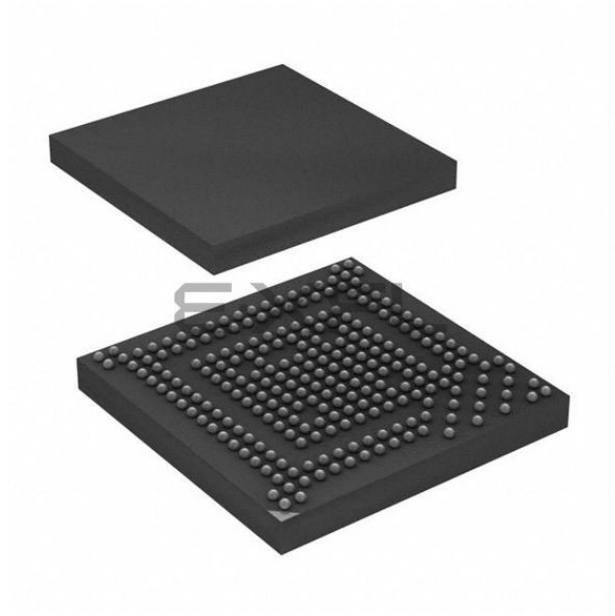


Welcome to E-XFL.COM

Understanding Embedded - Microprocessors

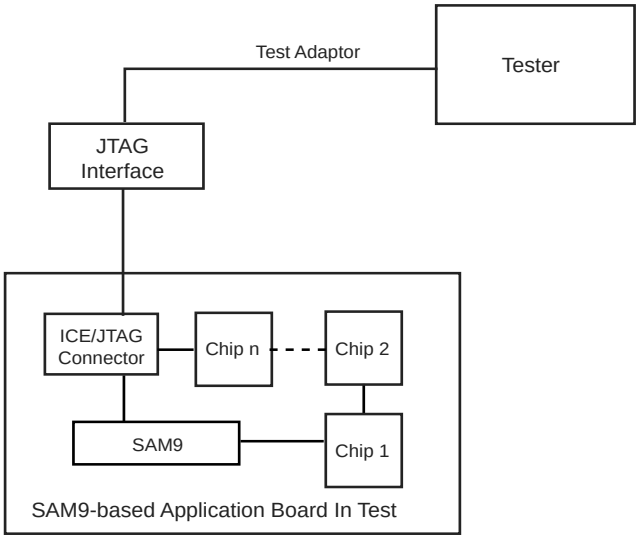
Embedded microprocessors are specialized computing chips designed to perform specific tasks within an embedded system. Unlike general-purpose microprocessors found in personal computers, embedded microprocessors are tailored for dedicated functions within larger systems, offering optimized performance,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

Applications of Embedded - Microprocessors

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

Details	
Product Status	Active
Core Processor	ARM926EJ-S
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	400MHz
Co-Processors/DSP	-
RAM Controllers	DDR2, SDRAM, SRAM
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	10/100Mbps
SATA	-
USB	USB 2.0 (3)
Voltage - I/O	1.8V, 3.3V
Operating Temperature	-40°C ~ 85°C (TA)
Security Features	-
Package / Case	247-LFBGA
Supplier Device Package	247-BGA (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at91sam9g25-cfu

Figure 9-3. Application Test Environment Example



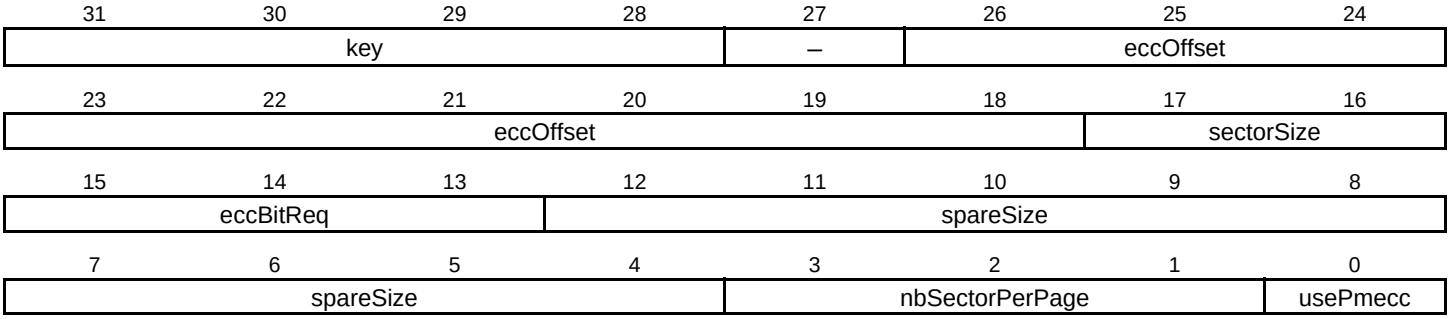
9.5 Debug and Test Pin Description

Table 9-1. Debug and Test Pin List

Pin Name	Function	Type	Active Level
Reset/Test			
NRST	Microcontroller Reset	Input/Output	Low
TST	Test Mode Select	Input	High
ICE and JTAG			
NTRST	Test Reset Signal	Input	Low
TCK	Test Clock	Input	
TDI	Test Data In	Input	
TDO	Test Data Out	Output	
TMS	Test Mode Select	Input	
RTCK	Returned Test Clock	Output	
JTAGSEL	JTAG Selection	Input	
Debug Unit			
DRXD	Debug Receive Data	Input	
DTXD	Debug Transmit Data	Output	

NAND Flash Specific Header Detection

This is the first method used to determine NAND Flash parameters. After Initialization and Reset command, the Boot Program reads the first page without ECC check, to determine if the NAND parameter header is present. The header is made of 52 times the same 32-bit word (for redundancy reasons) which must contain NAND and PMECC parameters used to correctly perform the read of the rest of the data in the NAND. This 32-bit word is described below:



- **usePmecc: Use PMECC**

0: Do not use PMECC to detect and correct the data.

