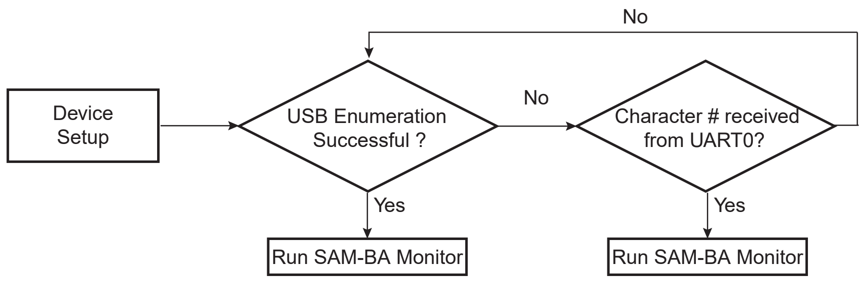
What is "Embedded - Microcontrollers"?

"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	Active
Core Processor	ARM® Cortex®-M7
Core Size	32-Bit Single-Core
Speed	300MHz
Connectivity	EBI/EMI, I²C, IrDA, LINbus, MMC/SD/SDIO, QSPI, SPI, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, I²S, POR, PWM, WDT
Number of I/O	114
Program Memory Size	1MB (1M x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	384K x 8
Voltage - Supply (Vcc/Vdd)	1.08V ~ 3.6V
Data Converters	A/D 24x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	144-LFBGA
Supplier Device Package	144-LFBGA (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsams70q20a-cn

Figure 17-1. Boot Program Algorithm Flow Diagram


Device Setup
USB Enumeration Successful ?
No
Character # received from UART0?
No
Yes
Yes
Run SAM-BA Monitor
Run SAM-BA Monitor

The SAM-BA boot program looks for a source clock, either from the embedded main oscillator with external crystal (main oscillator enabled) or from a supported frequency signal applied to the XIN pin (Main oscillator in bypass mode).

If a clock is supplied by one of the two sources, the boot program checks that the frequency is one of the supported external frequencies. If the frequency is supported, USB activation is allowed. If no clock is supplied, or if a clock is supplied but the frequency is not a supported external frequency, the internal 12 MHz RC oscillator is used as the main clock. In this case, the USB is not activated due to the frequency drift of the 12 MHz RC oscillator.

17.5 Device Initialization

Initialization by the boot program follows the steps described below:

Stack setup.

1. Embedded Flash Controller setup.
2. External clock (crystal or external clock on XIN) detection.
3. External crystal or clock with supported frequency supplied.
   a. If yes, USB activation is allowed.

   b. If no, USB activation is not allowed. The internal 12 MHz RC oscillator is used.
4. Master clock switch to main oscillator.
5. C variable initialization.
6. PLLA setup: PLLA is initialized to generate a 48 MHz clock.
7. Watchdog disable.
8. Initialization of UART0 (115200 bauds, 8, N, 1).
9. Initialization of the USB Device Port (only if USB activation is allowed; see Step 4.).
10. Wait for one of the following events:
    a. Check if USB device enumeration has occurred.

    b. Check if characters have been received in UART0.
11. Jump to SAM-BA Monitor (refer to 17.6 SAM-BA Monitor)

17.6 SAM-BA Monitor

Once the communication interface is identified, the monitor runs in an infinite loop, waiting for different commands, as shown in the following table.

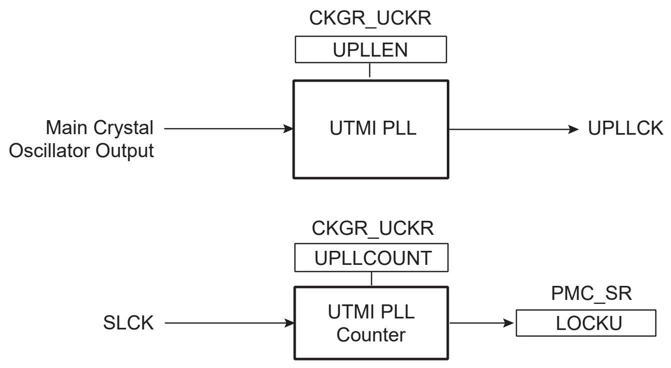
30.7 UTMI PLL Clock

The source of the UTMI PLL (UPLL) is the Main Crystal oscillator. The UPLL provides the UTMI PLL Clock (UPLLCK) and UPLLCKDIV clock signals.

The UPLL has two possible multiplying factors: x40 and x30. To generate UPLLCK at 480 MHz (typical USB case), this leads to two possible crystal oscillator frequencies: 12 or 16 MHz. The crystal oscillator frequency (12 or 16 MHz) must be programmed in UTMI_CKTRIM.FREQ prior to enabling the UPLL.

