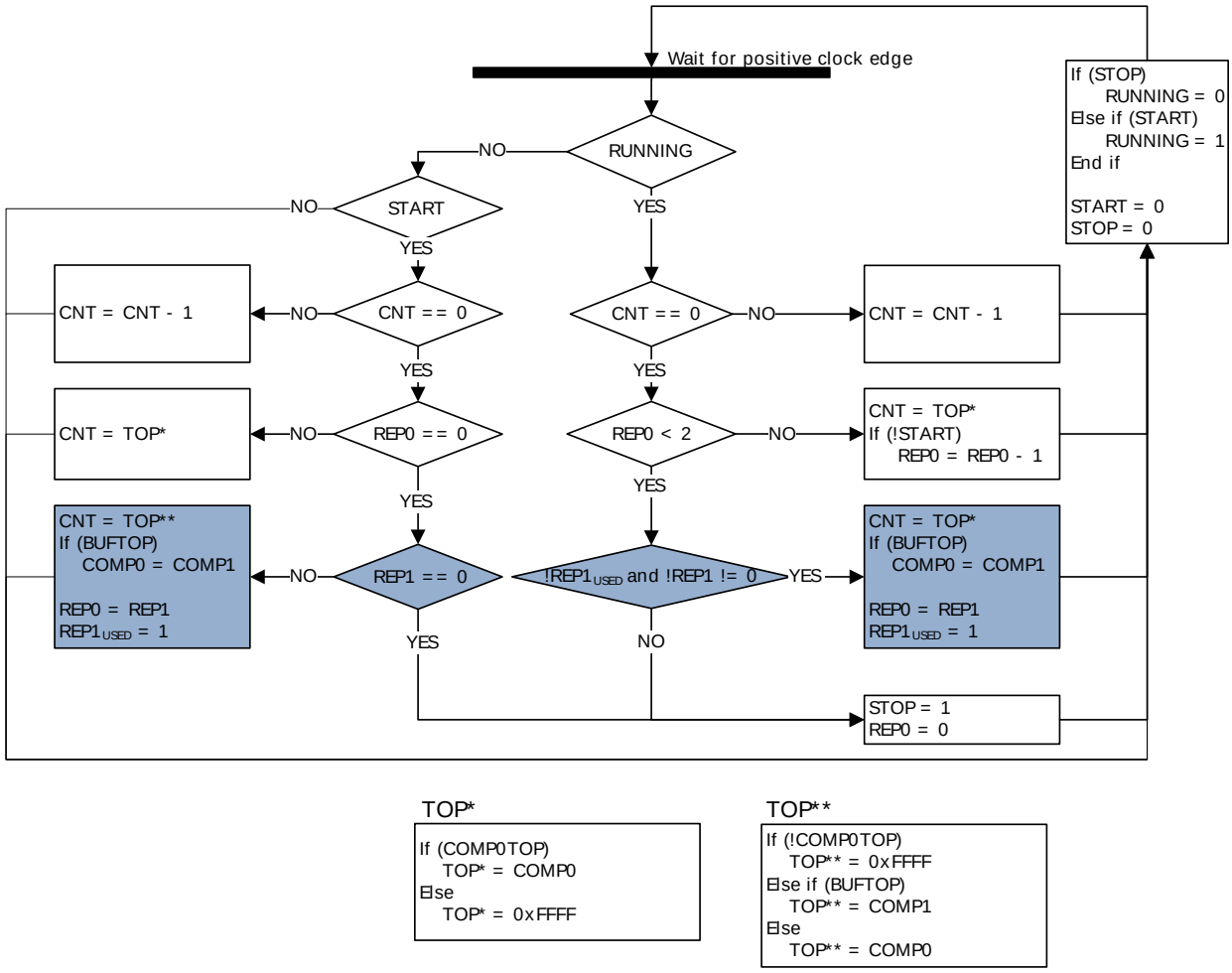
REPMODE	Mode	Description
00	Free	The timer runs until it is stopped
01	One-shot	The timer runs as long as LETIMERn_REP0 != 0. LETIMERn_REP0 is decremented at each timer underflow.
10	Buffered	The timer runs as long as LETIMERn_REP0 != 0. LETIMERn_REP0 is decremented on each timer underflow. If LETIMERn_REP1 has been written, it is loaded into LETIMERn_REP0 when LETIMERn_REP0 is about to be decremented to 0.
11	Double	The timer runs as long as LETIMERn_REP0 != 0 or LETIMERn_REP1 != 0. Both LETIMERn_REP0 and LETIMERn_REP1 are decremented at each timer underflow.