1: Use PMECC to detect and correct the data.

- **nbSectorPerPage: Number of sectors per page**
- **spareSize: Size of the spare zone in bytes**
- **eccBitReq: Number of ECC bits required**
- **sectorSize: Size of the ECC sector**

0: 512 bytes

1: 1024 bytes per sector

Other value for future use.

- **eccOffset: Offset of the first ECC byte in the spare zone**

A value below 2 is not allowed and will be considered as 2.

- **key: value 0xC must be written here to validate the content of the whole word.**

If the header is valid, the Boot Program will continue with the detection of valid code.

ONFI 2.2 Parameters

In case no valid header has been found, the Boot Program will check if the NAND Flash is ONFI compliant, sending a Read Id command (0x90) with 0x20 as parameter for the address. If the NAND Flash is ONFI compliant, the Boot Program retrieves the following parameters with the help of the Get Parameter Page command:

- Number of bytes per page (byte 80)
- Number of bytes in spare zone (byte 84)
- Number of ECC bit correction required (byte 112)
- ECC sector size: by default set to 512 bytes, or 1024 bytes if the ECC bit capability above is 0xFF

By default, ONFI NAND Flash detection will turn ON the usePmecc parameter, and ECC correction algorithm is automatically activated.

12. Advanced Interrupt Controller (AIC)

12.1 Description

The Advanced Interrupt Controller (AIC) is an 8-level priority, individually maskable, vectored interrupt controller, providing handling of up to 32 interrupt sources. It is designed to substantially reduce the software and real-time overhead in handling internal and external interrupts.

The AIC drives the nFIQ (fast interrupt request) and the nIRQ (standard interrupt request) inputs of an ARM processor. Inputs of the AIC are either internal peripheral interrupts or external interrupts coming from the product's pins.

The 8-level Priority Controller allows the user to define the priority for each interrupt source, thus permitting higher priority interrupts to be serviced even if a lower priority interrupt is being treated.

Internal interrupt sources can be programmed to be level sensitive or edge triggered. External interrupt sources can be programmed to be positive-edge or negative-edge triggered or high-level or low-level sensitive.

The Fast Forcing feature redirects any internal or external interrupt source to provide a fast interrupt rather than a normal interrupt.

12.2 Embedded Characteristics

- Controls the Interrupt Lines (nIRQ and nFIQ) of an ARM® Processor
- 32 Individually Maskable and Vectored Interrupt Sources
  - Source 0 is Reserved for the Fast Interrupt Input (FIQ)
  - Source 1 is Reserved for System Peripherals
  - Source 2 to Source 31 Control up to 30 Embedded Peripheral Interrupts or External Interrupts
  - Programmable Edge-triggered or Level-sensitive Internal Sources
  - Programmable Positive/Negative Edge-triggered or High/Low Level-sensitive External Sources
- 8-level Priority Controller
  - Drives the Normal Interrupt of the Processor
  - Handles Priority of the Interrupt Sources 1 to 31
  - Higher Priority Interrupts Can Be Served During Service of Lower Priority Interrupt
- Vectoring
  - Optimizes Interrupt Service Routine Branch and Execution
  - One 32-bit Vector Register per Interrupt Source
  - Interrupt Vector Register Reads the Corresponding Current Interrupt Vector
- Protect Mode
  - Easy Debugging by Preventing Automatic Operations when Protect Models Are Enabled
- Fast Forcing
  - Permits Redirecting any Normal Interrupt Source to the Fast Interrupt of the Processor
- General Interrupt Mask
  - Provides Processor Synchronization on Events Without Triggering an Interrupt
- Register Write Protection

12.8.3.4 Interrupt Handlers

This section gives an overview of the fast interrupt handling sequence when using the AIC. It is assumed that the programmer understands the architecture of the ARM processor, and especially the processor interrupt modes and the associated status bits.