When the UPLL is enabled by writing a ‘1’ to bit UPLLEN in the UTMI Clock Register (CKGR_UCKR), the LOCKU bit in PMC_SR is automatically cleared. The values written in the PLLCOUNT field in CKGR_UCKR are loaded in the UTMI PLL counter. The UTMI PLL counter then decrements at the speed of SLCK divided by 8 until it reaches ‘0’. At this time, the LOCKU bit is set in PMC_SR and can trigger an interrupt to the processor. The user has to load the number of SLCK cycles required to cover the UTMI PLL transient time into the PLLCOUNT field.

Figure 30-5. UTMI PLL Block Diagram



36.9.9 XDMAC Global Channel Disable Register

Name:	XDMAC_GD
Offset:	0x20
Reset:	–
Property:	Write-only

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	DI23	DI22	DI21	DI20	DI19	DI18	DI17	DI16
Access	W	W	W	W	W	W	W	W
Reset	–	–	–	–	–	–	–	–

Bit	15	14	13	12	11	10	9	8
	DI15	DI14	DI13	DI12	DI11	DI10	DI9	DI8
Access	W	W	W	W	W	W	W	W
Reset	–	–	–	–	–	–	–	–

Bit	7	6	5	4	3	2	1	0
	DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0
Access	W	W	W	W	W	W	W	W
Reset	–	–	–	–	–	–	–	–

Bits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23 – DI XDMAC Channel x Disable

Value	Description
`0`	This bit has no effect.
`1`	Disables channel x.

38.8.40 GMAC Frames Transmitted

Name:	GMAC_FT
Offset:	0x108
Reset:	0x00000000
Property:	Read-only

Bit	31	30	29	28	27	26	25	24
	FTX[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	FTX[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FTX[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FTX[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – FTX[31:0] Frames Transmitted without Error
Frames transmitted without error. This register counts the number of frames successfully transmitted, i.e., no underrun and not too many retries. Excludes pause frames.

38.8.53 GMAC Multiple Collision Frames Register

Name:	GMAC_MCF
Offset:	0x13C
Reset:	0x00000000
Property:	-

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
							MCOL[17:16]	
Access							R	R
Reset							0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
	MCOL[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	MCOL[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 17:0 – MCOL[17:0] Multiple Collision
This register counts the number of frames experiencing between two and fifteen collisions prior to being successfully transmitted, i.e., no underrun and not too many retries.

38.8.87 GMAC 1588 Timer Adjust Register

Name:	GMAC_TA
Offset:	0x1D8
Reset:	0x00000000
Property:	-

Bit	31	30	29	28	27	26	25	24
	ADJ	▨	ITDT[29:24]					
Access	W		W	W	W	W	W	W
Reset	0		0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ITDT[23:16]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ITDT[15:8]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ITDT[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit 31 – ADJ Adjust 1588 Timer
Write as '1' to subtract from the 1588 timer. Write as '0' to add to it.

Bits 29:0 – ITDT[29:0] Increment/Decrement
The number of nanoseconds to increment or decrement the IEEE 1588 Timer Nanoseconds Register. If necessary, the IEEE 1588 Seconds Register will be incremented or decremented.

Offset	Name	Bit Pos.								
0x0570	USBHS_HSTPIPICR4 (INTPIPES)	7:0	SHORTPACKETIC	RXSTALLDIC	OVERFIC	NAKEDIC		UNDERFIC	TXOUTIC	RXINIC
		15:8								
		23:16								
		31:24								
0x0570	USBHS_HSTPIPICR4 (ISOPIPES)	7:0	SHORTPACKETIC	CRCERRIC	OVERFIC	NAKEDIC		UNDERFIC	TXOUTIC	RXINIC
		15:8								
		23:16								
		31:24								
0x0574	USBHS_HSTPIPICR5	7:0	SHORTPACKETIC	RXSTALLDIC	OVERFIC	NAKEDIC		TXSTPIC	TXOUTIC	RXINIC
		15:8								
		23:16								
		31:24								
0x0574	USBHS_HSTPIPICR5 (INTPIPES)	7:0	SHORTPACKETIC	RXSTALLDIC	OVERFIC	NAKEDIC		UNDERFIC	TXOUTIC	RXINIC
		15:8								
		23:16								
		31:24								
0x0574	USBHS_HSTPIPICR5 (ISOPIPES)	7:0	SHORTPACKETIC	CRCERRIC	OVERFIC	NAKEDIC		UNDERFIC	TXOUTIC	RXINIC
		15:8								
		23:16								
		31:24								
0x0578	USBHS_HSTPIPICR6	7:0	SHORTPACKETIC	RXSTALLDIC	OVERFIC	NAKEDIC		TXSTPIC	TXOUTIC	RXINIC
		15:8								
		23:16								
		31:24								
0x0578	USBHS_HSTPIPICR6 (INTPIPES)	7:0	SHORTPACKETIC	RXSTALLDIC	OVERFIC	NAKEDIC		UNDERFIC	TXOUTIC	RXINIC
		15:8								
		23:16								
		31:24								
0x0578	USBHS_HSTPIPICR6 (ISOPIPES)	7:0	SHORTPACKETIC	CRCERRIC	OVERFIC	NAKEDIC		UNDERFIC	TXOUTIC	RXINIC
		15:8								
		23:16								
		31:24								
0x057C	USBHS_HSTPIPICR7	7:0	SHORTPACKETIC	RXSTALLDIC	OVERFIC	NAKEDIC		TXSTPIC	TXOUTIC	RXINIC
		15:8								
		23:16								
		31:24								

