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	Active
Core Processor	ARM® Cortex®-M7
Core Size	32-Bit Single-Core
Speed	300MHz
Connectivity	CANbus, Ethernet, 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	75
Program Memory Size	1MB (1M x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	384K x 8
Voltage - Supply (Vcc/Vdd)	1.62V ~ 3.6V
Data Converters	A/D 10x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	100-TFBGA
Supplier Device Package	100-TFBGA (9x9)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsame70n20b-cn

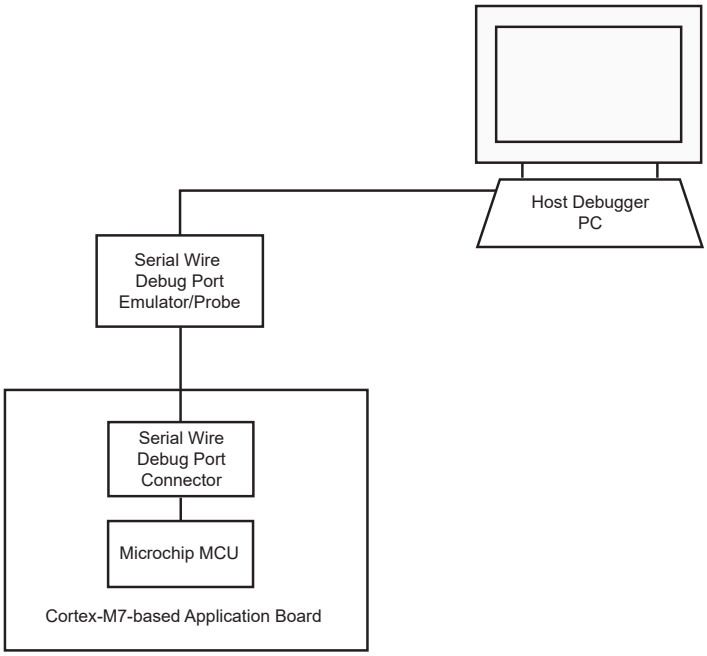
Signal Name	Function	Type	Active Level
TRACECLK	Trace Clock	Output	–
TRACED0–3	Trace Data	Output	–

16.6 Application Examples

16.6.1 Debug Environment

The figure below shows a complete debug environment example. The SW-DP interface is used for standard debugging functions, such as downloading code and single-stepping through the program and viewing core and peripheral registers.

Figure 16-2. Application Debug Environment Example



16.6.2 Test Environment

The figure below shows a test environment example (JTAG Boundary scan). Test vectors are sent and interpreted by the tester. In this example, the “board in test” is designed using a number of JTAG-compliant devices. These devices can be connected to form a single scan chain.

3. When programming is completed, the bit EEFC_FSR.FRDY rises. If an interrupt has been enabled by setting the bit EEFC_FMR.FRDY, the corresponding interrupt line of the interrupt controller is activated.

Two errors can be detected in EEFC_FSR after this sequence:

- Command Error: A bad keyword has been written in EEFC_FCR.
- Flash Error: At the end of the programming, the WriteVerify test of the Flash memory has failed.

The sequence to erase the user signature area is the following:

1. Execute the 'Erase User Signature' command by writing EEFC_FCR.FCMD with the EUS command. Field EEFC_FCR.FARG is meaningless.
2. When programming is completed, the bit EEFC_FSR.FRDY rises. If an interrupt has been enabled by setting the bit EEFC_FMR.FRDY, the corresponding interrupt line of the interrupt controller is activated.

Two errors can be detected in EEFC_FSR after this sequence:

- Command Error: A bad keyword has been written in EEFC_FCR.
- Flash Error: At the end of the programming, the EraseVerify test of the Flash memory has failed.

22.4.3.10 ECC Errors and Corrections

The Flash embeds an ECC module able to correct one unique error and able to detect two errors. The errors are detected while a read access is performed into memory array and stored in EEFC_FSR (see "EEFC Flash Status Register"). The error report is kept until EEFC_FSR is read.

There is one flag for a unique error on lower half part of the Flash word (64 LSB) and one flag for the upper half part (MSB). The multiple errors are reported in the same way.

Due to the anticipation technique to improve bandwidth throughput on instruction fetch, a reported error can be located in the next sequential Flash word compared to the location of the instruction being executed, which is located in the previously fetched Flash word.

If a software routine processes the error detection independently from the main software routine, the entire Flash located software must be rewritten because there is no storage of the error location.

If only a software routine is running to program and check pages by reading EEFC_FSR, the situation differs from the previous case. Performing a check for ECC unique errors just after page programming completion involves a read of the newly programmed page. This read sequence is viewed as data accesses and is not optimized by the Flash controller. Thus, in case of unique error, only the current page must be reprogrammed.

22.4.4 Register Write Protection

To prevent any single software error from corrupting EEFC behavior, certain registers in the address space can be write-protected by setting the WPEN bit in the "EEFC Write Protection Mode Register" (EEFC_WPMR).

The following register can be write-protected:

- "EEFC Flash Mode Register"

31.20.18 PMC Fast Startup Mode Register

Name: PMC_FSMR
Offset: 0x0070
Reset: 0x00000000
Property: Read/Write

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

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	FFLPM	FLPM[1:0]		LPM		USBAL	RTCAL	RTTAL
Access								
Reset	0	0	0	0		0	0	0

Bit	15	14	13	12	11	10	9	8
	FSTT15	FSTT14	FSTT13	FSTT12	FSTT11	FSTT10	FSTT9	FSTT8
Access								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FSTT7	FSTT6	FSTT5	FSTT4	FSTT3	FSTT2	FSTT1	FSTT0
Access								
Reset	0	0	0	0	0	0	0	0

Bit 23 – FFLPM Force Flash Low-power Mode

Value	Description
0	The Flash Low-power mode, defined in the FLPM field, is automatically applied when in Wait mode and released when going back to Active mode.
1	The Flash Low-power mode is user defined by the FLPM field and immediately applied.