It is assumed that:

- The Advanced Interrupt Controller has been programmed, Source Vector registers are loaded with corresponding interrupt service routine addresses and interrupts are enabled.
- The instruction at the ARM interrupt exception vector address is required to work with the vectoring

```
LDR PC, [PC, # -&F20]
```

When nIRQ is asserted, if the bit “I” of CPSR is 0, the sequence is as follows:

1. The CPSR is stored in SPSR_irq, the current value of the Program Counter is loaded in the Interrupt link register (R14_irq) and the Program Counter (R15) is loaded with 0x18. In the following cycle during fetch at address 0x1C, the ARM core adjusts R14_irq, decrementing it by four.
2. The ARM core enters Interrupt mode, if it has not already done so.
3. When the instruction loaded at address 0x18 is executed, the program counter is loaded with the value read in AIC_IVR. Reading the AIC_IVR has the following effects:
   - Sets the current interrupt to be the pending and enabled interrupt with the highest priority. The current level is the priority level of the current interrupt.
   - De-asserts the nIRQ line on the processor. Even if vectoring is not used, AIC_IVR must be read in order to de-assert nIRQ.
   - Automatically clears the interrupt, if it has been programmed to be edge-triggered.
   - Pushes the current level and the current interrupt number on to the stack.
   - Returns the value written in the AIC_SVR corresponding to the current interrupt.
4. The previous step has the effect of branching to the corresponding interrupt service routine. This should start by saving the link register (R14_irq) and SPSR_IRQ. The link register must be decremented by four when it is saved if it is to be restored directly into the program counter at the end of the interrupt. For example, the instruction `SUB PC, LR, #4` may be used.
5. Further interrupts can then be unmasked by clearing the “I” bit in CPSR, allowing re-assertion of the nIRQ to be taken into account by the core. This can happen if an interrupt with a higher priority than the current interrupt occurs.
6. The interrupt handler can then proceed as required, saving the registers that will be used and restoring them at the end. During this phase, an interrupt of higher priority than the current level will restart the sequence from step 1.

Note: If the interrupt is programmed to be level sensitive, the source of the interrupt must be cleared during this phase.

7. The “I” bit in CPSR must be set in order to mask interrupts before exiting to ensure that the interrupt is completed in an orderly manner.
8. The AIC_EOICR must be written in order to indicate to the AIC that the current interrupt is finished. This causes the current level to be popped from the stack, restoring the previous current level if one exists on the stack. If another interrupt is pending, with lower or equal priority than the old current level but with higher priority than the new current level, the nIRQ line is re-asserted, but the interrupt sequence does not immediately start because the “I” bit is set in the core. SPSR_irq is restored. Finally, the saved value of the link register is restored directly into the PC. This has the effect of returning from the interrupt to whatever was being executed before, and of loading the CPSR with the stored SPSR, masking or unmasking the interrupts depending on the state saved in SPSR_irq.

Note: The “I” bit in SPSR is significant. If it is set, it indicates that the ARM core was on the verge of masking an interrupt when the mask instruction was interrupted. Hence, when SPSR is restored, the mask instruction is completed (interrupt is masked).

15.5.4 Periodic Interval Timer Image Register

Name: PIT_PIIR

Address: 0xFFFFFE3C

Access: Read-only

31	30	29	28	27	26	25	24
PICNT							

23	22	21	20	19	18	17	16
PICNT				CPIV			

15	14	13	12	11	10	9	8
CPIV							

7	6	5	4	3	2	1	0
CPIV							

- **CPIV: Current Periodic Interval Value**

Returns the current value of the periodic interval timer.

- **PICNT: Periodic Interval Counter**

Returns the number of occurrences of periodic intervals since the last read of PIT_PIVR.

22.6.40 PIO Level Select Register

Name: PIO_LSR

Address: 0xFFFFF4C4 (PIOA), 0xFFFFF6C4 (PIOB), 0xFFFFF8C4 (PIOC), 0xFFFFFAC4 (PIOD)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0–P31: Level Interrupt Selection**

0: No effect.

1: The interrupt source is a level-detection event.

22.6.45 PIO Write Protection Mode Register

Name: PIO_WPMR

Address: 0xFFFFF4E4 (PIOA), 0xFFFFF6E4 (PIOB), 0xFFFFF8E4 (PIOC), 0xFFFFFAE4 (PIOD)

Access: Read/Write

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

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x50494F (“PIO” in ASCII).

1: Enables the write protection if WPKEY corresponds to 0x50494F (“PIO” in ASCII).

See Section 22.5.15 “Register Write Protection” for the list of registers that can be protected.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x50494F	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

25. External Bus Interface (EBI)

25.1 Description

The External Bus Interface (EBI) is designed to ensure the successful data transfer between several external devices and the embedded memory controller of an ARM-based device.

The Static Memory, DDR, SDRAM and ECC controllers are all featured external memory controllers on the EBI. These external memory controllers are capable of handling several types of external memory and peripheral devices, such as SRAM, PROM, EPROM, EEPROM, Flash, DDR2 and SDRAM. The EBI operates with a 1.8V or 3.3V power supply (VDDIOM).

The EBI also supports the NAND Flash protocols via integrated circuitry that greatly reduces the requirements for external components. Furthermore, the EBI handles data transfers with up to six external devices, each assigned to six address spaces defined by the embedded memory controller. Data transfers are performed through a 16-bit or 32-bit data bus, an address bus of up to 26 bits, up to six chip select lines (NCS[5:0]) and several control pins that are generally multiplexed between the different external memory controllers.