39.6.12 Device Endpoint Register

Name: USBHS_DEVEPT
Offset: 0x001C
Reset: 0x00000000
Property: Read/Write

Bit	31	30	29	28	27	26	25	24
							EPRST9	EPRST8
Access								
Reset							0	0

Bit	23	22	21	20	19	18	17	16
	EPRST7	EPRST6	EPRST5	EPRST4	EPRST3	EPRST2	EPRST1	EPRST0
Access								
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
							EPEN9	EPEN8
Access								
Reset							0	0

Bit	7	6	5	4	3	2	1	0
	EPEN7	EPEN6	EPEN5	EPEN4	EPEN3	EPEN2	EPEN1	EPEN0
Access								
Reset	0	0	0	0	0	0	0	0

Bits 16, 17, 18, 19, 20, 21, 22, 23, 24, 25 – EPRST Endpoint x Reset
The whole endpoint mechanism (FIFO counter, reception, transmission, etc.) is reset apart from the Data Toggle Sequence field (USBHS_DEVEPTISRx.DTSEQ), which can be cleared by setting the USBHS_DEVEPTIMRx.RSTDT bit (by writing a one to the USBHS_DEVEPTIERx.RSTDTS bit).

The endpoint configuration remains active and the endpoint is still enabled.

This bit is cleared upon receiving a USB reset.

Value	Description
`0`	Completes the reset operation and starts using the FIFO.
`1`	Resets the endpoint x FIFO prior to any other operation, upon hardware reset or when a USB bus reset has been received. This resets the endpoint x registers (USBHS_DEVEPTCFGx, USBHS_DEVEPTISRx, USBHS_DEVEPTIMRx) but not the endpoint configuration (USBHS_DEVEPTCFGx.ALLOC, USBHS_DEVEPTCFGx.EPBK, USBHS_DEVEPTCFGx.EPSIZE, USBHS_DEVEPTCFGx.EPDIR, USBHS_DEVEPTCFGx.EPTYPE).

Bits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 – EPEN Endpoint x Enable

Value	Description
`0`	Endpoint x is disabled, forcing the endpoint x state to inactive (no answer to USB requests) and resetting the endpoint x registers (USBHS_DEVEPTCFGx, USBHS_DEVEPTISRx,

Bit 16 – RWALL Read/Write Allowed
For an OUT pipe, this bit is set when the current bank is not full, i.e., the software can write further data into the FIFO.

For an IN pipe, this bit is set when the current bank is not empty, i.e., the software can read further data from the FIFO.

This bit is cleared otherwise.

This bit is also cleared when the RXSTALLDI or the PERRI bit = 1.

Bits 15:14 – CURRBK[1:0] Current Bank
For non-control pipe, this field indicates the number of the current bank.

This field may be updated 1 clock cycle after the RWALL bit changes, so the user should not poll it as an interrupt bit.

Value	Name	Description
0	BANK0	Current bank is bank0
1	BANK1	Current bank is bank1
2	BANK2	Current bank is bank2
3	Reserved	

Bits 13:12 – NBUSYBK[1:0] Number of Busy Banks
This field indicates the number of busy banks.

For an OUT pipe, this field indicates the number of busy banks, filled by the user, ready for OUT transfer. When all banks are busy, this triggers a PEP_x interrupt if USBHS_HSTPIPIMRx.NBUSYBKE = 1.

For an IN pipe, this field indicates the number of busy banks filled by IN transaction from the Device. When all banks are free, this triggers a PEP_x interrupt if USBHS_HSTPIPIMRx.NBUSYBKE = 1.

Value	Name	Description
0	0_BUSY	0 busy bank (all banks free)
1	1_BUSY	1 busy bank
2	2_BUSY	2 busy banks
3	3_BUSY	3 busy banks

Bits 9:8 – DTSEQ[1:0] Data Toggle Sequence
This field indicates the data PID of the current bank.

For an OUT pipe, this field indicates the data toggle of the next packet that is to be sent.

For an IN pipe, this field indicates the data toggle of the received packet stored in the current bank.