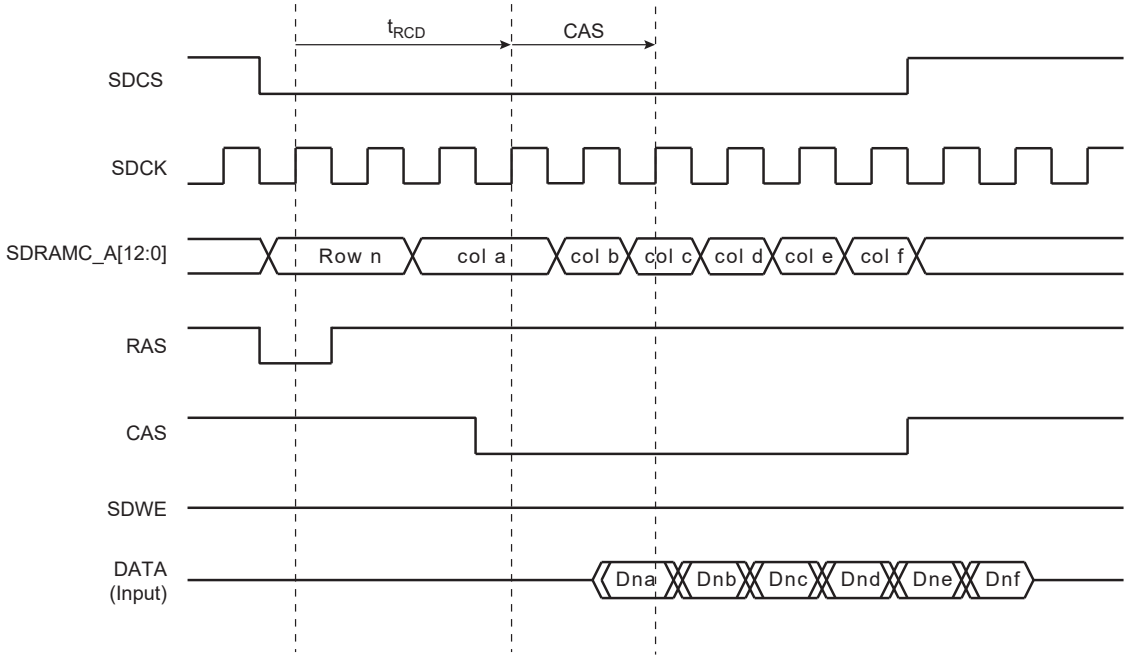
Bits 22:21 – FLPM[1:0] Flash Low-power Mode

Value	Name	Description
0	FLASH_STANDBY	Flash is in Standby Mode when system enters Wait Mode
1	FLASH_DEEP_POWERDOWN	Flash is in Deep-powerdown mode when system enters Wait Mode
2	FLASH_IDLE	Idle mode

Bit 20 – LPM Low-power Mode

Value	Description
0	The WaitForInterrupt (WFI) or the WaitForEvent (WFE) instruction of the processor makes the processor enter Sleep mode.
1	The WaitForEvent (WFE) instruction of the processor makes the system enter Wait mode.

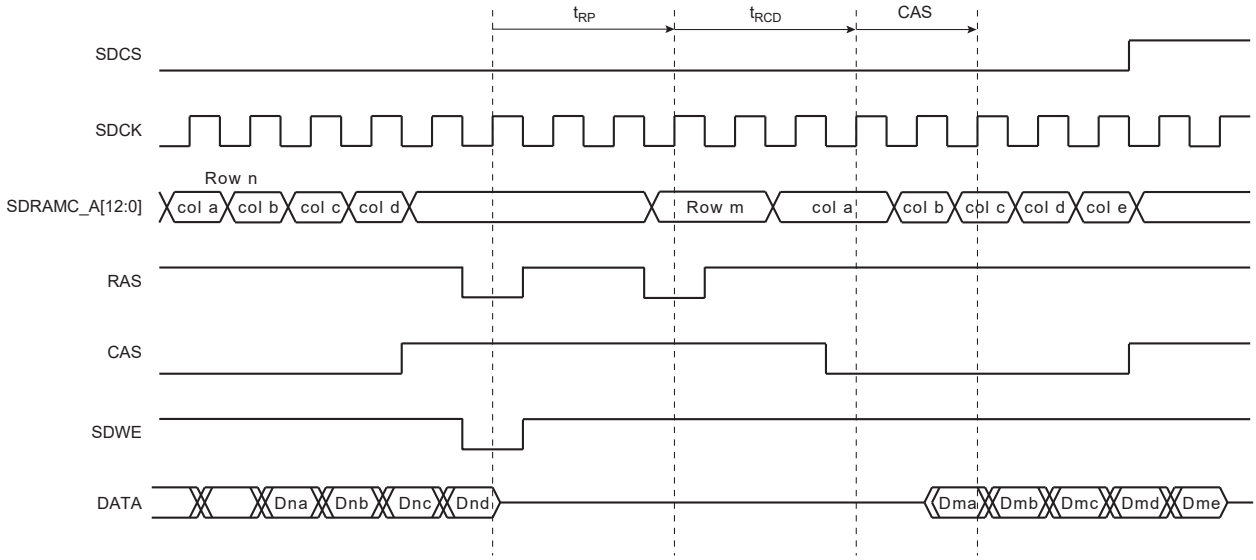
Figure 34-3. Read Burst SDRAM Access



34.6.3 Border Management

When the memory row boundary has been reached, an automatic page break is inserted. In this case, the SDRAMC generates a precharge command, activates the new row and initiates a read or write command. To comply with SDRAM timing parameters, an additional clock cycle is inserted between the precharge and the active command (t_{RP}) and between the active and the read command (t_{RCD}). Refer to the following figure.