25.2 Embedded Characteristics

- Integrates three External Memory Controllers:
  - Static Memory Controller
  - DDR2/SDRAM Controller
  - 8-bit NAND Flash ECC Controller
- Up to 26-bit Address Bus (up to 64 Mbytes linear per chip select)
- Up to 6 chip selects, Configurable Assignment:
  - Static Memory Controller on NCS0, NCS1, NCS2, NCS3, NCS4, NCS5
  - DDR2/SDRAM Controller (SDCS) or Static Memory Controller on NCS1
  - NAND Flash support on NCS3

Table 29-7. Interleaved Mapping for SDRAM Configuration: 8K Rows, 512/1024/2048/4096 Columns

CPU Address Line																										
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
			Row[12:0]													Bk[1:0]		Column[8:0]									M0
		Row[12:0]													Bk[1:0]		Column[9:0]										M0
	Row[12:0]													Bk[1:0]		Column[10:0]											M0
Row[12:0]													Bk[1:0]		Column[11:0]												M0

Table 29-8. Interleaved Mapping for SDRAM Configuration: 16K Rows, 512/1024/2048 Columns

CPU Address Line																										
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
		Row[13:0]														Bk[1:0]		Column[8:0]									M0
	Row[13:0]														Bk[1:0]		Column[9:0]										M0
Row[13:0]														Bk[1:0]		Column[10:0]											M0

29.6.2 SDRAM Address Mapping for 16-bit Memory Data Bus Width and Eight Banks

Table 29-9. Linear Mapping for SDRAM Configuration: 8K Rows, 1024 Columns

CPU Address Line																										
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
	Bk[2:0]			Row[12:0]													Column[9:0]										M0

Table 29-10. Linear Mapping for SDRAM Configuration: 16K Rows, 1024 Columns

CPU Address Line																										
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
Bk[2:0]			Row[13:0]														Column[9:0]										M0

Table 29-11. Interleaved Mapping for SDRAM Configuration: 8K Rows, 1024 Columns

CPU Address Line																										
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
	Row[12:0]													Bk[2:0]			Column[9:0]										M0

Table 29-12. Interleaved Mapping for SDRAM Configuration: 16K Rows, 1024 Columns

CPU Address Line																										
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
Row[12:0]														Bk[2:0]			Column[9:0]										M0

Figure 30-2. DMAC Transfer Hierarchy for Non-Memory Peripheral

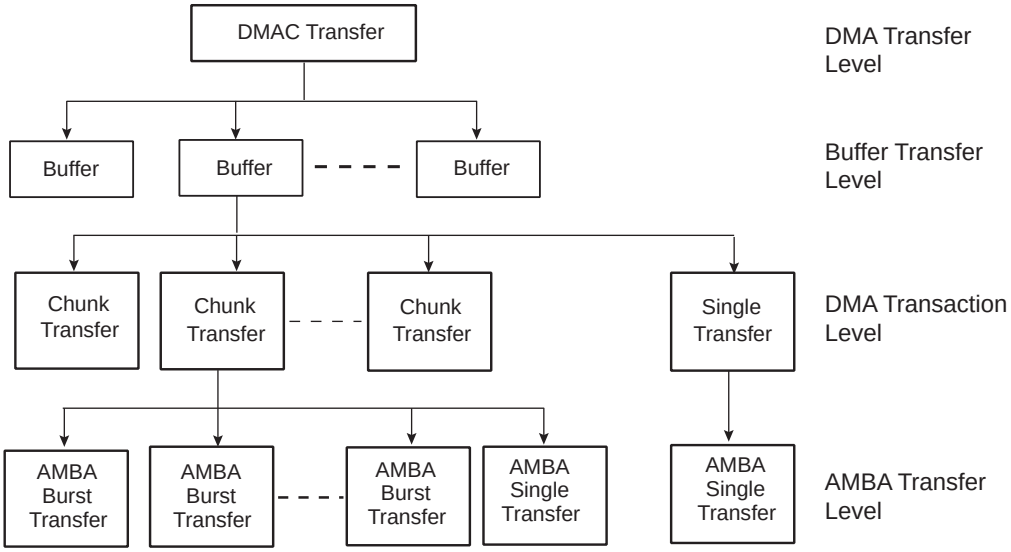
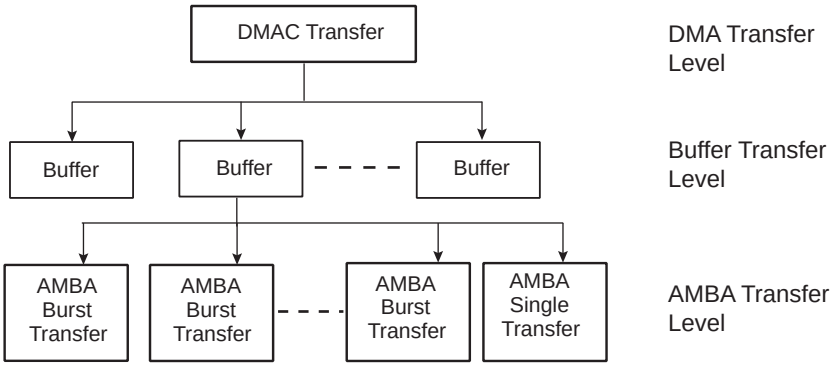


Figure 30-3. DMAC Transfer Hierarchy for Memory



Buffer: A buffer of DMAC data. The amount of data (length) is determined by the flow controller. For transfers between the DMAC and memory, a buffer is broken directly into a sequence of AMBA bursts and AMBA single transfers.