Value	Name	Description
0	DATA0	Data0 toggle sequence
1	DATA1	Data1 toggle sequence
2	Reserved	
3	Reserved	

Bit 7 – SHORTPACKETI Short Packet Interrupt

Arbitration starts as soon as two or more masters place information on the bus at the same time, and stops (arbitration is lost) for the master that intends to send a logical one while the other master sends a logical zero.

As soon as arbitration is lost by a master, it stops sending data and listens to the bus in order to detect a stop. When the stop is detected, the master that has lost arbitration may put its data on the bus by respecting arbitration.

Arbitration is illustrated in Arbitration Cases.

43.6.4.2 Different Multimaster Modes

Two Multimaster modes may be distinguished:

1. The TWIHS is considered as a master only and is never addressed.
2. The TWIHS may be either a master or a slave and may be addressed.

Note: Arbitration in supported in both Multimaster modes.

43.6.4.2.1 TWIHS as Master Only

In this mode, the TWIHS is considered as a master only (MSEN is always at one) and must be driven like a master with the ARBLST (Arbitration Lost) flag in addition.

If arbitration is lost (ARBLST = 1), the user must reinitiate the data transfer.

If starting a transfer (ex.: DADR + START + W + Write in THR) and if the bus is busy, the TWIHS automatically waits for a STOP condition on the bus to initiate the transfer (see User Sends Data While the Bus is Busy).

Note: The state of the bus (busy or free) is not indicated in the user interface.

43.6.4.2.2 TWIHS as Master or Slave

The automatic reversal from master to slave is not supported in case of a lost arbitration.

Then, in the case where TWIHS may be either a master or a slave, the user must manage the pseudo Multimaster mode described in the steps below:

1. Program the TWIHS in Slave mode (SADR + MSDIS + SVEN) and perform a slave access (if TWIHS is addressed).
2. If the TWIHS has to be set in Master mode, wait until TXCOMP flag is at 1.
3. Program the Master mode (DADR + SVDIS + MSEN) and start the transfer (ex: START + Write in THR).
4. As soon as the Master mode is enabled, the TWIHS scans the bus in order to detect if it is busy or free. When the bus is considered free, the TWIHS initiates the transfer.
5. As soon as the transfer is initiated and until a STOP condition is sent, the arbitration becomes relevant and the user must monitor the ARBLST flag.
6. If the arbitration is lost (ARBLST is set to 1), the user must program the TWIHS in Slave mode in case the master that won the arbitration needs to access the TWIHS.
7. If the TWIHS has to be set in Slave mode, wait until the TXCOMP flag is at 1 and then program the Slave mode.

Note: If the arbitration is lost and the TWIHS is addressed, the TWIHS does not acknowledge, even if it is programmed in Slave mode as soon as ARBLST is set to 1. Then the master must repeat SADR.

43.7.6 TWIHS Status Register

Name:	TWIHS_SR
Offset:	0x20
Reset:	0x03000009
Property:	Read-only

Bit	31	30	29	28	27	26	25	24
							SDA	SCL
Access							R	R
Reset							1	1

Bit	23	22	21	20	19	18	17	16
			SMBHHM	SMBDAM	PECERR	TOUT		MCACK
Access			R	R	R	R		R
Reset			0	0	0	0		0

Bit	15	14	13	12	11	10	9	8
					EOSACC	SCLWS	ARBLST	NACK
Access					R	R	R	R
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	UNRE	OVRE	GACC	SVACC	SVREAD	TXRDY	RXRDY	TXCOMP
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	1	0	0	1

Bit 25 – SDA SDA Line Value

Value	Description
0	SDA line sampled value is ‘0’.
1	SDA line sampled value is ‘1’.

Bit 24 – SCL SCL Line Value

Value	Description
0	SCL line sampled value is ‘0’.
1	SCL line sampled value is ‘1.’

Bit 21 – SMBHHM SMBus Host Header Address Match (cleared on read)

Value	Description
0	No SMBus Host Header Address received since the last read of TWIHS_SR.
1	An SMBus Host Header Address was received since the last read of TWIHS_SR.

Bit 20 – SMBDAM SMBus Default Address Match (cleared on read)

Value	Description
0	No SMBus Default Address received since the last read of TWIHS_SR.
1	An SMBus Default Address was received since the last read of TWIHS_SR.

whichever speeds a particular Device supports. All MediaLB Devices must support the rules for synchronization to MediaLB.

For MediaLB Controllers, all System commands are optional, including support for dynamic system configuration and DeviceAddresses.

For MediaLB Devices, support for all transport methods is optional. If a MediaLB Device supports a particular transport method, it must fully support it including all Command bytes and RxStatus responses associated with that transport method. For asynchronous and control methods, the Protocol error responses can be expanded for additional error checking, based on specific implementations. Any extra error checking that causes a Protocol error to be transmitted must be listed in the Device documentation.

For MediaLB Devices, support for System responses and dynamic configuration are optional. If dynamic configuration is supported, it must comply with the specifications listed in this document.