Figure 34-4. Read Burst with Boundary Row Access



34.6.4 SDRAM Controller Refresh Cycles

An autorefresh command is used to refresh the SDRAM device. Refresh addresses are generated internally by the SDRAM device and incremented after each autorefresh automatically. The SDRAMC

Figure 35-27. Write Access with NWAIT Assertion in Frozen Mode (SMC_MODE.EXNW_MODE = 10)

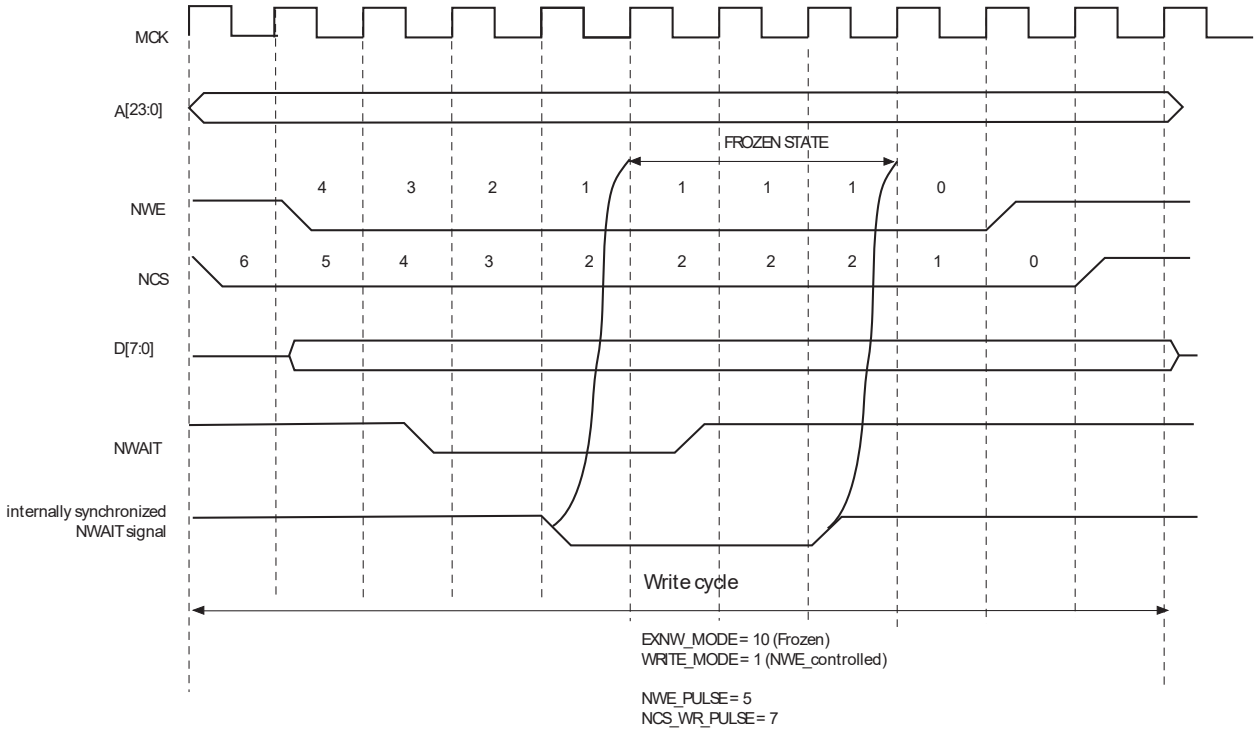
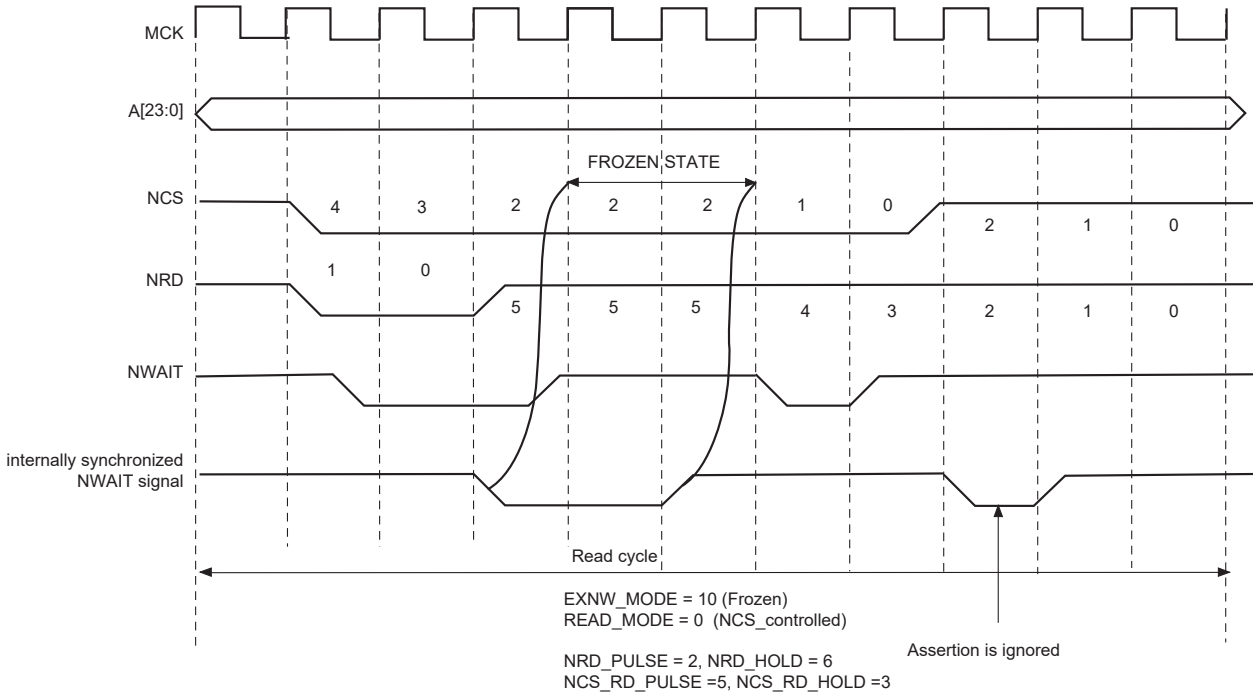