For transfers between the DMAC and a non-memory peripheral, a buffer is broken into a sequence of DMAC transactions (single and chunks). These are in turn broken into a sequence of AMBA transfers.

Transaction: A basic unit of a DMAC transfer as determined by either the hardware or software handshaking interface. A transaction is only relevant for transfers between the DMAC and a source or destination peripheral if the source or destination peripheral is a non-memory device. There are two types of transactions: single transfer and chunk transfer.

- **Single transfer:** The length of a single transaction is always 1 and is converted to a single AMBA access.
- **Chunk transfer:** The length of a chunk is programmed into the DMAC. The chunk is then converted into a sequence of AHB access.DMAC executes each AMBA burst transfer by performing incremental bursts that are no longer than 16 beats.

DMAC transfer: Software controls the number of buffers in a DMAC transfer. Once the DMAC transfer has completed, then hardware within the DMAC disables the channel and can generate an interrupt to signal the completion of the DMAC transfer. It is then possible to re-program the channel for a new DMAC transfer.

30.8.12 DMAC Channel Handler Status Register

Name: DMAC_CHSR

Address: 0xFFFFEC30 (0), 0xFFFFEE30 (1)

Access: Read-only

31	30	29	28	27	26	25	24
STAL7	STAL6	STAL5	STAL4	STAL3	STAL2	STAL1	STAL0
23	22	21	20	19	18	17	16
EMPT7	EMPT6	EMPT5	EMPT4	EMPT3	EMPT2	EMPT1	EMPT0
15	14	13	12	11	10	9	8
SUSP7	SUSP6	SUSP5	SUSP4	SUSP3	SUSP2	SUSP1	SUSP0
7	6	5	4	3	2	1	0
ENA7	ENA6	ENA5	ENA4	ENA3	ENA2	ENA1	ENA0

- **ENAx: Enable [7:0]**

A one in any position of this field indicates that the relevant channel is enabled.

- **SUSPx: Suspend [7:0]**

A one in any position of this field indicates that the channel transfer is suspended.

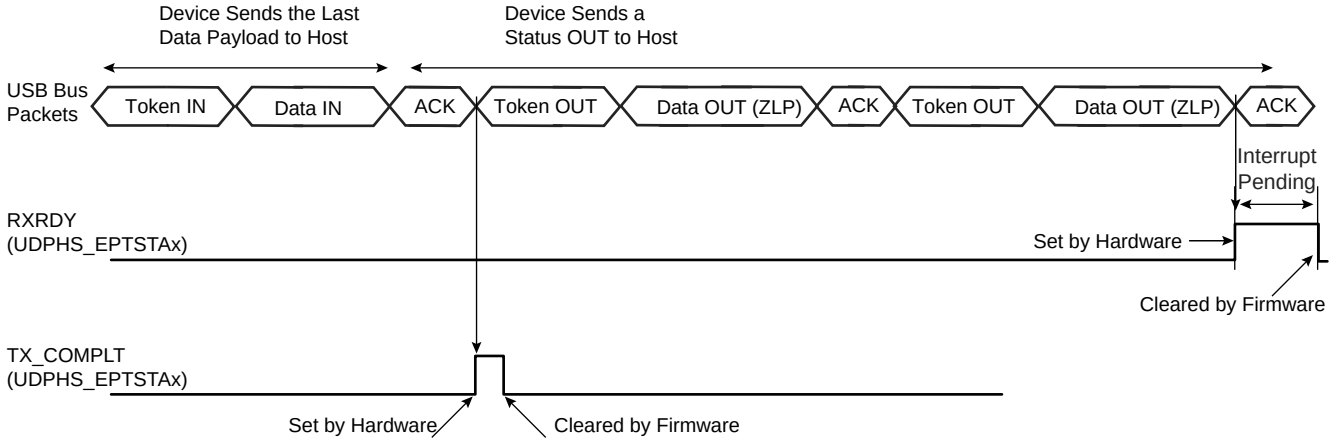
- **EMPTx: Empty [7:0]**

A one in any position of this field indicates that the relevant channel is empty.

- **STALx: Stalled [7:0]**

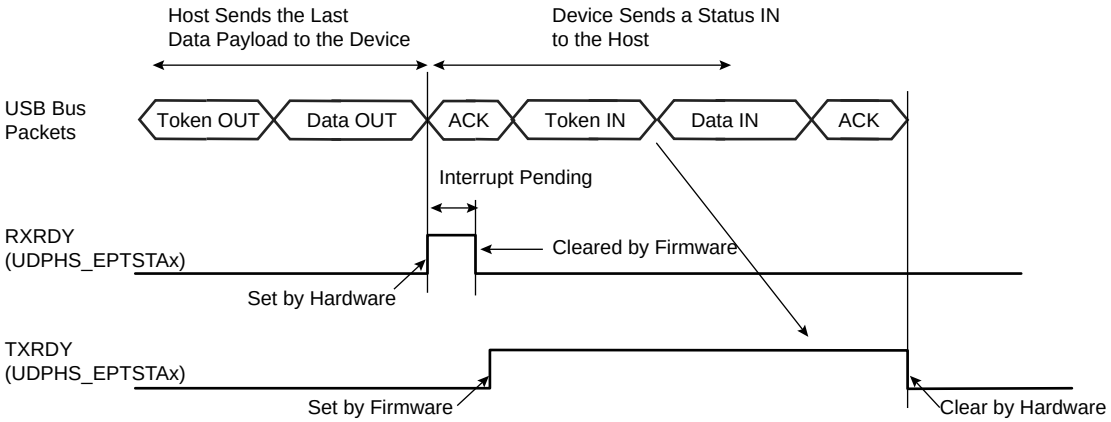
A one in any position of this field indicates that the relevant channel is stalling.