The interrupt flags REP0 and REP1 in LETIMERn_IF are set whenever LETIMERn_REP0 or LETIMERn_REP1 are decremented to 0 respectively. REP0 is also set when the value of LETIMERn_REP1 is loaded into LETIMERn_REP0 in buffered mode.

21.3.3.2.1 Free Mode

In the free running mode, the LETIMER acts as a regular timer, and the repeat counter is disabled. When started, the timer runs until it is stopped using the STOP command bit in LETIMERn_CMD. A state machine for this mode is shown in Figure 21.2 (p. 322) .

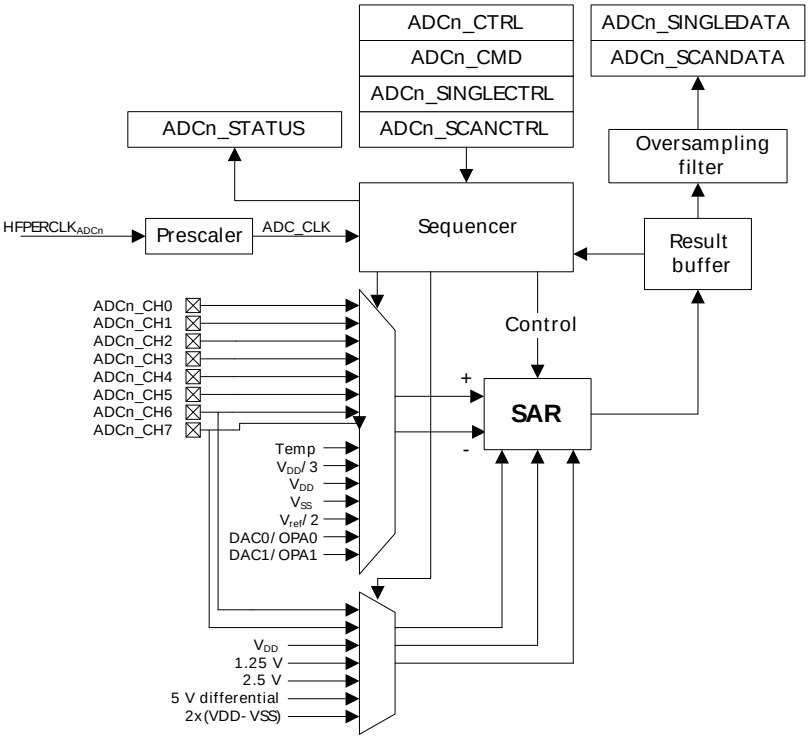
Figure 21.4. LETIMER Buffered Repeat State Machine



21.3.3.2.4 Double Mode

The Double repeat mode works much like the one-shot repeat mode. The difference is that, where the one-shot mode counts as long as LETIMERn_REP0 is larger than 0, the double mode counts as long as either LETIMERn_REP0 or LETIMERn_REP1 is larger than 0. As an example, say LETIMERn_REP0 is 3 and LETIMERn_REP1 is 10 when the timer is started. If no further interaction is done with the timer, LETIMERn_REP0 will now be decremented 3 times, and LETIMERn_REP1 will be decremented 10 times. The timer counts a total of 10 times, and LETIMERn_REP0 is 0 after the first three timer underflows and stays at 0. LETIMERn_REP0 and LETIMERn_REP1 can be written at any time. After a write to either of these, the timer is guaranteed to underflow at least the written number of times if the timer is running. Use the Double repeat mode to generate output on both the LETIMER outputs at the same time. The state machine for this repeat mode can be seen in Figure 21.5 (p. 325) .