Figure 35-28. Read Access with NWAIT Assertion in Frozen Mode (SMC_MODE.EXNW_MODE = 10)



Peripheral Name	Transfer Type	HW Interface Number (XDMAC_CC.PERID)
I2SC0	Receive Left	45
I2SC1	Transmit Left	46
I2SC1	Receive Left	47
I2SC0	Transmit Right	48
I2SC0	Receive Right	49
I2SC1	Transmit Right	50
I2SC1	Receive Right	51

36.5 Functional Description

36.5.1 Basic Definitions

Source Peripheral: Slave device, memory mapped on the interconnection network, from where the XDMAC reads data. The source peripheral teams up with a destination peripheral to form a channel. A data read operation is scheduled when the peripheral transfer request is asserted.

Destination Peripheral: Slave device, memory mapped on the interconnection network, to which the XDMAC writes. A write data operation is scheduled when the peripheral transfer request is asserted.

Channel: The data movement between source and destination creates a logical channel.

Transfer Type: The transfer is hardware-synchronized when it is paced by the peripheral hardware request, otherwise the transfer is self-triggered (memory to memory transfer).

36.5.2 Transfer Hierarchy Diagram

XDMAC Master Transfer: The Master Transfer is composed of a linked list of blocks. The channel address, control and configuration registers can be modified at the inter block boundary. The descriptor structure modifies the channel registers conditionally. Interrupts can be generated on a per block basis or when the end of linked list event occurs.

XDMAC Block: An XDMAC block is composed of a programmable number of microblocks. The channel configuration registers remain unchanged at the inter microblock boundary. The source and destination addresses are conditionally updated with a programmable signed number.

XDMAC Microblock: The microblock is composed of a programmable number of data. The channel configuration registers remain unchanged at the data boundary. The data address may be fixed (a FIFO location, a peripheral transmit or receive register), incrementing (a memory-mapped area) by a programmable signed number.

XDMAC Burst and Incomplete Burst: In order to improve the overall performance when accessing dynamic external memory, burst access is mandatory. Each data of the microblock is considered as a part of a memory burst. The programmable burst value indicates the largest memory burst allowed on a per channel basis. When the microblock length is not an integral multiple of the burst size, an incomplete burst is performed to read or write the last trailing bytes.

XDMAC Chunk and Incomplete Chunk: When a peripheral synchronized transfer is activated, the microblock splits into a number of data chunks. The chunk size is programmable. The larger the chunk is,

$$\begin{bmatrix} Y \\ C_r \\ C_b \end{bmatrix} = \begin{bmatrix} C_0 & C_1 & C_2 \\ C_3 & -C_4 & -C_5 \\ -C_6 & -C_7 & C_8 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} Y_{\text{off}} \\ \text{Cr}_{\text{off}} \\ \text{Cb}_{\text{off}} \end{bmatrix}$$

An example of coefficients is given below:

$$\begin{cases} Y = 0.257 \cdot R + 0.504 \cdot G + 0.098 \cdot B + 16 \\ C_r = 0.439 \cdot R - 0.368 \cdot G - 0.071 \cdot B + 128 \\ C_b = -0.148 \cdot R - 0.291 \cdot G + 0.439 \cdot B + 128 \end{cases}$$

37.5.5.2 Memory Interface

Dedicated FIFOs are used to support packed memory mapping. YCrCb pixel components are sent in a single 32-bit word in a contiguous space (packed). Data is stored in the order of natural scan lines. Planar mode is not supported.

37.5.5.3 DMA Features

Like preview datapath, codec datapath DMA mode uses linked list operation.

37.6.22 DMA Codec Control Register

Name: ISI_DMA_C_CTRL
Offset: 0x54
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
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					C_DONE	C_IEN	C_WB	C_FETCH
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – C_DONE Codec Transfer Done
This bit is only updated in the memory.

Value	Description
0	The transfer related to this descriptor has not been performed.
1	The transfer related to this descriptor has completed. This bit is updated in memory at the end of the transfer when writeback operation is enabled.

Bit 2 – C_IEN Transfer Done Flag Control

Value	Description
0	Codec transfer done flag generation is enabled.
1	Codec transfer done flag generation is disabled.

Bit 1 – C_WB Descriptor Writeback Control Bit

Value	Description
0	Codec channel writeback operation is disabled.
1	Codec channel writeback operation is enabled.

Bit 0 – C_FETCH Descriptor Fetch Control Bit

38.8.79 GMAC Receive Overruns Register

Name: GMAC_ROE
Offset: 0x1A4
Reset: 0x00000000
Property: -

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
							RXOVR[9:8]	
Access							R	R
Reset							0	0

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

Bits 9:0 – RXOVR[9:0] Receive Overruns
This bit field counts the number of frames that are address recognized but were not copied to memory due to a receive overrun.

the Upstream Resume Received Interrupt (USBHS_HSTISR.RXRSMI) bit is set. The user has to generate a Downstream Resume within 1 ms and for at least 20 ms by writing a one to the Send USB Resume (USBHS_HSTCTRL.RESUME) bit. It is mandatory to write a one to USBHS_HSTCTRL.SOFE before writing a one to USBHS_HSTCTRL.RESUME to enter the Ready state, otherwise USBHS_HSTCTRL.RESUME has no effect.

39.5.3.9 Management of Control Pipes

A control transaction is composed of three stages:

- SETUP
- Data (IN or OUT)
- Status (OUT or IN)

The user has to change the pipe token according to each stage.

For the control pipe only, each token is assigned a specific initial data toggle sequence:

- SETUP: Data0
- IN: Data1
- OUT: Data1

39.5.3.10 Management of IN Pipes

IN packets are sent by the USB device controller upon IN requests from the host. All data which acknowledges or not the bank can be read when it is empty.

The pipe must be configured first.