Figure 31-11. Data IN Followed By Status OUT Transfer at the End of a Control Transfer



Note: A NAK handshake is always generated at the first status stage token.

Figure 31-12. Data OUT Followed by Status IN Transfer



Note: Before proceeding to the status stage, the software should determine that there is no risk of extra data from the host (data stage). If not certain (non-predictable data stage length), then the software should wait for a NAK-IN interrupt before proceeding to the status stage. This precaution should be taken to avoid collision in the FIFO.

31.7.8 UDPHS Endpoint Configuration Register

Name: UDPHS_EPTCFGx [x=0..6]

Address: 0xF803C100 [0], 0xF803C120 [1], 0xF803C140 [2], 0xF803C160 [3], 0xF803C180 [4], 0xF803C1A0 [5], 0xF803C1C0 [6]

Access: Read/Write

31	30	29	28	27	26	25	24
EPT_MAPD	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	NB_TRANS	
7	6	5	4	3	2	1	0
BK_NUMBER		EPT_TYPE		EPT_DIR	EPT_SIZE		

- **EPT_SIZE: Endpoint Size (cleared upon USB reset)**

Set this field according to the endpoint size(1) in bytes (see Section 31.6.6 ”Endpoint Configuration”).

Value	Name	Description
0	8	8 bytes
1	16	16 bytes
2	32	32 bytes
3	64	64 bytes
4	128	128 bytes
5	256	256 bytes
6	512	512 bytes
7	1024	1024 bytes

Note: 1. 1024 bytes is only for isochronous endpoint.

- **EPT_DIR: Endpoint Direction (cleared upon USB reset)**

0: Clear this bit to configure OUT direction for Bulk, Interrupt and Isochronous endpoints.

1: Set this bit to configure IN direction for Bulk, Interrupt and Isochronous endpoints.

For Control endpoints this bit has no effect and should be left at zero.

- **EPT_TYPE: Endpoint Type (cleared upon USB reset)**

Set this field according to the endpoint type (see Section 31.6.6 ”Endpoint Configuration”).

(Endpoint 0 should always be configured as control)

Value	Name	Description
0	CTRL8	Control endpoint
1	ISO	Isochronous endpoint
2	BULK	Bulk endpoint
3	INT	Interrupt endpoint

31.7.18 UDPHS Endpoint Clear Status Register (Isochronous Endpoint)

Name: UDPHS_EPTCLRSTAx [x=0..6] (ISOENDPT)

Address: 0xF803C118 [0], 0xF803C138 [1], 0xF803C158 [2], 0xF803C178 [3], 0xF803C198 [4], 0xF803C1B8 [5], 0xF803C1D8 [6]

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	ERR_FLUSH	ERR_CRC_NTR	ERR_FL_ISO	–	TX_COMPLT	RXRDY_TXKL	–
7	6	5	4	3	2	1	0
–	TOGGLESQ	–	–	–	–	–	–

This register view is relevant only if EPT_TYPE = 0x1 in “UDPHS Endpoint Configuration Register” .

For additional information, see “UDPHS Endpoint Status Register (Isochronous Endpoint)” .

- **TOGGLESQ: Data Toggle Clear**

0: No effect.

1: Clear the PID data of the current bank

For OUT endpoints, the next received packet should be a DATA0.

For IN endpoints, the next packet will be sent with a DATA0 PID.

- **RXRDY_TXKL: Received OUT Data Clear**

0: No effect.

1: Clear the RXRDY_TXKL flag of UDPHS_EPTSTAx.

- **TX_COMPLT: Transmitted IN Data Complete Clear**

0: No effect.

1: Clear the TX_COMPLT flag of UDPHS_EPTSTAx.

- **ERR_FL_ISO: Error Flow Clear**

0: No effect.

1: Clear the ERR_FL_ISO flags of UDPHS_EPTSTAx.

- **ERR_CRC_NTR: Number of Transaction Error Clear**

0: No effect.

1: Clear the ERR_CRC_NTR flags of UDPHS_EPTSTAx.

- **ERR_FLUSH: Bank Flush Error Clear**

0: No effect.

1: Clear the ERR_FLUSH flags of UDPHS_EPTSTAx.