Figure 25.1. ADC Overview



25.3.1 Clock Selection

The ADC has an internal prescaler (PRESC bits in ADCn_CTRL) which can divide the peripheral clock (HFPERCLK) by any factor between 1 and 128. Note that the resulting ADC_CLK should not be set to a higher frequency than 13 MHz and not lower than 32 kHz.

25.3.2 Conversions

A conversion consists of two phases. The input is sampled in the acquisition phase before it is converted to digital representation during the approximation phase. The acquisition time can be configured independently for scan and single conversions (see Section 25.3.7 (p. 379)) by setting AT in ADCn_SINGLECTRL/ADCn_SCANCTRL. The acquisition times can be set to any integer power of 2 from 1 to 256 ADC_CLK cycles.

Note

For high impedance sources the acquisition time should be adjusted to allow enough time for the internal sample capacitor to fully charge. The minimum acquisition time for the internal temperature sensor and $V_{dd}/3$ is given in the electrical characteristics for the device.

The analog to digital converter core uses one clock cycle per output bit in the approximation phase.

ADC Total Conversion Time (in ADC_CLK cycles) Per Output

$$T_{conv} = (T_A + N) \times OSR \quad (25.1)$$

T_A equals the number of acquisition cycles and N is the resolution. OSR is the oversampling ratio (see Section 25.3.7.7 (p. 381)). The minimum conversion time is 7 ADC_CYCLES with 6 bit resolution and 13 ADC_CYCLES with 12 bit resolution. The maximum conversion time is 1097728 ADC_CYCLES with the longest acquisition time, 12 bit resolution and highest oversampling rate.

Corrected DAC clock prescaling equation (Equation 26.1 (p. 399)).

30.7 Revision 0.82

November 20th, 2009

Description of LFXOSEL and LFRCOSEL bits of CMU_STATUS register corrected.

Updated description of EM4 sequence in Table 10.2 (p. 90) .

Updated documentation of WORDTIMEOUT and WDATAREADY in MSC_STATUS.

30.8 Revision 0.81

November 13th, 2009

Note added to Section 7.3.5 (p. 33) .

Note added to Section 7.3.5 (p. 33) .

Internal reference added to Section 5.6 (p. 23) .

DMA_CHx_CTRL register description updated.

Reference to synchronous pin interrupts removed from Chapter 10 (p. 86) .

ACMP wakeup triggering updated in Chapter 10 (p. 86) .

Internal reference added to note in Section 11.3.1.2 (p. 98) .

Figure 11.4 (p. 101) and Figure 11.5 (p. 101) added.

Section 15.3.7 (p. 180) updated.

Note added in Section 18.3.3 (p. 251) .

Section 25.3.6 (p. 378) added and ADCn_BIASPROG register added.

Section 26.3.3 (p. 399) added and DACn_BIASPROG register added.

Section 26.3.8 (p. 401) updated.

Glitch suppression filter added to Figure 28.1 (p. 426) , Figure 28.2 (p. 427) and Figure 28.4 (p. 428) .

Section 29.3.5 (p. 461) and Section 29.3.6 (p. 462) updated.

LCD_DISPCTRL register updated.

Added PRS example in Section 13.3.4 (p. 131) .

Split CCPEN and CDTIPEN bits in TIMERn_ROUTE into CCxPEN and CDTIxPEN bits.

Description and enumeration of EMVREG in EMU_CTRL updated.

30.9 Revision 0.80

October 19th, 2009

Initial preliminary revision.