When the host requires data from the device, the user has to first select the IN Request mode with the IN Request Mode bit in the Pipe x IN Request register (USBHS_HSTPIPINRQx.INMODE):

- When USBHS_HSTPIPINRQx.INMODE = 0, the USBHS performs (INRQ + 1) IN requests before freezing the pipe.
- When USBHS_HSTPIPINRQx.INMODE = 1, the USBHS performs IN requests endlessly when the pipe is not frozen by the user.

The generation of IN requests starts when the pipe is unfrozen (the Pipe Freeze (USBHS_HSTPIPIMRx.PFREEZE) field in USBHS_HSTPIPIMRx is zero).

The Received IN Data Interrupt (USBHS_HSTPIPISRx.RXINI) bit is set at the same time as the FIFO Control (USBHS_HSTPIPIMRx.FIFOCON) bit when the current bank is full. This triggers a PEP_x interrupt if the Received IN Data Interrupt Enable (USBHS_HSTPIPIMRx.RXINE) bit is one.

USBHS_HSTPIPISRx.RXINI is cleared by software (by writing a one to the Received IN Data Interrupt Clear bit in the Host Pipe x Clear register (USBHS_HSTPIPIDRx.RXINIC)) to acknowledge the interrupt, which has no effect on the pipe FIFO.

The user then reads from the FIFO and clears the USBHS_HSTPIPIMRx.FIFOCON bit (by writing a one to the FIFO Control Clear (USBHS_HSTPIPIDRx.FIFOCONC) bit) to free the bank. If the IN pipe is composed of multiple banks, this also switches to the next bank. The USBHS_HSTPIPISRx.RXINI and USBHS_HSTPIPIMRx.FIFOCON bits are updated in accordance with the status of the next bank.

USBHS_HSTPIPISRx.RXINI is always cleared before clearing USBHS_HSTPIPIMRx.FIFOCON.

The Read/Write Allowed (USBHS_HSTPIPISRx.RWALL) bit is set when the current bank is not empty, i.e., when the software can read further data from the FIFO.

Offset	Name	Bit Pos.								
0x0220	USBHS_DEVEPTIDR0	7:0	SHORTPACKETEC	STALLEDEC	OVERFEC	NAKINEC	NAKOUTEC	RXSTPEC	RXOUTEC	TXINEC
		15:8		FIFOCONC		NBUSYBKEC				
		23:16					STALLRQC		NYETDISC	EPDISHDMAC
		31:24								
0x0220	USBHS_DEVEPTIDR0 (ISOENPT)	7:0	SHORTPACKETEC	CRCERREC	OVERFEC	HBISOFLUSHEC	HBISOINERREC	UNDERFEC	RXOUTEC	TXINEC
		15:8		FIFOCONC		NBUSYBKEC		ERRORTRANSEC	DATAXEC	MDATEC
		23:16								EPDISHDMAC
		31:24								
0x0224	USBHS_DEVEPTIDR1	7:0	SHORTPACKETEC	STALLEDEC	OVERFEC	NAKINEC	NAKOUTEC	RXSTPEC	RXOUTEC	TXINEC
		15:8		FIFOCONC		NBUSYBKEC				
		23:16					STALLRQC		NYETDISC	EPDISHDMAC
		31:24								
0x0224	USBHS_DEVEPTIDR1 (ISOENPT)	7:0	SHORTPACKETEC	CRCERREC	OVERFEC	HBISOFLUSHEC	HBISOINERREC	UNDERFEC	RXOUTEC	TXINEC
		15:8		FIFOCONC		NBUSYBKEC		ERRORTRANSEC	DATAXEC	MDATEC
		23:16								EPDISHDMAC
		31:24								
0x0228	USBHS_DEVEPTIDR2	7:0	SHORTPACKETEC	STALLEDEC	OVERFEC	NAKINEC	NAKOUTEC	RXSTPEC	RXOUTEC	TXINEC
		15:8		FIFOCONC		NBUSYBKEC				
		23:16					STALLRQC		NYETDISC	EPDISHDMAC
		31:24								
0x0228	USBHS_DEVEPTIDR2 (ISOENPT)	7:0	SHORTPACKETEC	CRCERREC	OVERFEC	HBISOFLUSHEC	HBISOINERREC	UNDERFEC	RXOUTEC	TXINEC
		15:8		FIFOCONC		NBUSYBKEC		ERRORTRANSEC	DATAXEC	MDATEC
		23:16								EPDISHDMAC
		31:24								
0x022C	USBHS_DEVEPTIDR3	7:0	SHORTPACKETEC	STALLEDEC	OVERFEC	NAKINEC	NAKOUTEC	RXSTPEC	RXOUTEC	TXINEC
		15:8		FIFOCONC		NBUSYBKEC				
		23:16					STALLRQC		NYETDISC	EPDISHDMAC
		31:24								
0x022C	USBHS_DEVEPTIDR3 (ISOENPT)	7:0	SHORTPACKETEC	CRCERREC	OVERFEC	HBISOFLUSHEC	HBISOINERREC	UNDERFEC	RXOUTEC	TXINEC