All MediaLB Devices must specify clearly in documentation what MediaLB speeds, System commands, and transport methods they support. In addition, MediaLB Devices must clearly state the DeviceAddress as well as the Index and associated transport method used in configuring the ChannelAddress.

48.6.3 Internal Flow Description

The internal functional blocks of the MLB include:

- MediaLB Block (MLB PHY) - Implements the physical and link-layer requirements of a MediaLB 3-pin interface. Serial-to-parallel and parallel-to-serial data transformations are implemented, as well as MediaLB frame synchronization.
- Host Bus Interface Block (HBI) - Provides 16-bit parallel slave access to all MOST channels and data types for the external Host Controller (HC). The HBI supports up to 64 independent channels with a minimum access latency of 40 ns per word and a maximum bandwidth of 400 Mbps.
- Routing Fabric Block (RF) - Manages the flow of data between the MediaLB block and the HBI block, implementing a bus arbiter and multiplexing logic to the Channel Table RAM (CTR) and the Data Buffer RAM (DBR).
- Memory Interface Block (MIF) - Implements a bridge between the I/O bus and the customer-implemented RAMs (i.e. Channel Table and Data Buffer).
- Interrupt Interface Block (INTIF) - Sends notifications to HBI that there are changes to the channel descriptors.
- Clocks, Power, and Reset Block (CPR) - Implements clock and reset multiplexing and synchronization.
- AHB Block (AHB) - Implements a bus bridge between the AHB master and the HBI slave interfaces.
- APB Block (APB) - Implements a bus bridge that translates the two-cycle APB interface signals to the single-cycle I/O interface signals.

48.6.3.1 MediaLB Block

The Media Local Bus (MediaLB) block supports a MediaLB 3-pin interface that provides real-time access to all network data types including streaming, packet, control, and isochronous data.

The MediaLB interface supports the MediaLB protocol for single-ended 3-pin mode, with a maximum data rate of 1024xFs (49.152 MHz at Fs=48 kHz).

MediaLB Channel Address to Logical Channel Mapping

The MediaLB channel addresses are mapped to the logical channels as follows:

Table 48-15. Synchronous CDT Entry Field Definitions

Field	Description	Details	Accessibility
BA	Buffer Base Address	- BA can start at any byte in the 16k DBR	r,w
BD	Buffer Depth	- BD = size of buffer in bytes - 1 - Buffer end address = BA + BD - BD = 4 x m x bpf - 1, where: m = frames per sub-buffer (for MFE = 0, m = 1) bpf = bytes per frame.	r,w
RPTR	Read Pointer	- Software initializes to zero, hardware updates - Counts the read address offset within a buffer - DMA read address = BA + RPTR	r,w,u(1)
WPTR	Write Pointer	- Software initializes to zero, hardware updates - Counts the write address offset within a buffer - DMA write address = BA + WPTR	r,w,u (1)
RSBC	Read Sub-buffer Counter	- Software initializes to zero, hardware updates - Counts the read sub-buffer offset - DMA uses for pointer management	r,w,u (1)
WSBC	Write Sub-buffer Counter	- Software initializes to zero, hardware updates - Counts the write sub-buffer offset - DMA uses for pointer management	r,w,u (1)

RSTS	Read Status	- Software initializes to zero, hardware updates
		- RSTS states:(2)
		xxx0 = normal operation (no mute)
		xxx1 = normal operation (mute)
		xx0x = idle
			r,w,u (1)

WSTS	Write Status	- Software initializes to zero, hardware updates
		- WSTS states:(2)
		xxx0 = normal operation (no mute)
		xxx1 = normal operation (mute)
		xx0x = idle
		1xxx = command protocol error
			r,w,u (1)

Reserved	Reserved	- Software writes a zero to all reserved bits when the entry is initialized. The reserved bits are Read-only after initialization.	r,w,u (1)

Notes: 1. “u” means “Updated periodically by hardware”.

48.7.1 MediaLB Control 0 Register

Name:	MLB_MLBC0
Offset:	0x000
Reset:	0x00000000
Property:	Read/Write

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
							FCNT[2:1]	
Access								
Reset							0	0

Bit	15	14	13	12	11	10	9	8
	FCNT[0:0]	CTLRETRY		ASYRETRY				
Access								
Reset	0	0		0				

Bit	7	6	5	4	3	2	1	0
	MLBLK		ZERO	MLBCLK[2:0]				MLBEN
Access								
Reset	0		0	0	0	0		0

Bits 17:15 – FCNT[2:0] The number of frames per sub-buffer for synchronous channels