Figure 38-22. Timeguard Operations

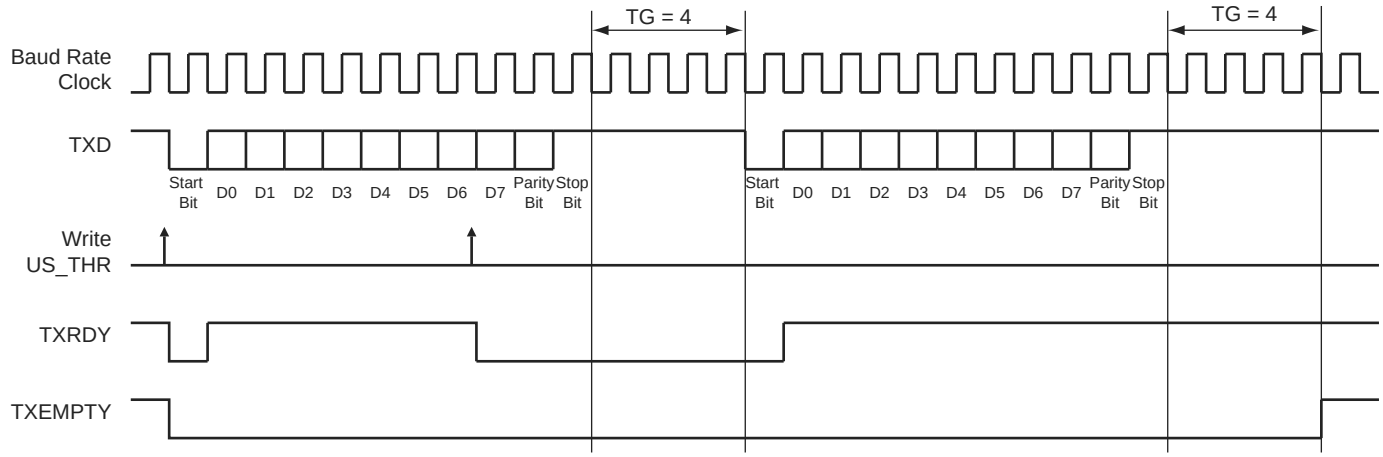


Table 38-9 indicates the maximum length of a timeguard period that the transmitter can handle in relation to the function of the baud rate.

Table 38-9. Maximum Timeguard Length Depending on Baud Rate

Baud Rate (bit/s)	Bit Time (µs)	Timeguard (ms)
1,200	833	212.50
9,600	104	26.56
14,400	69.4	17.71
19,200	52.1	13.28
28,800	34.7	8.85
38,400	26	6.63
56,000	17.9	4.55
57,600	17.4	4.43
115,200	8.7	2.21

38.6.3.11 Receiver Time-out

The Receiver Time-out provides support in handling variable-length frames. This feature detects an idle condition on the RXD line. When a time-out is detected, the bit TIMEOUT in the US_CSR rises and can generate an interrupt, thus indicating to the driver an end of frame.

The time-out delay period (during which the receiver waits for a new character) is programmed in the TO field of the Receiver Time-out register (US_RTOR). If the TO field is written to 0, the Receiver Time-out is disabled and no time-out is detected. The TIMEOUT bit in the US_CSR remains at 0. Otherwise, the receiver loads a 16-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, the TIMEOUT bit in US_CSR rises. Then, the user can either:

- Stop the counter clock until a new character is received. This is performed by writing a 1 to the STTTO (Start Time-out) bit in the US_CR. In this case, the idle state on RXD before a new character is received will not provide a time-out. This prevents having to handle an interrupt before a character is received and allows waiting for the next idle state on RXD after a frame is received.
- Obtain an interrupt while no character is received. This is performed by writing a 1 to the RETTO (Reload and Start Time-out) bit in the US_CR. If RETTO is performed, the counter starts counting down immediately from the value TO. This enables generation of a periodic interrupt so that a user time-out can be handled, for example when no key is pressed on a keyboard.

38.6.7.2 Baud Rate

In SPI mode, the baud rate generator operates in the same way as in USART Synchronous mode. See Section 38.6.1.3 ”Baud Rate in Synchronous Mode or SPI Mode”. However, there are some restrictions:

In SPI Master mode:

- The external clock SCK must not be selected (USCLKS ≠ 0x3), and the bit CLKO must be set to 1 in the US_MR, in order to generate correctly the serial clock on the SCK pin.
- To obtain correct behavior of the receiver and the transmitter, the value programmed in CD must be superior or equal to 6.
- If the divided peripheral clock is selected, the value programmed in CD must be even to ensure a 50:50 mark/space ratio on the SCK pin, this value can be odd if the peripheral clock is selected.

In SPI Slave mode:

- The external clock (SCK) selection is forced regardless of the value of the USCLKS field in the US_MR. Likewise, the value written in US_BRGR has no effect, because the clock is provided directly by the signal on the USART SCK pin.
- To obtain correct behavior of the receiver and the transmitter, the external clock (SCK) frequency must be at least 6 times lower than the system clock.

38.6.7.3 Data Transfer

Up to nine data bits are successively shifted out on the TXD pin at each rising or falling edge (depending of CPOL and CPHA) of the programmed serial clock. There is no Start bit, no Parity bit and no Stop bit.

The number of data bits is selected by the CHRL field and the MODE 9 bit in the US_MR. The nine bits are selected by setting the MODE 9 bit regardless of the CHRL field. The MSB data bit is always sent first in SPI mode (Master or Slave).