Figure 46-46. Master Node Configuration, NACT = SUBSCRIBE

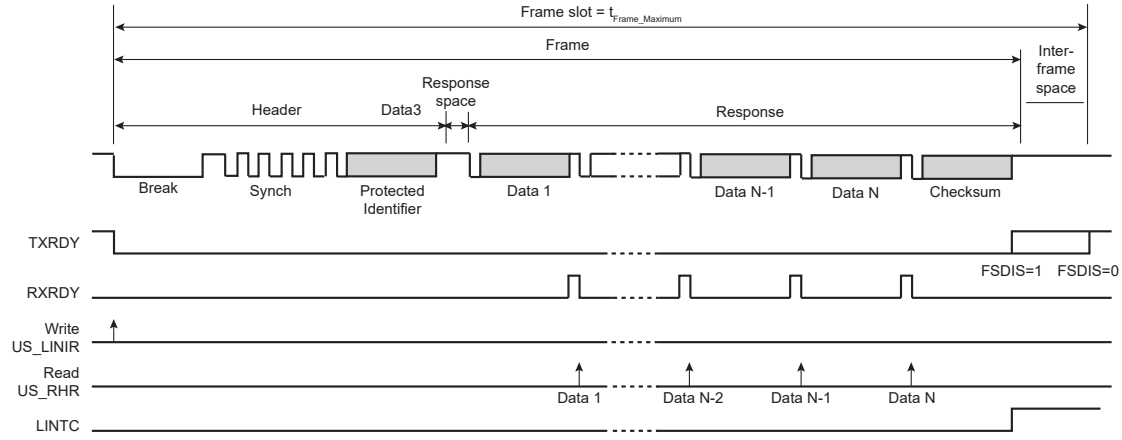
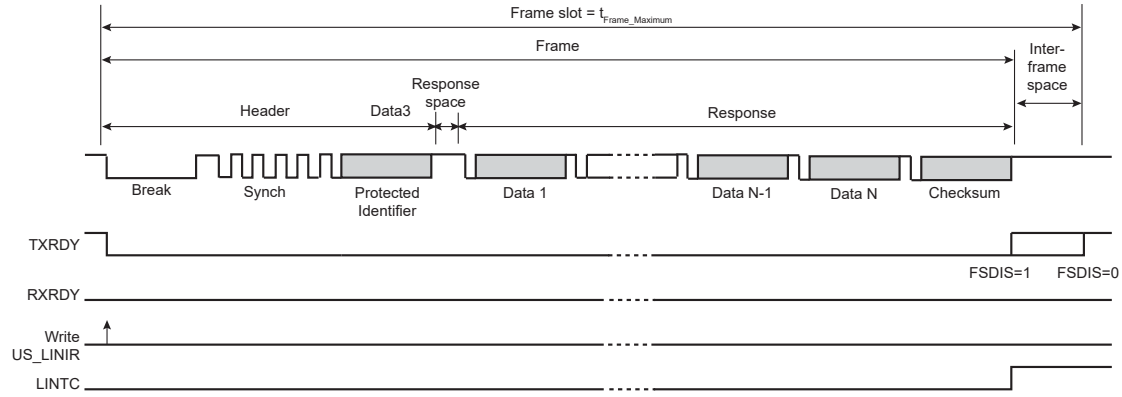


Figure 46-47. Master Node Configuration, NACT = IGNORE



46.6.9.15.2 Slave Node Configuration

- Write TXEN and RXEN in US_CR to enable both the transmitter and the receiver.
- Write USART_MODE in US_MR to select the LIN mode and the slave node configuration.
- Write CD and FP in US_BRGR to configure the baud rate.
- Wait until LINID in US_CSR rises.
- Check LINISFE and LINPE errors.
- Read IDCHR in US_RHR.
- Write NACT, PARDIS, CHKDIS, CHKTYPE, DLCM and DLC in US_LINMR to configure the frame transfer.

IMPORTANT: If the NACT configuration for this frame is PUBLISH, the US_LINMR must be written with NACT = PUBLISH even if this field is already correctly configured, in order to set the TXREADY flag and the corresponding write transfer request.

What comes next depends on the NACT configuration:

- Case 1: NACT = PUBLISH, the LIN controller sends the response
  - Wait until TXRDY in US_CSR rises.
  - Write TCHR in US_THR to send a byte.
  - If all the data have not been written, redo the two previous steps.
  - Wait until LINTC in US_CSR rises.
  - Check the LIN errors.

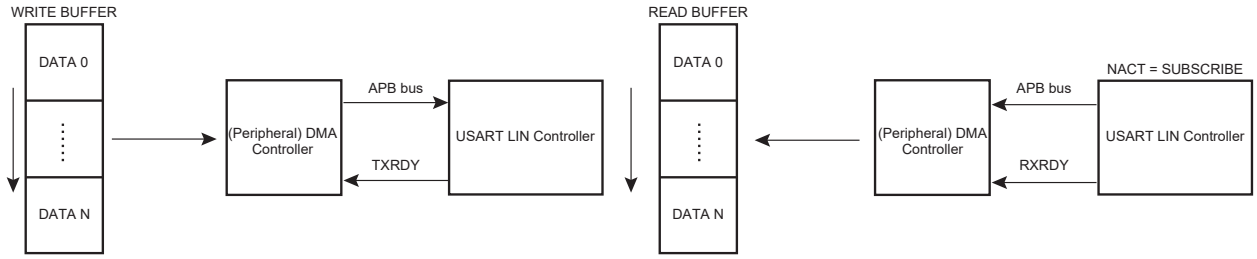
46.6.9.16.2 Slave Node Configuration

In this configuration, the DMAC transfers only the DATA. The Identifier must be read by the user in the LIN Identifier register (US_LINIR). The LIN mode must be written by the user in US_LINMR.

The WRITE buffer contains the DATA if the USART sends the response (NACT = PUBLISH).

The READ buffer contains the DATA if the USART receives the response (NACT = SUBSCRIBE).

Figure 46-53. Slave Node with DMAC



46.6.9.17 Wakeup Request

Any node in a sleeping LIN cluster may request a wakeup.

In the LIN 2.0 specification, the wakeup request is issued by forcing the bus to the dominant state from 250 µs to 5 ms. For this, it is necessary to send the character 0xF0 in order to impose five successive dominant bits. Whatever the baud rate is, this character complies with the specified timings.

- Baud rate min = 1 kbit/s -> t_{bit} = 1 ms -> 5 t_{bit} = 5 ms
- Baud rate max = 20 kbit/s -> t_{bit} = 50 µs -> 5 t_{bit} = 250 µs

In the LIN 1.3 specification, the wakeup request should be generated with the character 0x80 in order to impose eight successive dominant bits.