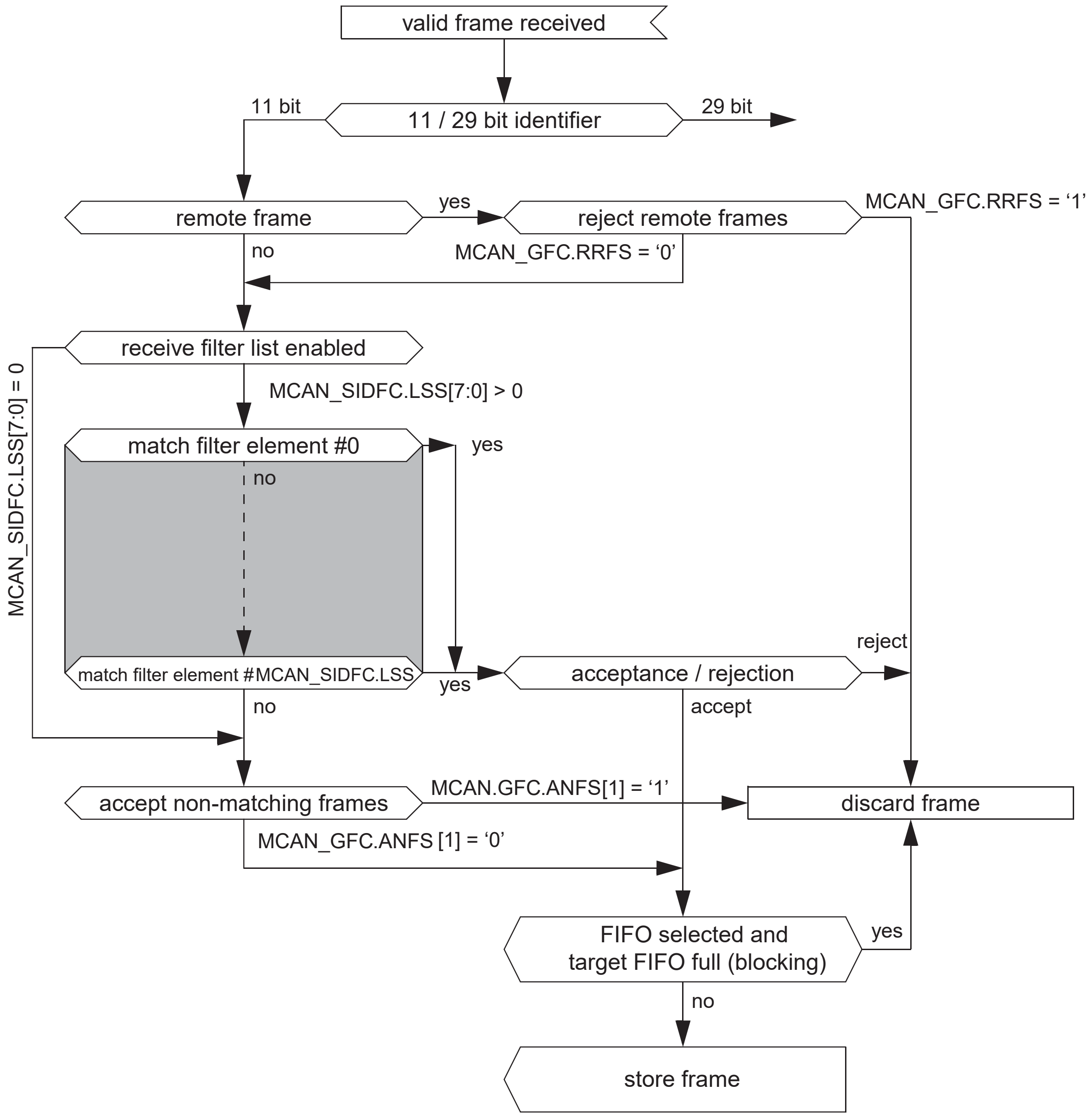
Value	Name	Description
0	1_FRAME	1 frame per sub-buffer (Operation is the same as Standard mode.)
1	2_FRAMES	2 frames per sub-buffer
2	4_FRAMES	4 frames per sub-buffer
3	8_FRAMES	8 frames per sub-buffer
4	16_FRAMES	16 frames per sub-buffer
5	32_FRAMES	32 frames per sub-buffer
6	64_FRAMES	64 frames per sub-buffer

Bit 14 – CTLRETRY Control Tx Packet Retry

Value	Description
0	A control packet that is flagged with a Break or ProtocolError by the receiver is skipped.
1	A control packet that is flagged with a Break or ProtocolError by the receiver is retransmitted.

Bit 12 – ASYRETRY Asynchronous Tx Packet Retry

Figure 49-5. Standard Message ID Filter Path



Extended Message ID Filtering

The figure below shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID Filter element is described in 49.5.7.6 Extended Message ID Filter Element.

Controlled by MCAN_GFC and MCAN_XIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

MCAN_XIDAM is ANDed with the received identifier before the filter list is executed.