Four combinations of polarity and phase are available for data transfers. The clock polarity is programmed with the CPOL bit in the US_MR. The clock phase is programmed with the CPHA bit. These two parameters determine the edges of the clock signal upon which data is driven and sampled. Each of the two parameters has two possible states, resulting in four possible combinations that are incompatible with one another. Thus, a master/slave pair must use the same parameter pair values to communicate. If multiple slaves are used and fixed in different configurations, the master must reconfigure itself each time it needs to communicate with a different slave.

Table 38-13. SPI Bus Protocol Mode

SPI Bus Protocol Mode	CPOL	CPHA
0	0	1
1	0	0
2	1	1
3	1	0

43.6.5 ISI Color Space Conversion YCrCb to RGB Set 0 Register

Name: ISI_Y2R_SET0

Address: 0xF8048010

Access: Read/Write

31	30	29	28	27	26	25	24
C3							

23	22	21	20	19	18	17	16
C2							

15	14	13	12	11	10	9	8
C1							

7	6	5	4	3	2	1	0
C0							

- **C0: Color Space Conversion Matrix Coefficient C0**

C0 element default step is 1/128, ranges from 0 to 1.9921875.

- **C1: Color Space Conversion Matrix Coefficient C1**

C1 element default step is 1/128, ranges from 0 to 1.9921875.

- **C2: Color Space Conversion Matrix Coefficient C2**

C2 element default step is 1/128, ranges from 0 to 1.9921875.

- **C3: Color Space Conversion Matrix Coefficient C3**

C3 element default step is 1/128, ranges from 0 to 1.9921875.

43.6.7 ISI Color Space Conversion RGB to YCrCb Set 0 Register

Name: ISI_R2Y_SET0

Address: 0xF8048018

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	Roff

23	22	21	20	19	18	17	16
–	C2						

15	14	13	12	11	10	9	8
–	C1						

7	6	5	4	3	2	1	0
–	C0						

- **C0: Color Space Conversion Matrix Coefficient C0**

C0 element default step is 1/256, from 0 to 0.49609375.

- **C1: Color Space Conversion Matrix Coefficient C1**

C1 element default step is 1/128, from 0 to 0.9921875.

- **C2: Color Space Conversion Matrix Coefficient C2**

C2 element default step is 1/512, from 0 to 0.2480468875.

- **Roff: Color Space Conversion Red Component Offset**

0: No offset

1: Offset = 16

43.6.8 ISI Color Space Conversion RGB to YCrCb Set 1 Register

Name: ISI_R2Y_SET1

Address: 0xF804801C

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	Goff

23	22	21	20	19	18	17	16
–	C5						

15	14	13	12	11	10	9	8
–	C4						

7	6	5	4	3	2	1	0
–	C3						

- **C3: Color Space Conversion Matrix Coefficient C3**

C0 element default step is 1/128, ranges from 0 to 0.9921875.

- **C4: Color Space Conversion Matrix Coefficient C4**

C1 element default step is 1/256, ranges from 0 to 0.49609375.

- **C5: Color Space Conversion Matrix Coefficient C5**

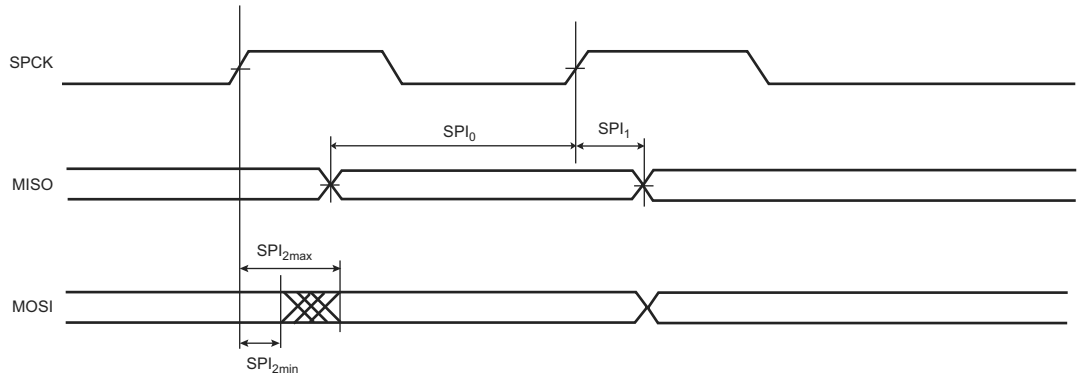
C1 element default step is 1/512, ranges from 0 to 0.2480468875.

- **Goff: Color Space Conversion Green Component Offset**

0: No offset.

1: Offset = 128.

Figure 45-15. Minimum and Maximum Access Time for SPI Output Signal



45.18.2 SSC

45.18.2.1 Timing Conditions

Timings are given assuming a capacitance load as defined in Table 45-40.

Table 45-40. Capacitance Load

Supply	Corner	
	Max	Min
3.3V (1)	30 pF	5 pF
1.8V (2)	20 pF	5 pF

Notes: 1. 3.3V domain: V_{DDIO} from 3.0V to 3.6V, maximum external capacitor = 30 pF.
2. 1.8V domain: V_{DDIO} from 1.65V to 1.95V, maximum external capacitor = 20 pF.

45.18.2.2 Timing Extraction

Figure 45-16. SSC Transmitter, TK and TF in Output