The user can choose by the WKUPTYP bit in US_LINMR either to send a LIN 2.0 wakeup request (WKUPTYP = 0) or to send a LIN 1.3 wakeup request (WKUPTYP = 1).

A wakeup request is transmitted by writing a ‘1’ to US_CR.LINWKUP. Once the transfer is completed, US_SR.LINTC flag is asserted. It is cleared by writing a ‘1’ to US_CR.RSTSTA.

46.6.9.18 Bus Idle Timeout

If the LIN bus is inactive for a certain duration, the slave nodes shall automatically enter in Sleep mode. In the LIN 2.0 specification, this timeout is fixed at 4 seconds. In the LIN 1.3 specification, it is fixed at 25,000 t_{bit}.

In slave Node configuration, the receiver timeout detects an idle condition on the RXD line. When a timeout is detected, US_CSR.TIMEOUT rises and can generate an interrupt, thus indicating to the driver to go into Sleep mode.

The timeout delay period (during which the receiver waits for a new character) is programmed in US_RTOR.TO. If a ‘0’ is written to TO, the Receiver Timeout is disabled and no timeout is detected. US_CSR.TIMEOUT remains at ‘0’. Otherwise, the receiver loads a 17-bit counter with the value programmed in TO. This counter is decremented at each bit period and reloaded each time a new character is received. If the counter reaches 0, US_CSR.TIMEOUT rises.

If US_CR.STTTO is written to ‘1’, the counter clock is stopped until a first character is received.

If US_CR.RETTO is written to ‘1’, the counter starts counting down immediately from the value TO.

49.6.36 MCAN Tx FIFO/Queue Status

Name: MCAN_TXFQS
Offset: 0xC4
Reset: 0x00000000
Property: Read-only

The Tx FIFO/Queue status is related to the pending Tx requests listed in register MCAN_TXBRP. Therefore the effect of Add/Cancellation requests may be delayed due to a running Tx scan (MCAN_TXBRP not yet updated).

In case of mixed configurations where dedicated Tx Buffers are combined with a Tx FIFO or a Tx Queue, the Put and Get Indices indicate the number of the Tx Buffer starting with the first dedicated Tx Buffers. Example: For a configuration of 12 dedicated Tx Buffers and a Tx FIFO of 20 Buffers a Put Index of 15 points to the fourth buffer of the Tx FIFO.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
			TFQF	TFQPI[4:0]				
Access			R	R	R	R	R	R
Reset			0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
				TFGI[4:0]				
Access				R	R	R	R	R
Reset				0	0	0	0	0

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

Bit 21 – TFQF Tx FIFO/Queue Full

Value	Description
0	Tx FIFO/Queue not full.
1	Tx FIFO/Queue full.

Bits 20:16 – TFQPI[4:0] Tx FIFO/Queue Put Index
Tx FIFO/Queue write index pointer, range 0 to 31.

Bits 12:8 – TFGI[4:0] Tx FIFO Get Index
Tx FIFO read index pointer, range 0 to 31. Read as zero when Tx Queue operation is configured (MCAN_TXBC.TFQM = ‘1’).

52.7.11 AFEC Interrupt Disable Register

Name: AFEC_IDR
Offset: 0x28
Reset: –
Property: Write-only

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

Bit	31	30	29	28	27	26	25	24
		TEMPCHG				COMPE	GOVRE	DRDY
Access		W				W	W	W
Reset		–				–	–	–

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
					EOC11	EOC10	EOC9	EOC8
Access					W	W	W	W
Reset					–	–	–	–

Bit	7	6	5	4	3	2	1	0
	EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0
Access	W	W	W	W	W	W	W	W
Reset	–	–	–	–	–	–	–	–

Bit 30 – TEMPCHG Temperature Change Interrupt Disable

Bit 26 – COMPE Comparison Event Interrupt Disable

Bit 25 – GOVRE General Overrun Error Interrupt Disable

Bit 24 – DRDY Data Ready Interrupt Disable

Bits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 – EOCx End of Conversion Interrupt Disable x

An interrupt is generated if the bit RHC[i] is written to 1 in the ICM_IER (if RHC[i] is set in ICM_RCTRL of region i) or if the bit REC[i] is written to 1 in the ICM_IER (if REC[i] is set in ICM_RCTRL of region i).

55.5.4.2 Processing Period

The SHA engine processing period can be configured.

The short processing period allows to allocate bandwidth to the SHA module whereas the long processing period allocates more bandwidth on the system bus to other applications.

In SHA mode, the shortest processing period is 85 clock cycles + 2 clock cycles for start command synchronization. The longest period is 209 clock cycles + 2 clock cycles.

In SHA256 and SHA224 modes, the shortest processing period is 72 clock cycles + 2 clock cycles for start command synchronization. The longest period is 194 clock cycles + 2 clock cycles.

55.5.5 ICM Automatic Monitoring Mode

ICM_CFG.ASCD is used to activate the ICM Automatic Monitoring mode. When ICM_CFG.ASCD is set and bits CDWBN and EOM in ICM.RCFG equal 0, the ICM performs the following actions:

1. The ICM passes through the Main List once to calculate the message digest of the monitored area.
2. When WRAP = 1 in ICM_RCFG, the ICM begins monitoring. CDWBN in ICM_RCFG is now automatically set and EOM is cleared. These bits have no effect during the monitoring period that ends when EOM is set.