49.6.45 MCAN Transmit Event FIFO Configuration

Name:	MCAN_TXEFC
Offset:	0xF0
Reset:	0x00000000
Property:	Read/Write

This register can only be written if the bits CCE and INIT are set in MCAN CC Control Register.

Bit	31	30	29	28	27	26	25	24
			EFWM[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
			EFS[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	EFSA[13:6]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	EFSA[5:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W		
Reset	0	0	0	0	0	0		

Bits 29:24 – EFWM[5:0] Event FIFO Watermark

Value	Description
`0`	Watermark interrupt disabled.
`1-32`	Level for Tx Event FIFO watermark interrupt (MCAN_IR.TEFW).
`>32`	Watermark interrupt disabled.

Bits 21:16 – EFS[5:0] Event FIFO Size
The Tx Event FIFO elements are indexed from 0 to EFS - 1.

Value	Description
`0`	Tx Event FIFO disabled.
`1-32`	Number of Tx Event FIFO elements.
`>32`	Values greater than 32 are interpreted as 32.

Bits 15:2 – EFSA[13:0] Event FIFO Start Address
Start address of Tx Event FIFO in Message RAM (32-bit word address, see Message RAM Configuration).

Write EFSA with the bits [15:2] of the 32-bit address.

50.7.3 TC Channel Mode Register: Waveform Mode

Name:	TC_CMRx
Offset:	0x04 + x*0x40 [x=0..2]
Reset:	0x00000000
Property:	Read/Write

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

Bit	31	30	29	28	27	26	25	24
	BSWTRG[1:0]		BEEVT[1:0]		BCPC[1:0]		BCPB[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ASWTRG[1:0]		AEEVT[1:0]		ACPC[1:0]		ACPA[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	WAVE	WAVSEL[1:0]		ENETRG	EEVT[1:0]		EEVTEDG[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CPCDIS	CPCSTOP	BURST[1:0]		CLKI	TCCLKS[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:30 – BSWTRG[1:0] Software Trigger Effect on TIOBx

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

Bits 29:28 – BEEVT[1:0] External Event Effect on TIOBx

Value	Name	Description
0	NONE	None
1	SET	Set
2	CLEAR	Clear
3	TOGGLE	Toggle

Bits 27:26 – BCPC[1:0] RC Compare Effect on TIOBx

Value	Name	Description
0	NONE	None
1	SET	Set

54.7.5 ACC Interrupt Mask Register

Name:	ACC_IMR
Offset:	0x2C
Reset:	0x00000000
Property:	Read-only

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

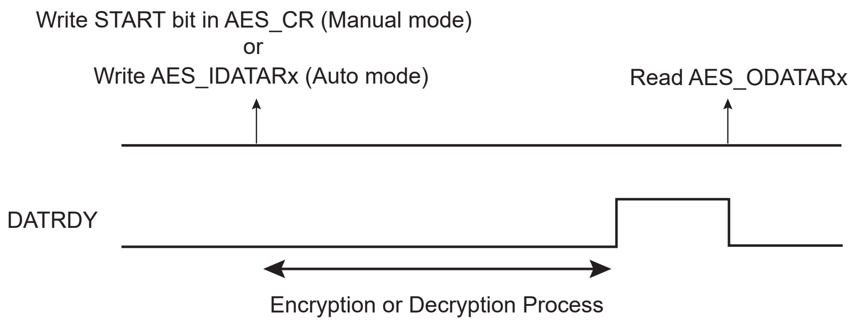
Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
								CE
Access								R
Reset								0

Bit 0 – CE Comparison Edge

Value	Description
0	The interrupt is disabled.
1	The interrupt is enabled.

Figure 57-1. Manual and Auto Modes with AES_MR.LOD = 0


Write START bit in AES_CR (Manual mode)
or
Write AES_IDATARx (Auto mode)
Read AES_ODATARx
DATRDY
Encryption or Decryption Process

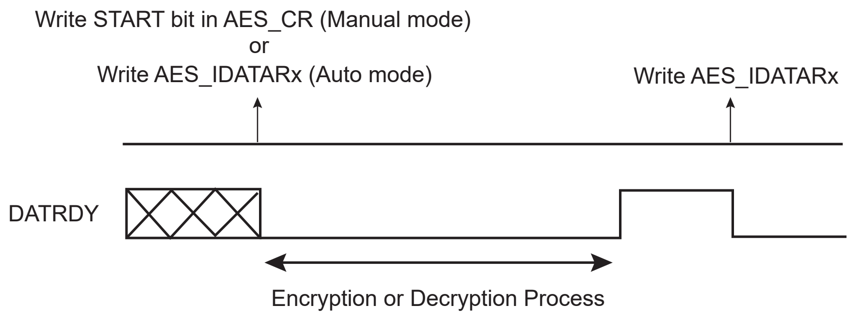
If the user does not want to read AES_ODATARx between each encryption/decryption, the DATRDY flag will not be cleared. If the DATRDY flag is not cleared, the user cannot know the end of the following encryptions/decryptions.