55.5.6 Programming the ICM

Table 55-7. Region Attributes

Transfer Type		Main List	ICM_RCFG			ICM_RNEXT	Comments
			CDWBN	WRAP	EOM	NEXT	
Single Region	Contiguous list of blocks Digest written to memory Monitoring disabled	1 item	0	0	1	0	The Main List contains only one descriptor. The Secondary List is empty for that descriptor. The digest is computed and saved to memory.
	Non-contiguous list of blocks Digest written to memory Monitoring disabled	1 item	0	0	1	Secondary List address of the current region identifier	The Main List contains only one descriptor. The Secondary List describes the layout of the non-contiguous region.
	Contiguous list of blocks Digest comparison	1 item	1	1	0	0	When the hash computation is terminated, the

57.5.2 AES Mode Register

Name: AES_MR
Offset: 0x04
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
	CKEY[3:0]					CFBS[2:0]		
Access	W	W	W	W		R/W	R/W	R/W
Reset	0	0	0	–		0	0	0

Bit	15	14	13	12	11	10	9	8
	LOD	OPMOD[2:0]			KEYSIZE[1:0]		SMOD[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
	PROCDLY[3:0]				DUALBUFF		GTAGEN	CIPHER
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bits 23:20 – CKEY[3:0] Key

Value	Name	Description
0xE	PASSWD	This field must be written with 0xE the first time AES_MR is programmed. For subsequent programming of AES_MR, any value can be written, including that of 0xE. Always reads as 0.

Bits 18:16 – CFBS[2:0] Cipher Feedback Data Size

Value	Name	Description
0	SIZE_128BIT	128-bit
1	SIZE_64BIT	64-bit
2	SIZE_32BIT	32-bit
3	SIZE_16BIT	16-bit
4	SIZE_8BIT	8-bit

57.5.13 AES GCM Intermediate Hash Word Register x

Name: AES_GHASHRx
Offset: 0x78 + x*0x04 [x=0..3]
Reset: 0x00000000
Property: R/W

Bit	31	30	29	28	27	26	25	24
	GHASH[31:24]							
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
	GHASH[23:16]							
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
	GHASH[15:8]							
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
	GHASH[7:0]							
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 31:0 – GHASH[31:0] Intermediate GCM Hash Word x
The four 32-bit Intermediate Hash Word registers expose the intermediate GHASH value. May be read to save the current GHASH value so processing can later be resumed, presumably on a later message fragment. Whenever a new key is written to the AES Key Register two automatic actions are processed:

- GCM hash subkey generation
- AES_GHASHRx Clear

See Key Writing and Automatic Hash Subkey Calculation for details.

If an application software-specific hash initial value is needed for the GHASH, it must be written to AES_GHASHRx:

- after a write to the AES Key Register, if any,
- before starting the input data feed.

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{IH}	High-level Input Voltage	GPIO_MLB	1.80	–	V_{DDIO} + 0.3	V
		GPIO_AD, GPIO_CLK	2		V_{DDIO} + 0.3	
		GPIO, CLOCK, RST, TEST	V_{DDIO} x 0.7	–	V_{DDIO} + 0.3	
V_{OH}	High-level Output Voltage	GPIO_MLB	2	–	–	V
		GPIO_AD, GPIO, RST, TEST, CLOCK , Low drive	V_{DDIO} - 0.4	–	–	
		GPIO_AD, GPIO, RST, TEST, CLOCK , High drive	V_{DDIO} - 0.4	–	–	
		GPIO_CLK, Low drive	V_{DDIO} - 0.4			
		GPIO_CLK, High drive	V_{DDIO} - 0.4	–	–	
V_{OL}	Low-level Output Voltage	GPIO_MLB,	–	–	0.4	V
		GPIO_AD, GPIO, RST, TEST, CLOCK, Low drive	–	–	0.4	
		GPIO_AD, GPIO, RST, TEST, CLOCK, High drive	–	–	0.4	
		GPIO_CLK, Low drive	–	–	0.4	
		GPIO_CLK, High drive	–	–	0.4	
V_{hys}	Hysteresis Voltage	GPIO with Hysteresis mode enabled	150	–	–	mV
I_{IL}	Low-level Input Current	Pullup OFF	-1	–	1	µA
		Pullup ON	10	–	55	
I_{IH}	High-level Input Current	Pulldown OFF	-1	–	1	µA
		Pulldown ON	10	–	55	
R_{SERIAL}	Serial Resistor	GPIO_MLB	–	9	–	Ohm
		GPIO_AD, GPIO_CLK	–	14	–	
		GPIO, CLOCK, RST, TEST	–	26	–	

Table 59-5. Voltage Regulator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{DDOUT}	DC Output Voltage	Normal mode, I_{LOAD} = 100 mA	1.2	1.23	1.26	V
		Standby mode	–	0	–	
I_{LOAD}	Maximum DC Output Current		–	–	150	mA
C_{DIN}	Input Decoupling Capacitor	(1)	–	4.7	–	µF
C_{DOUT}	Output Decoupling Capacitor	(2)	–	1	–	µF

where:

- Z_{IN} is input impedance in Single-ended or Differential mode
- C_{IN} = 2 to 8 pF ±20% depending on the gain value and mode (SE or DIFF); temperature dependency is negligible
- R_{ON} is typical 2 kΩ and 8 kΩ max (worst case process and high temperature)

The following formula is used to calculate input impedance:

$$Z_{\mathrm{IN}} = \frac{1}{f_S \times C_{\mathrm{IN}}}$$

where:

- f_S is the sampling frequency of the AFE channel
- Typ values are used to compute AFE input impedance Z_{IN}

Table 59-37. Input Capacitance (C_{IN}) Values

Gain Selection	Single-ended	Differential	Unit
1	2	2	pF
2	4	4	
4	8	8	

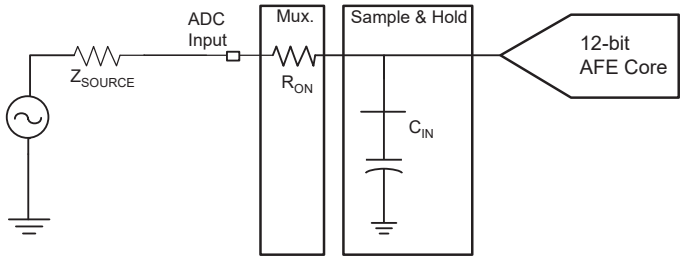
Table 59-38. Z_{IN} Input Impedance

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

59.8.6.1 Track and Hold Time versus Source Output Impedance

The figure below shows a simplified acquisition path.

Figure 59-16. 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.