57.4.3.1.2 If AES_MR.LOD = 1

This mode is optimized to process AES CBC-MAC operating mode.

The DATRDY flag is cleared when at least one AES_IDATAR is written (see the figure below). No additional AES_ODATAR reads are necessary between consecutive encryptions/decryptions.

Figure 57-2. Manual and Auto Modes with AES_MR.LOD = 1


Write START bit in AES_CR (Manual mode)
or
Write AES_IDATARx (Auto mode)
Write AES_IDATARx
DATRDY
Encryption or Decryption Process

57.4.3.2 DMA Mode

57.4.3.2.1 If AES_MR.LOD = 0

This mode may be used for all AES operating modes except CBC-MAC where AES_MR.LOD = 1 mode is recommended.

The end of the encryption/decryption is indicated by the end of DMA transfer associated to AES_ODATARx (see the figure below). Two DMA channels are required: one for writing message blocks to AES_IDATARx and one to obtain the result from AES_ODATARx.

Figure 57-3. DMA Transfer with AES_MR.LOD = 0

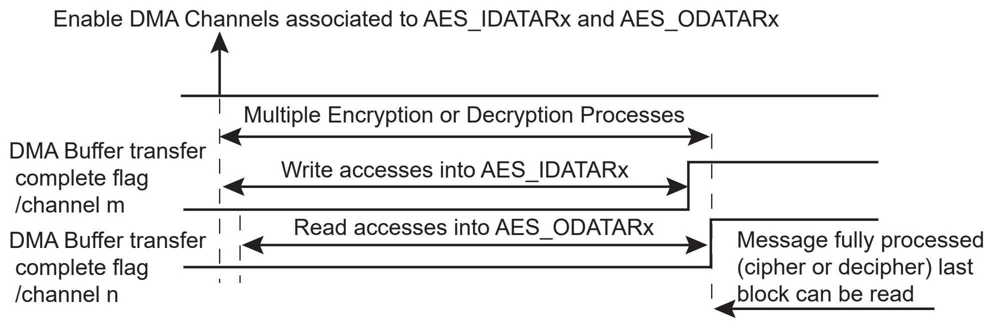

Enable DMA Channels associated to AES_IDATARx and AES_ODATARx
Multiple Encryption or Decryption Processes
DMA Buffer transfer complete flag /channel m
Write accesses into AES_IDATARx
DMA Buffer transfer complete flag /channel n
Read accesses into AES_ODATARx
Message fully processed (cipher or decipher) last block can be read

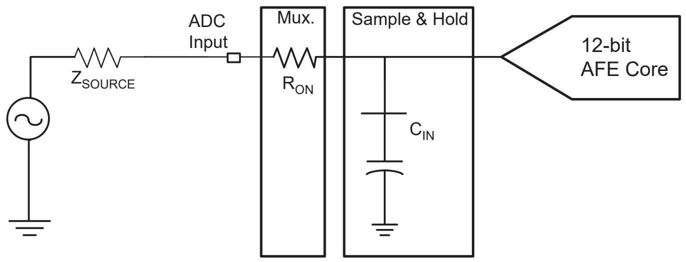
Table 58-38. Z_{IN} Input Impedance

f_S (MHz)	1	0.5	0.25	0.125	0.0625	0.03125	0.015625	0.007813
C_{IN} = 2 pF								
Z_{IN} (MΩ)	0.5	1	2	4	8	16	32	64
C_{IN} = 4 pF								
Z_{IN} (MΩ)	0.25	0.5	1	2	4	8	16	32
C_{IN} = 8 pF								
Z_{IN} (MΩ)	0.125	0.25	0.5	1	2	4	8	16

58.8.6.1 Track and Hold Time versus Source Output Impedance

The figure below shows a simplified acquisition path.

Figure 58-16. Simplified Acquisition Path



During the tracking phase, the AFE tracks the input signal during the tracking time shown below:

$t_{TRACK} = n \times C_{IN} \times (R_{ON}+Z_{SOURCE})/1000$

- Tracking time expressed in ns and Z_{SOURCE} expressed in Ω.
- n depends on the expected accuracy
- R_{ON}= 2 kOhm

Table 58-39. Number of Tau:n

Resolution (bits)	12	13	14	15	16
RES	0	2	3	4	5
n	8	9	10	11	12

The AFEC already includes a tracking time of 15 $t_{AFE\ Clock.}$

58.8.6.2 AFE DAC Offset Compensation

Table 58-40. DAC Static Performances (see Note 1)

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
N	Resolution (see **Note 2**)	–	–	9	10	LSB
INL	Integral Non Linearity	–	-2.5	±0.7	2	LSB
DNL	Differential Non Linearity	–	-3	±0.5	1.8	LSB

Note:

1. DAC Offset is included in the AFE EO performances.